[MS-WSSFOB]:

Windows SharePoint Services (WSS): File Operations Database Communications Base Protocol

---

Revision Summary

Date	Revision History	Revision Class	Comments
6/17/2011	0.1	Major	Initial Availability
9/23/2011	0.1	None	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	0.1	None	No changes to the meaning, language, or formatting of the technical content.
3/30/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
7/12/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
9/12/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
2/11/2013	1.0	None	No changes to the meaning, language, or formatting of the technical content.
7/30/2013	1.0	None	No changes to the meaning, language, or formatting of the technical content.
11/18/2013	1.0	None	No changes to the meaning, language, or formatting of the technical content.
2/10/2014	1.1	Minor	Clarified the meaning of the technical content.
4/30/2014	1.2	Minor	Clarified the meaning of the technical content.
7/31/2014	1.3	Minor	Clarified the meaning of the technical content.
10/30/2014	1.4	Minor	Clarified the meaning of the technical content.
2/26/2016	2.0	Major	Significantly changed the technical content.
7/15/2016	2.0	None	No changes to the meaning, language, or formatting of the technical content.
9/14/2016	2.0	None	No changes to the meaning, language, or formatting of the technical content.
6/20/2017	3.0	Major	Significantly changed the technical content.
9/19/2017	3.0	None	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

The File Operations Database Communications Base Protocol specifies the File Operations Database Communications Base Protocol, the communication sequences used by protocol clients to perform data query and update commands on protocol servers as part of file, user, and group administration operations.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

access control list (ACL): A list of access control entries (ACEs) that collectively describe the security rules for authorizing access to some resource; for example, an object or set of objects.

Active Directory account creation mode: A type of account creation mode that retrieves and uses user accounts in a specific Active Directory Domain Services (AD DS) organizational unit.

alert: A message that is passed to a protocol client to notify it when specific criteria are met.

anonymous user: A user who presents no credentials when identifying himself or herself. The process for determining an anonymous user can differ based on the authentication protocol, and the documentation for the relevant authentication protocol should be consulted.

assembly name: The name of a collection of one or more files that is versioned and deployed as a unit. See also assembly.

attachment: An external file that is included with an Internet message or associated with an item in a SharePoint list.

author: The user who created a **list item**.

back-end database server: A server that hosts data, configuration settings, and stored procedures that are associated with one or more applications.

backward link: A hyperlink between a referenced document and a referencing party. For example, if Document A contains a hyperlink to Document B, then Document B has a backward link to Document A.

base type: An XML-based schema that defines the data and rendering fields that can be used in a **list (1)**. Every list is derived from a specific base type.

Boolean: An operation or expression that can be evaluated only as either true or false.

bot: A structured HTML comment that is processed by a front-end web server when the containing document is opened by or saved to the server. Also referred to as web bot.

character set: A mapping between the characters of a written language and the values that are used to represent those characters to a computer.

check out: The process of retrieving a writable copy of a file or project from a source repository. This locks the file for editing to prevent other users from overwriting or editing it inadvertently.

checked out: A publishing level that indicates that a document has been created and locked for exclusive editing by a user in a version control system.

class identifier (CLSID): A **GUID** that identifies a software component; for instance, a DCOM object class or a COM class.

Collaborative Application Markup Language (CAML): An XML-based language that is used to describe various elements, such as queries and views, in sites that are based on SharePoint Products and Technologies.

collation order: A rule for establishing a sequence for textual information.

Component Object Model (COM): An object-oriented programming model that defines how objects interact within a single process or between processes. In **COM**, clients have access to an object through interfaces implemented on the object. For more information, see [MS-DCOM].

configuration database: A database that is stored on a **back-end database server** and contains both persisted objects and site collection metadata for lookup purposes.

content database: A database that is stored on a **back-end database server** and contains stored procedures, site collections, and the contents of those site collections.

content type: A named and uniquely identifiable collection of settings and fields that store metadata for individual items in a SharePoint list. One or more content types can be associated with a list, which restricts the contents to items of those types.

Coordinated Universal Time (UTC): A high-precision atomic time standard that approximately tracks Universal Time (UT). It is the basis for legal, civil time all over the Earth. Time zones around the world are expressed as positive and negative offsets from UTC. In this role, it is also referred to as Zulu time (Z) and Greenwich Mean Time (GMT). In these specifications, all references to UTC refer to the time at UTC-0 (or GMT).

current user: The user who is authenticated during processing operations on a **front-end web server** or a **back-end database server**.

current version: The latest version of a document that is available to a user, based on the permissions of the user and the publishing level of the document.

default view: The layout and organization of a document or list that appears automatically when users open that document or display that list.

directory name: A segment of a **store-relative URL** that refers to a directory. A directory name is everything that appears before the last slash in a store-relative form URL.

directory service (DS): A service that stores and organizes information about a computer network's users and network shares, and that allows network administrators to manage users' access to the shares. See also Active Directory.

dirty: The condition of an entity, such as a component or a file, that indicates that the entity or properties of the entity were changed after the entity was last saved.

discussion board: A list in which users can read, post, and reply to messages from other users who are members of the same discussion board.

display name: A text string that is used to identify a principal or other object in the user interface. Also referred to as title.

displayed version: Document version information that is formatted for display in the user interface. The displayed version uses the format MajorVersion.MinorVersion, where MajorVersion is the published version number and MinorVersion is the draft version number, separated by a decimal point. See also major version and minor version.

document: An object in a **content database** such as a file, folder, **list (1)**, or **site**. Each object is identified by a URI.

document identifier: A GUID that identifies a document.

document library: A type of list that is a container for documents and folders.

document stream: A byte stream that is associated with a document, such as the content of a file. Some documents do not have document streams.

document template: A file that serves as the basis for new documents.

document version: A copy of a list item that has a version number. A document version can be either a historical version or a current version.

domain group: A container for security and distribution groups. A domain group can also contain other domain groups.

draft: A version of a document or list item that does not have a publishing level of "Published" or "Checked Out".

dynamic web template: An HTML-based master copy of a page that contains settings, formatting, and elements such as text, graphics, page layout, styles, and regions of a page that can be modified. Dynamic web templates have a .dwt file name extension.

email address: A string that identifies a user and enables the user to receive Internet messages.

empty string: A string object or variable that is initialized with the value "".

event: (1) Any significant occurrence in a system or an application that requires users to be notified or an entry to be added to a log.

(2) An action or occurrence to which an application might respond. Examples include state changes, data transfers, key presses, and mouse movements.

event receiver: A structured modular component that enables built-in or user-defined managed code classes to act upon objects, such as list items, **lists (1)**, or content types, when specific triggering actions occur.

event sink: A structured, modular component that enables built-in or user-defined classes to act on documents in document libraries when specific triggering actions occur. Event sinks are a deprecated, implementation-specific capability of Windows SharePoint Services 2.0. In Windows SharePoint Services 3.0 and Microsoft SharePoint Foundation 2010, they are replaced by the capabilities of event receivers.

feature: A package of SharePoint elements that can be activated or deactivated for a specific feature scope.

field: A container for metadata within a SharePoint list and associated list items.

field definition: The definition of a field in the **Collaborative Application Markup Language (CAML)**.

folder: A file system construct. File systems organize a volume's data by providing a hierarchy of objects, which are referred to as folders or directories, that contain files and can also contain other folders.

form: A document with a set of controls into which users can enter information. Controls on a form can be bound to elements in the data source of the form, such as fields and groups. See also bind.

forward link: A hyperlink between a referenced document and a referencing party. For example, if Document A contains a hyperlink to Document B, Document A has a forward link to Document B.

front-end web server: A server that hosts webpages, performs processing tasks, and accepts requests from protocol clients and sends them to the appropriate back-end server for further processing.

ghosted: A property that is not deleted by the server if the element is not included in a Sync <Change> request message.

globally unique identifier (GUID): A term used interchangeably with universally unique identifier (UUID) in Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the value. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] must be used for generating the **GUID**. See also universally unique identifier (UUID).

group: A named collection of users who share similar access permissions or roles.

hash: A fixed-size result that is obtained by applying a one-way mathematical function, which is sometimes referred to as a hash algorithm, to an arbitrary amount of data. If the input data changes, the hash also changes. The hash can be used in many operations, including authentication and digital signing.

host header: An Internet host and port number that identifies a network resource.

HTTP GET: An HTTP method for retrieving a resource, as described in [RFC2616].

HTTP HEAD: An HTTP method for retrieving header information for a resource, as described in [RFC2616].

Hypertext Markup Language (HTML): An application of the Standard Generalized Markup Language (SGML) that uses tags to mark elements in a document, as described in [HTML].

Hypertext Transfer Protocol (HTTP): An application-level protocol for distributed, collaborative, hypermedia information systems (text, graphic images, sound, video, and other multimedia files) on the World Wide Web.

Input Method Editor (IME): An application that is used to enter characters in written Asian languages by using a standard 101-key keyboard. An IME consists of both an engine that converts keystrokes into phonetic and ideographic characters and a dictionary of commonly used ideographic words.

Integrated Windows authentication: A configuration setting that enables negotiation of authentication protocols in Internet Information Services (IIS). Integrated Windows authentication is more secure than Basic authentication, because the user name and password are hashed instead of plaintext.

internal version number: A number that increases monotonically and is used to identify conflicts when saving an item.

item: A unit of content that can be indexed and searched by a search application.

item identifier: An integer that uniquely identifies an item in a SharePoint list.

language code identifier (LCID): A 32-bit number that identifies the user interface human language dialect or variation that is supported by an application or a client computer.

leaf name: The segment of a URL that follows the last slash. If the resource is a directory, the leaf name can be an **empty string**.

level: A relative position in a hierarchy of data. A level is frequently used when describing how to navigate a hierarchy in an Online Analytical Processing (OLAP) database or a PivotTable report.

link: An attribute value that refers to a directory object and whose Attribute-Schema object specifies an even value for the linkId attribute. Also referred to as forward link.

list: (1) A container within a SharePoint site that stores list items. A list has a customizable schema that is composed of one or more fields.

(2) An organization of a region of cells into a tabular structure in a workbook.

list item: An individual entry within a SharePoint list. Each list item has a schema that maps to fields in the list that contains the item, depending on the content type of the item.

list server template: A value that identifies the template that is used for a SharePoint list.

list template: An XML-based definition of list settings, including fields and views, and optionally list items. List templates are stored in .stp files in the content database.

list view: A named collection of settings for querying and displaying items in a SharePoint list. There are two types of views: Personal, which can be used only by the user who created the view; and Public, which can be used by all users who have permission to access to the site.

locale: A collection of rules and data that are specific to a language and a geographical area. A locale can include information about sorting rules, date and time formatting, numeric and monetary conventions, and character classification.

locked: The condition of a cell, worksheet, or other object that restricts edits or modifications to it by users.

login name: A string that is used to identify a user or entity to an operating system, directory service, or distributed system. For example, in Windows-integrated authentication, a login name uses the form "DOMAIN\username".

Meeting Workspace site: A SharePoint site that is based on a Meeting Workspace site template and has a template ID value of "2". A Meeting Workspace site is used for planning, posting, and working together on meeting materials.

member: A user in the Members group of a site.

metadict: A dictionary that has strongly typed values.

moderation status: A content approval status that indicates whether a list item was approved by a moderator.

navigation node: An element in the navigational structure of a site. The element is a link or a series of links to a specific page in the site.

navigation node element identifier: An integer that identifies a navigation node. This value is unique for every navigation node in the navigational structure of a SharePoint site.

navigation structure: A hierarchical organization of links between related content on a site.

owner: A **security principal** who has the requisite permission to manage a security group.

page: A file that consists of HTML and can include references to graphics, scripts, or dynamic content such as Web Parts.

page type: An integer that specifies the type of a page.

parent site: The site that is above the current site in the hierarchy of the site collection.

permission: A rule that is associated with an object and that regulates which users can gain access to the object and in what manner. See also rights.

permission level: A set of permissions that can be granted to principals or SharePoint groups on an entity such as a site, list, folder, item, or document.

personal view: A view of a list that is created by a user for personal use. The view is unavailable to other users.

principal: (1) An authenticated entity that initiates a message or channel in a distributed system.

(2) An identifier of such an entity.

property bag: A container that stores data but is not defined in the schema for a SharePoint list. Instead of interpreting data in a property bag, the server only passes the data in response to requests. See also **metadict**.

published: A condition of portions of a workbook that are marked as being available to the user when that workbook is processed by a protocol server.

published version: The version of a list item that is approved and can be seen by all users. The user interface (UI) version number for a published version is incremented to the next positive major version number and the minor version is "0" (zero). See also major version and minor version.

read-only mode: An attribute that indicates that an object cannot be changed or deleted. The object can only be accessed or displayed.

result set: A list of records that results from running a stored procedure or query, or applying a filter. The structure and content of the data in a result set varies according to the implementation.

return code: A code that is used to report the outcome of a procedure or to influence subsequent events when a routine or process terminates (returns) and passes control of the system to another routine. For example, a return code can indicate whether an operation was successful.

role: A symbolic name that defines a class of users for a set of components. A role defines which users can call interfaces on a component.

role assignment: An association between a principal or a site group and a role definition.

role definition: A named set of permissions for a SharePoint site. See also **permission level**.

root folder: The folder at the top of a hierarchy of folders in a list.

security group: A named group of principals on a SharePoint site.

security principal: An identity that can be used to regulate access to resources. A security principal can be a user, a computer, or a group that represents a set of users.

security provider: A Component Object Model (COM) object that provides methods that return custom information about the security of a site.

security role: A defined set of access privileges. The security role that is assigned to a user determines the tasks that a user can perform and which parts of the user interface a user can view.

security scope: A tree structure of objects in which every object has the same security settings as the root.

Security Support Provider Interface (SSPI): A Windows-specific API implementation that provides the means for connected applications to call one of several security providers to establish authenticated connections and to exchange data securely over those connections. This

is the Windows equivalent of Generic Security Services (GSS)-API, and the two families of APIs are on-the-wire compatible.

server-relative URL: A relative URL that does not specify a scheme or host, and assumes a base URI of the root of the host, as described in [RFC3986].

setup path: The location where supporting files for a product or technology are installed.

short-term lock: A type of check-out process in Windows SharePoint Services. Short-term checkouts are implicit and are done when a file is opened for editing. A lock is applied to the file while it is being edited in the client application so that other users cannot modify it. After the client application is closed, the lock is released.

site: A group of related pages and data within a SharePoint site collection. The structure and content of a site is based on a site definition. Also referred to as SharePoint site and web site.

site collection: A set of websites that are in the same **content database**, have the same owner, and share administration settings. A site collection can be identified by a **GUID** or the **URL** of the top-level site for the site collection. Each site collection contains a top-level site, can contain one or more subsites, and can have a shared navigational structure.

site collection administrator: A user who has administrative permissions for a site collection.

site collection identifier: A GUID that identifies a site collection. In stored procedures, the identifier is typically "@SiteId" or "@WebSiteId". In databases, the identifier is typically "SiteId/tp_SiteId".

site definition: A family of site definition configurations. Each site definition specifies a name and contains a list of associated site definition configurations.

site template: An XML-based definition of site settings, including formatting, lists, views, and elements such as text, graphics, page layout, and styles. Site templates are stored in .stp files in the content database.

SQL authentication: One of two mechanisms for validating attempts to connect to instances of SQL Server. In SQL authentication, users specify a SQL Server login name and password when they connect. The SQL Server instance ensures that the login name and password combination are valid before permitting the connection to succeed.

stored procedure: A precompiled collection of SQL statements and, optionally, control-of-flow statements that are stored under a name and processed as a unit. They are stored in a SQL database and can be run with one call from an application. Stored procedures return an integer return code and can additionally return one or more result sets. Also referred to as sproc.

store-relative form: See **store-relative URL**.

store-relative URL: A URL that consists only of a path segment and does not include the leading and trailing slash.

stream: An element of a compound file, as described in [MS-CFB]. A stream contains a sequence of bytes that can be read from or written to by an application, and they can exist only in storages.

Structured Query Language (SQL): A database query and programming language that is widely used for accessing, querying, updating, and managing data in relational database systems.

subsite: A complete website that is stored in a named subdirectory of another website. The parent website can be the top-level site of a site collection or another subsite. Also referred to as subweb.

SystemID: A binary identifier that is used to uniquely identify a **security principal**. For Windows integrated authentication, it is a security identifier (SID). For an ASP.NET Forms Authentication provider, it is the binary representation that is derived from a combination of the provider name and the user login name.

thicket: A means of storing a complex HTML document with its related files. It consists of a thicket main file and a hidden thicket folder that contains a thicket manifest and a set of thicket supporting files that, together, store the referenced content of the document.

thicket folder: A hidden folder that contains a thicket manifest and a set of thicket supporting files that, together, store the referenced content of a complex HTML document.

thicket main file: The core file of a complex HTML document. It references contained elements such as graphics, pictures, or other media that are stored as thicket supporting files in a thicket folder. The thicket main file is the target that is used by a protocol client to access the content of the document.

Transact-Structured Query Language (T-SQL): A language that contains the commands that are used to manage instances of Microsoft SQL Server, create and manage all objects in an instance of SQL Server, and to insert, retrieve, modify, and delete all data in SQL Server tables. Transact-SQL is an extension of the language that is defined in the SQL standards that are published by the International Standards Organization (ISO) and the American National Standards Institute (ANSI).

type information: A collection of information that describes the characteristics and capabilities of an object, including the properties, events, and methods for the object.

Uniform Resource Locator (URL): A string of characters in a standardized format that identifies a document or resource on the World Wide Web. The format is as specified in [RFC1738].

unique identifier (UID): A pair consisting of a **GUID** and a version sequence number to identify each resource uniquely. The UID is used to track the object for its entire lifetime through any number of times that the object is modified or renamed.

user identifier: An integer that uniquely identifies a **security principal** as distinct from all other **security principals** and site groups within the same site collection.

user interface (UI) version: A single 4-byte integer that stores the version number that appears as a document version number in the user interface. The lower 9 bits correspond to the minor version number of the displayed version. The remaining 23 bits correspond to the major version number of the displayed version. See also **displayed version**.

user name: A unique name that identifies a specific user account. The user name of an account is unique among the other group names and user names within its own domain or workgroup.

UTF-8: A byte-oriented standard for encoding Unicode characters, defined in the Unicode standard. Unless specified otherwise, this term refers to the UTF-8 encoding form specified in [UNICODE5.0.0/2007] section 3.9.

version: See **displayed version**, historical version, major version, and minor version.

view: See form view (Microsoft InfoPath), **list view** (SharePoint Products and Technologies), or View (Microsoft Business Connectivity Services).

view identifier: A GUID that is used to uniquely identify a view.

virus scanner: Software that is used to search for and remove computer viruses, worms, and Trojan horses.

web bot: See **bot**.

web discussion: A component and add-in that enables users to enter comments about documents and pages without modifying the actual content of those documents or pages.

Web Distributed Authoring and Versioning Protocol (WebDAV): The Web Distributed Authoring and Versioning Protocol, as described in [RFC2518] or [RFC4918].

Web Part: A reusable component that contains or generates web-based content such as **XML**, **HTML**, and scripting code. It has a standard property schema and displays that content in a cohesive unit on a webpage. See also Web Parts Page.

Web Part Page: An ASP.NET webpage that includes Web Part controls that enable users to customize the page, such as specifying which information to display. Referred to as Web Parts Page in SharePoint Foundation 2010.

Web Part zone: A structured HTML section of a Web Parts Page that contains zero or more Web Parts and can be configured to control the organization and format of those Web Parts.

Web Part zone identifier: A string that identifies a Web Part zone on a Web Parts Page.

Welcome page: A page, such as default.aspx, that can be specified as the default redirect target when users browse to a URL without specifying a **leaf name**.

Windows code page: A table that relates the character codes (code point values) that are used by an application to keys on a keyboard or to characters on a display. This provides support for **character sets** and keyboard layouts for different countries or regions. Also referred to as character set or charset.

Windows collation name: A string identifier that follows the format of the **Transact-Structured Query Language (T-SQL)** COLLATE clause.

XML: The Extensible Markup Language, as described in [XML1.0].

XML document: A document object that is well formed, as described in [XML10/5], and might be valid. An XML document has a logical structure that is composed of declarations, elements, comments, character references, and processing instructions. It also has a physical structure that is composed of entities, starting with the root, or document, entity.

XML Path Language (XPath): A language used to create expressions that can address parts of an XML document, manipulate strings, numbers, and Booleans, and can match a set of nodes in the document, as specified in [XPATH]. XPath models an XML document as a tree of nodes of different types, including element, attribute, and text. XPath expressions can identify the nodes in an XML document based on their type, name, and values, as well as the relationship of a node to other nodes in the document.

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

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

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.

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

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

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

[MSDN-TSQL-Ref] Microsoft Corporation, "Transact-SQL Reference", SQL Server 2005 Books Online (November 2008), http://msdn.microsoft.com/en-us/library/ms189826(SQL.90).aspx

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

[TSQL-Ref] Microsoft Corporation, "Transact-SQL Reference (Transact-SQL)", http://msdn.microsoft.com/en-us/library/dd884419.aspx

[XMLSCHEMA1] Thompson, H., Beech, D., Maloney, M., and Mendelsohn, N., Eds., "XML Schema Part 1: Structures", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[XMLSCHEMA2] Biron, P.V., Ed. and Malhotra, A., Ed., "XML Schema Part 2: Datatypes", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

1.2.2 Informative References

[CSS3UI] Celik, T., Ed., "CSS Basic User Interface Module Level 3 (CSS3 UI)", W3C Working Draft 17, January 2012, http://www.w3.org/TR/2012/WD-css3-ui-20120117/

[MC-FPSEWM] Microsoft Corporation, "FrontPage Server Extensions: Website Management Protocol".

[MS-WSSO] Microsoft Corporation, "Windows SharePoint Services Overview".

[RFC2518] Goland, Y., Whitehead, E., Faizi, A., et al., "HTTP Extensions for Distributed Authoring - WebDAV", RFC 2518, February 1999, http://www.ietf.org/rfc/rfc2518.txt

[RFC2616] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.rfc-editor.org/rfc/rfc2616.txt

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 file access and administration of users and **groups** within Windows SharePoint Services. This client-to-server protocol uses the Tabular Data Stream (TDS) Protocol (as described in [MS-TDS]) as its transport between the front-end Web server, acting as a client, and the back-end database server, acting as a server. **Transact-Structured Query Language (T-SQL)** (as described in [TSQL-Ref]) is used to define the queries and returned data that is transported over TDS.

End-user clients use remote file access protocols to communicate with front-end Web servers, specifically using the FrontPage Server Extensions Remote Protocol (as described in [MS-FPSE]), **Hypertext Transfer Protocol (HTTP)** (as described in [RFC2616]), and **Web Distributed Authoring and Versioning Protocol (WebDAV)** (as described in [RFC2518]).

Further information about the interoperation of the clients with the front-end Web server, and the front-end Web server with the back-end database server, can be found in the Windows SharePoint Services Overview ([MS-WSSO]).

1.3.1 File Operations

This protocol provides methods for retrieving and manipulating files' properties, along with support for retrieving and manipulating files' security information. When client requests for files or file information are sent to the front-end Web server, the front-end Web server sends a series of **stored procedure** calls to the back-end database server for the requested information. The stored procedures return data that in turn can be used for further calls to other stored procedures. The front-end Web server turns the values in the stored procedures' **return codes** and **result sets** into the data and metadata for the files requested by the client, and sends it back to the client using the same protocol used by the initial request.

1.3.2 User and Group Operations

This protocol provides methods for retrieving and manipulating information about individual users and **groups**, along with support for retrieving information from a **directory service (DS)** about users. When the Object Model on the front-end Web server operates on requests to query or update users or groups, the front-end Web server confirms whether the data is already populated in the local objects that represent the specific user or groups. If it does not exist, the front-end Web server sends a series of stored procedure calls to the back-end database server for the requested information. The stored procedures return data, which in turn can be used for further calls to other stored procedures. The front-end Web server turns the values in the stored procedures' return codes and result sets into objects that contain the data and metadata for the requested users or groups, and uses the objects according to implementation-specific procedures.

1.4 Relationship to Other Protocols

This protocol relies on TDS (see [MS-TDS]) as its transport protocol to call stored procedures to inspect and manipulate **document** properties via result sets and return codes. Database queries or calls to stored procedures, and the returned result sets, are written in the T-SQL language.

Requests to a WSS front-end Web server via FrontPage Server Extensions (as described in [MS-FPSE]) and WebDAV (as described in [RFC2518]) rely on this protocol, via the front-end Web server, to retrieve and manipulate file and security information persistently stored on the back-end database server and to service requests for files and their properties from their clients.

1.5 Prerequisites/Preconditions

Unless otherwise specified, the stored procedures and any related tables are present in the **content database** that is being queried on the back-end database server. The tables in the content database have to contain valid data in a consistent state in order to be queried successfully by the stored procedures.

For operations defined in this document, any file access, addition, or modification has to be to a valid location, such as a **site**, **list (1)**, **document library**, **folder**, or document, as defined by the data within the tables and the front-end Web server, in order for the request to be successfully processed. The user making the request to the front-end Web server has to have adequate **permission** to access the content of the specified valid location in order for the request to be successfully processed.

1.6 Applicability Statement

This protocol is only applicable to front-end Web servers when communicating with the back-end database server for file, user, and **group** administration operations.

1.7 Versioning and Capability Negotiation

The client and server in this protocol perform explicit **version** verifications. This protocol supports the **Security Support Provider Interface (SSPI)** and **SQL authentication** with the back-end database server. These authentication methods are described in [MS-TDS].

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

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

2.2 Common Data Types

The following are common data types used in conjunction with this protocol. The low-**level** data type and size are specified using commonly-known data type descriptions. It is possible that the variable can be stored in multiple T-SQL data types, depending on the actual implementation of each stored procedure, result set, or database table. If the data type is only implemented in this protocol using a T-SQL data type, then the data type is specifically listed as a T-SQL data type.

2.2.1 Simple Data Types and Enumerations

The following are common simple data types used in conjunction with this protocol. When the data type is defined as being a **GUID**, it is possible for it to be represented in T-SQL as a uniqueidentifier or as a string. The specific T-SQL language data type used to hold the GUID is determined by the actual definition of the stored procedure, result set, or database table.

2.2.1.1 Calendar Type

A **Calendar Type** is a 2-byte integer value that specifies the type of calendar to use in a particular context. The only valid values of the **Calendar Type** are specified as follows.

Value	Description
0	None
1	Gregorian (localized)
6	Hijri (Arabic Lunar)
7	Thai (Buddhist)
8	Hebrew (Lunar)
16	Saka Era

2.2.1.2 CharSet Enumeration

A **CharSet Enumeration** is an optional **character set** associated with the document. The only valid values of the **CharSet Enumeration** are specified as follows.

Value	Meaning
0	US-ASCII
1	Latin 1
2	Windows (US/Western Europe)

Value	Meaning
3	Euro support
4	Windows Latin 2 (Central Europe)
5	ISO Latin 2 (Central Europe)
6	Latin 4 or Baltic
7	Cyrillic (Slavic)
8	Russian KOI8R
9	Arabic
10	Greek
11	Hebrew
12	Latin 5 (Turkish)
13	Vietnamese
14	Japanese (SHIFT-JIS)
15	Japanese (JIS)
16	Japanese (EUC)
17	Korean (Wansung)
18	Korean (EUC)
19	Traditional Chinese
20	Simplified Chinese
21	Simplified Chinese (GB18030)
22	Thai
23	Unicode 2.0 (UCS-2)
24	Unicode 2.0 (**UTF-8**)
25	UnicodeFFFE 2.0 (UCS-8)
26	dynamically set charset when 1st key typed
27	Blank

2.2.1.3 Collation Order Enumeration

Collation Order Enumeration is a 2-byte integer value indicating **collation order** mapped to a **Windows collation name**, as specified in [Iseminger]. The only valid values of the **Collation Order Enumeration** are specified as follows.

Value	Meaning
0	Albanian

Value	Meaning
1	Arabic
2	Chinese_PRC
3	Chinese_PRC_Stroke
4	Chinese_Taiwan_Bopomofo
5	Chinese_Taiwan_Stroke
6	Croatian
7	Cyrillic_General
8	Czech
9	Danish_Norwegian
10	Estonian
11	Finnish_Swedish
12	French
13	Georgian_Modern_Sort
14	German_PhoneBook
15	Greek
16	Hebrew
17	Hindi
18	Hungarian
19	Hungarian_Technical
20	Icelandic
21	Japanese
22	Japanese_Unicode
23	Korean_Wansung
24	Korean_Wansung_Unicode
25	Latin1_General
26	Latvian
27	Lithuanian
28	Lithuanian_Classic
29	Traditional_Spanish
30	Modern_Spanish
31	Polish
32	Romanian

Value	Meaning
33	Slovak
34	Slovenian
35	Thai
36	Turkish
37	Ukrainian
38	Vietnamese

2.2.1.4 Document Identifier

A **Document Identifier** is a GUID used to uniquely identify a document within a **site collection**. Specialized varieties of **document identifier** include **Site Identifiers** (section 2.2.1.19) and **List Identifiers** (section 2.2.1.10).

2.2.1.5 Global Identifier

A **Global Identifier** is a GUID used to uniquely identify the global settings.

2.2.1.6 LinkDynamic Type

A **LinkDynamic Type** is a 1-byte value represented as a single, uppercase ASCII character that tracks various special **link** types. A **LinkDynamic Type** MUST have only one value at a time. A NULL value for **LinkDynamic Type** is used for a **backward link**. The only valid non-NULL values of the **LinkDynamic Type** are specified as follows.

Value	Description
D	The **URL** is "dynamic", which is a link to <Site URL>/_vti_bin/shtml.dll/DirName/LeafName. Such links are used to invoke the FrontPage SmartHTML interpreter on a file.
G	A non-absolute link from a ghosted document that does not fall into any other category.
H	The URL is a history link (that is, it contains a path segment with the string "_vti_history").
L	The URL is to a layouts **page** (that is, it contains a path segment with the string "_layouts").
S	The URL is "static", which is the default and requires no special handling.

2.2.1.7 LinkSecurity Type

A **LinkSecurity Type** is a 1-byte value represented as a single, uppercase ASCII character specifying the URI scheme for a link, such as HTTP or HTTPS. A **LinkSecurity Type** MUST have only one value at a time. A NULL value for **LinkSecurity Type** is used for a backward link. The only valid non-NULL values of the **LinkSecurity Type** are specified as follows.

Value	Description
H	The URL begins with "http://" (a nonsecure link using the http: scheme).
S	The URL begins with "https://" (an SSL link using the https: or snews: scheme).
T	The URL begins with "shttp://" (an S-HTTP link using Terisa's shttp: scheme).
U	The URL is of another unknown scheme.

2.2.1.8 LinkType Type

A **LinkType Type** is a 1-byte value represented as a single, uppercase ASCII character; it specifies **type information** about a link. A **LinkType Type** MUST have only one value at a time. -1 value for **LinkType Type** is used for a backward link. The only valid non-NULL values of the **LinkType Type** are specified as follows.

Value	Description
A	The link is from the ACTION attribute of an **HTML** form tag.
B	The link is from the attribute markup of a **bot**.
C	The link is from an autogenerated table of contents. Agents can ignore the link type when determining unreferenced files within a **site**.
D	The link references programmatic content, as in the HTML OBJECT or APPLET tags.
E	The link is from a cascading style sheet (CSS).
F	The link is from the SRC attribute of an HTML FRAME tag.
G	The link is to a **dynamic web template** for the containing document.
H	The link is from an HTML HREF attribute. This can also be used as a default link type value if a more precise type does not apply.
I	The link is to a document that the containing document includes via an include bot.
K	Identical to "H", except that the link contains an HTML bookmark specifier.
L	The link is a target in an HTML image map generated from an image map bot.
M	The link is to an image used in an HTML image map generated from an image map bot.
P	The link is part of the markup of a **Web Part** within the source of the containing document.
Q	The link references a CSS document that provides style information for the containing document.
S	The link is from an HTML SRC attribute.
T	The link is to the index file used by a text search bot on this page.
X	The link is from an **XML** island within an HTML document.
Y	The link references an HTML document whose HTML BODY tag attributes are used as a template for the attributes of the containing document's BODY tag.
Z	The link is part of the markup of a Web Part that exists in a **Web Part zone identifier** in the containing document and is consequently not stored within the source of the containing document.

2.2.1.9 List Base Type

A **List Base Type** is a 32-bit integer enumeration of possible **base types** for lists. All lists are created with one of these base types, which define implementation-specific common values for list properties. The only valid values of the **List Base Type** are specified as follows.

Value	Meaning
0	Generic **list (1)**
1	Document library
3	Discussion board list (1)
4	Survey list (1)
5	Issues list (1)

2.2.1.10 List Identifier

A **List Identifier** is a variety of **Document Identifier** (section 2.2.1.4), a GUID used to uniquely identify a **list (1)** within a site collection.

2.2.1.11 List Item Identifier

A **List Item Identifier** is a 4-byte integer value used to uniquely identify a **list item** within any **list (1)** in a particular site collection.

2.2.1.12 List Server Template

A **List Server Template** is a 32-bit integer enumeration of the possible values for the **list server template** defining the base structure of a **list (1)**. The only valid values of the **List Server Template** are specified as follows.

Value	Meaning
-1	Invalid Template
100	Generic List Template
101	Document Library Template
102	Survey Template
103	Links Template
104	Announcements Template
105	Contacts Template
106	Events Template
107	Tasks Template
108	Discussion Template

Value	Meaning
109	Image Library Template
110	Data Sources Template
111	Web Template Catalog Template
112	User Info Catalog Template
113	Web Part Gallery Template
114	List Template Catalog Template
115	XML Form Template
120	Custom Grid Template
200	Meetings Template
201	Agenda Template
202	Meeting User Template
204	Decision (Meeting) Template
207	Meeting Objective Template
210	Textbox Template
211	Things To Bring (Meeting) Template
212	Homepage Library Template
1100	Issue Tracking Template

2.2.1.13 Moderation Status

Moderation Status is a 4-byte integer indicating the **moderation status** of a **list item**. Configurations can require moderation approval to publish a list item or allow automatic approval. The only valid values of the **Moderation Status** are specified as follows.

Value	Description
0	The list item is approved.
1	The list item is denied.
2	The list item is pending approval.

2.2.1.14 Page Type

A **Page Type** is a 1-byte signed integer that is used to represent the possible **page types**. The only valid values of the **Page Type** are specified as follows.

Value	Meaning
-1	Invalid.
0	Default **view** of the corresponding **list (1)**.
1	A view of the corresponding list (1), but not the default view.
2	This value is only used internally within implementation-specific code and is never stored in a database.
3	This value is only used internally within implementation-specific code and is never stored in a database.
4	A display form of a list (1), suitable for displaying a single list item in **read-only mode**.
5	This value is only used internally within implementation-specific code and is never stored in a database.
6	An edit form for a list (1), suitable for presenting UI to update the properties of a list item.
7	Used to represent edit forms of a list suitable for displaying in HTML file dialogs to a client application.
8	A new form for a list (1), suitable for presenting UI to create a new list item.
9	Used to represent new forms of a list (1) suitable for displaying in HTML file dialogs to a client application. This value is from a previous implementation and is no longer valid.
10	This value is only used internally within implementation-specific code and is never stored in a database.

2.2.1.15 Role Identifier

A **Role Identifier** is a 4-byte integer value used to uniquely identify a **role definition** within a site collection. The only valid values of the **Role Identifier** are specified as follows.

Value	Definition
1073741825	Guest
1073741826	Reader
1073741827	Contributor
1073741828	Web Designer
1073741829	Administrator

2.2.1.16 Server Identifier

A **Server Identifier** is a GUID used to uniquely identify a server.

2.2.1.17 Site Collection Identifier

A **Site Collection Identifier** is a GUID used to uniquely identify a site collection within a content database.

2.2.1.18 Site Group Identifier

A **Site Group Identifier** is a 4-byte integer value used to uniquely identify a site group within a site collection. **Site Group Identifiers** are assigned from the same numbering space as **user identifiers** and cannot overlap. Values of -1 and 0 are reserved to indicate invalid or unknown user or site group identifiers.

2.2.1.19 Site Identifier

A **Site Identifier** is a variety of **Document Identifier** (section 2.2.1.4), a GUID used to uniquely identify a **site** within a site collection.

2.2.1.20 SystemID

A **SystemID** is a binary value of arbitrary but limited length that uniquely identifies a **principal (1)**, stored on the back-end database server as a **tSystemID** (section 2.2.1.23).

2.2.1.21 Time Zone Identifier

A **Time Zone Identifier** is a 2-byte integer value identifying a time zone. The values of the **Time Zone Identifier** are specified as follows.

Value	Meaning
1	(GMT) Greenwich Mean Time: Dublin, Edinburgh, Lisbon, London
2	(GMT+01:00) Sarajevo, Skopje, Warsaw, Zagreb
3	(GMT+01:00) Brussels, Copenhagen, Madrid, Paris
4	(GMT+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna
5	(GMT+02:00) Bucharest
6	(GMT+01:00) Belgrade, Bratislava, Budapest, Ljubljana, Prague
7	(GMT+02:00) Minsk, Athens, Istanbul
8	(GMT-03:00) Brasilia
9	(GMT-04:00) Atlantic Time (Canada)
10	(GMT-05:00) Eastern Time (U.S. and Canada)
11	(GMT-06:00) Central Time (U.S. and Canada)
12	(GMT-07:00) Mountain Time (U.S. and Canada)
13	(GMT-08:00) Pacific Time (U.S. and Canada), Tijuana
14	(GMT-09:00) Alaska
15	(GMT-10:00) Hawaii
16	(GMT-11:00) Midway Island, Samoa
17	(GMT+12:00) Auckland, Wellington
18	(GMT+10:00) Brisbane
19	(GMT+09:30) Adelaide

Value	Meaning
20	(GMT+09:00) Osaka, Sapporo, Tokyo
21	(GMT+08:00) Kuala Lumpur, Singapore
22	(GMT+07:00) Bangkok, Hanoi, Jakarta
23	(GMT+05:30) Chennai, Kolkata, Mumbai, New Delhi
24	(GMT+04:00) Abu Dhabi, Muscat
25	(GMT+03:30) Tehran
26	(GMT+03:00) Baghdad
27	(GMT+02:00) Jerusalem
28	(GMT-03:30) Newfoundland
29	(GMT-01:00) Azores
30	(GMT-02:00) Mid-Atlantic
31	(GMT) Casablanca, Monrovia
32	(GMT-03:00) Buenos Aires, Georgetown
33	(GMT-04:00) Caracas, La Paz
34	(GMT-05:00) Indiana (East)
35	(GMT-05:00) Bogota, Lima, Quito
36	(GMT-06:00) Saskatchewan
37	(GMT-06:00) Guadalajara, Mexico City
38	(GMT-07:00) Arizona
39	(GMT-12:00) Eniwetok, Kwajalein
40	(GMT+12:00) Fiji Is., Kamchatka, Marshall Is.
41	(GMT+11:00) Magadan, Solomon Is., New Caledonia
42	(GMT+10:00) Hobart
43	(GMT+10:00) Guam, Port Moresby
44	(GMT+09:30) Darwin
45	(GMT+08:00) Beijing, Chongqing, Hong Kong S.A.R., Urumqi
46	(GMT+06:00) Almaty, Novosibirsk
47	(GMT+05:00) Islamabad, Karachi, Tashkent
48	(GMT+04:30) Kabul
49	(GMT+02:00) Cairo
50	(GMT+02:00) Harare, Pretoria
51	(GMT+03:00) Moscow, St. Petersburg, Volgograd

Value	Meaning
53	(GMT-01:00) Cabo Verde Is.
54	(GMT+04:00) Baku, Tbilisi, Yerevan
55	(GMT-06:00) Central America
56	(GMT+03:00) Nairobi
57	(GMT+10:00) Canberra, Melbourne, Sydney
58	(GMT+05:00) Ekaterinburg
59	(GMT+02:00) Helsinki, Tallinn
60	(GMT-03:00) Greenland
61	(GMT+06:30) Yangon (Rangoon)
62	(GMT+05:45) Kathmandu
63	(GMT+08:00) Irkutsk, Ulaan Bataar
64	(GMT+07:00) Krasnoyarsk
65	(GMT-04:00) Santiago
66	(GMT+06:30) Sri Jayawardenepura
67	(GMT+13:00) Nuku'alofa
68	(GMT+10:00) Vladivostok
69	(GMT+01:00) West Central Africa
70	(GMT+09:00) Yakutsk
71	(GMT+06:00) Astana, Dhaka
72	(GMT+09:00) Seoul
73	(GMT+08:00) Perth
74	(GMT+03:00) Kuwait, Riyadh
75	(GMT+08:00) Taipei
76	(GMT+10:00) Canberra, Melbourne, Sydney
77	(GMT-07:00) Chihuahua, La Paz, Mazatlan

2.2.1.22 tPermMask

A **tPermMask** is an integer value stored on back-end database server as a T-SQL integer.

2.2.1.23 tSystemID

A **tSystemID** is a binary value of arbitrary but limited length stored on the back-end database server as a T-SQL varbinary(128).

2.2.1.24 User Identifier

A **User Identifier** is a 4-byte integer value used to uniquely identify a principal (1) within a site collection.

2.2.1.25 View Identifier

A **View Identifier** is a 4-byte integer value used to identify a **view** within a **list (1)** or document library. A **view identifier** is unique only within a particular list (1) or document library.

2.2.1.26 Virus Status

Virus Status is a 4-byte, integer enumerated type that specifies the current virus scan status of a document. The only valid values of the **Virus Status** are specified as follows.

Value	Description
0	This document is reported as clean from viruses.
1	This document had a virus reported by the **virus scanner** plug-in.
2	This document had a virus reported by the virus scanner plug-in, which the scanner determines that it can remove.
3	This document had a virus previously reported, but the virus scanner determines that it successfully removed it.
4	This document had a virus reported, and the virus scanner attempted to clean it but failed.
5	This document had a virus reported, and the scanner requested that the document be deleted.
6	This document had a timeout from the virus scanner when it was last processed.

2.2.1.27 Web Part Identifier

A **Web Part identifier** is a GUID used to uniquely identify a **Web Part** within a site collection.

2.2.2 Bit Fields and Flag Structures

2.2.2.1 Attachments Flag

The **Attachments Flag** is a 1-byte integer flag specifying whether an **item** appears to be an **attachment** or a folder related to attachments based on a document's URL. The only valid values of **Attachments Flag** are as follows.

Value	Description
0	The URL does not appear to be an attachment.
1	The URL is an attachment file.
2	The URL is an attachment subfolder.
3	The URL is an attachment **root folder**.

2.2.2.2 Doc Flags

The **Doc Flags** is a 4-byte unsigned integer bit mask that provides metadata about the document, which can have one or more flags set. The only valid values of **Doc Flags** are as follows.

Value	Description
0x00000000	None
0x00000001	This document contains dynamic content to be sent through the **CAML** interpreter, an implementation-specific dynamic content generation component. An example of this would be a category **Web bot** present in the source of the page.
0x00000002	The document is a "sub-image" of another document. This is set if this document is an automatically generated thumbnail or web image based on another item in the store.
0x00000004	The document is a type for which there was a registered parser available at the time it was saved. A parser is an implementation-specific component that can extract data and metadata from a document, which can then be used to build a list of hyperlinks and **fields** for content types.
0x00000008	The document is a type that can contain hyperlinks.
0x00000010	The document has an associated resource in the "_private" folder that is renamed in parallel when this file is renamed. An example of this is the count file for a hit counter Web bot.
0x00000020	The document is currently **checked out** to a user.
0x00000040	The document content is stored in the content database.
0x00000080	For a document that has Web Part personalization, the personal collection of Web Parts is returned by default unless otherwise specified.

2.2.2.3 Document Store Type

The **Document Store Type** is a 1-byte unsigned integer value that specifies the type of a document or the target of a link within or to a document. The only valid values of **Document Store Type** are as follows.

Value	Description
0x00	file
0x01	folder
0x02	site

2.2.2.4 List Flags

The **List Flags** is a 4-byte unsigned integer bit mask that provides metadata about the **list (1)**, which can have one or more flags set. **List Flags** identify an implementation-specific capability. The only valid values of **List Flags** are as follows.

Value	Description
0x00000001	This list (1) is an "ordered list" (for example, a Links List) and supports ordering and reordering of its items.

Value	Description
0x00000002	This list (1) is a "public list". This bit MUST be ignored.
0x00000004	This list (1) is "undeletable" (that is, it is crucial to the functioning of the containing **site** or site collection).
0x00000008	Attachments on list items are disabled. This bit MUST be set if the list (1) is a document library or survey.
0x00000010	This list (1) is a "catalog" (for example, a Web Part gallery or master page gallery).
0x00000020	This list (1) is associated with a site using the meetings workspace **site template** and contains data scoped to each instance of a recurring meeting.
0x00000040	This list (1) MUST send **alerts** when a list item is assigned to a user.
0x00000080	This list (1) has versioning enabled, and supports creating historical versions of list items when changes occur. This bit MUST be ignored for Lists with a **List Base Type** of survey.
0x00000100	This list (1) MUST be hidden from enumeration functions. This is intended for lists implementing infrastructure for an application.
0x00000200	This list (1) is configured to bring up a page to fill out a **form** to request access from the **owner** when a user is denied access while browsing its list items.
0x00000400	This list (1) has moderation enabled, requiring an approval process when content is created or modified.
0x00000800	If this list (1) is a survey, it will allow multiple responses for a given user rather than restricting users to a single response. This flag MUST be ignored for lists that do not have a **List Base Type** of survey.
0x00001000	This list (1) uses the value of each **field's** ForcedDisplay attribute when presenting data from that field. This is commonly used in anonymous surveys to display common placeholder text wherever the respondent's name would normally appear.
0x00002000	This list (1) MUST NOT be serialized as part of saving this site as a site template.
0x00004000	The **List Server Template** (section 2.2.1.12) for this list (1) can only be instantiated in the root site of a given site collection.
0x00008000	When a **List Server Template** is being created for this list (1), documents in the root of the list can also be serialized.
0x00010000	Insertion of list items via email is enabled for this list (1).
0x00020000	This is a "private" list. When a **List Server Template** based on this list (1) is created, the new list can be given an ACL so that only its owner and administrators can access the list.
0x00800000	This list (1) has had its schema customized from the version that exists in the on-disk schema file that was used to create it.
0x01000000	This enables to explicitly map a document property to a specific column.
0xFFFFFFFF	Invalid.

2.2.2.5 Put Flags Type

The **Put Flags Type** is a 4-byte integer bit mask containing option flags for adding or updating a document. Zero or more of the following bit flags can be set in a **Put Flags Type**. The only valid values of **Put Flags Type** are as follows.

Value	Description
0x00000000	Illegal value.
0x00000002	Unconditionally update the document.
0x00000010	Create a directory to hold the document, if necessary.
0x00000800	If this **list (1)** is a survey, it will allow multiple responses for a given user rather than restricting users to a single response. This flag MUST be ignored for lists that do not have a **List Base Type** of survey.
0x00002000	This list (1) MUST NOT be serialized as part of saving this **site** as a site template.

2.2.2.6 Rename Flags

The **Rename Flags** is a 4-byte integer bit mask that specifies option flags for renaming a document. This bit mask can have zero or more flags set. The only valid values of **Rename Flags** are as follows.

Value	Description
0x00000000	Default behavior: Rename all dependent items.
0x00000001	Do not update all related documents.
0x00000004	Server SHOULD find backward links in order to rename them and update the original document.

2.2.2.7 Site Collection Flags

The **Site Collection Flags** is a 4-byte, unsigned integer bit mask that specifies properties that are global to a site collection. This bit mask can have zero or more flags set. The only valid values of **Site Collection Flags** are as follows.

Value	Meaning
0x00000001	The site collection has been Write-locked, and user write operations will be blocked.
0x00000002	The site collection has been Fully-locked, and user read and write operation will be blocked.
0x00000004	The site collection has been Foundation-locked.
0x00000008	The site collection has been Disk-locked.
0x00000010	The site collection has been Bandwidth–locked.
0x00000020	The site collection has been Non Payment–locked.
0x00000040	The site collection has been Violation-locked.
0x00000080	The site collection has sent a notification indicating that the disk is locked.

Value	Meaning
0x00000100	The site collection has sent a notification that the bandwidth is locked.
0x00000200	The site collection has sent a notification that the user is locked.
0x00000400	The site collection has sent a notification indicating that the disk usage is near limit.
0x00000800	The site collection has sent a notification that the bandwidth usage is near to full.
0x00001000	The number of users in the site collection is large.

2.2.2.8 Site Property Flags

The **Site Property Flags** is a 4-byte, unsigned integer bit mask that tracks property flags applied to a **site**. The site can have one or more **Site Property Flags** set. These flags reference implementation-specific capabilities of WSS. The only valid values of **Site Property Flags** are as follows.

Value	Meaning
0x00000001	This site allows display of implementation-specific user presence information in the front-end Web server.
0x00000002	This site allows display of implementation-specific enhanced user presence information in the front-end Web server.
0x00000004	HTML views for file dialogs MUST NOT be displayed for this site.
0x00002000	Update the time zone for this site.

2.2.2.9 View Flags

The **View Flags** is a 4-byte, integer bit mask that corresponds to properties of a view. This bit mask can have zero or more flags set. The only valid values of **View Flags** are as follows.

Value	Meaning
0x00000001	Normal HTML-based view.
0x00000002	View has been modified by a client application such that it cannot be compatible with the web interface for view modification. Implementations MUST restrict modifying any properties they do not understand.
0x00000004	Unused.
0x00000008	View MUST NOT be displayed in enumerations of the views of this List (that is, in a view selector front-end Web server element).
0x00000010	Unused.
0x00000020	View is read-only, and implementations MUST NOT allow any modifications to its properties.
0x00000040	If the query for this view returns no rows, implementations of the front-end Web server MUST return an HTTP 410 error when displaying this view as part of an HTTP request, instead of displaying a normal empty view body.

Value	Meaning
0x00000080	Presents data in a nontabular fashion. Implementations can format results in a manner compatible with free-form presentation.
0x00000100	View suitable for displaying in an HTML-based file navigation dialog to client applications.
0x00000200	View suitable to display file dialog template to client applications.
0x00000400	View has functionality for aggregating data across multiple **XML documents** within an XML form library.
0x00000800	View presents a datasheet view to a rich client application (Grid view).
0x00001000	Displays all items in the **list (1)** recursively from the specified folder, instead of only displaying the immediate child objects of the current folder.
0x00002000	Requires that view's data be expanded based on a calendar recurrence.
0x00004000	View is the system-created view of a user's items awaiting moderation in a moderated list (1).
0x00008000	System-created moderator's view of a moderated list (1) that displays list items pending approval.
0x00010000	Threaded view for legacy **discussion boards** (lists with base type 3). Implementations MUST display results in a threaded fashion, and paging of results MUST be done in terms of threads instead of by individual list items.
0x00020000	Displays HTML-based graphical charts of list item data.
0x00040000	A **personal view**, which MUST only be displayed to the user who created the view.
0x00080000	Displays data on a calendar based on date and time properties of the list items.
0x00100000	Default form of the specified type for the corresponding list (1).
0x00200000	Does not display any list items that are folders (files only).
0x00400000	Displays list items based on the item order of the list (1). If this list is not an ordered list, this value MUST be zero.
0x80000000	Unused flag value. MUST be ignored by client applications.

2.2.2.10 WSS Rights Mask

The **WSS Rights Mask** is a 4-byte, unsigned integer that specifies the rights that can be assigned to a user or site group. The only valid values of **WSS Rights Mask** are as follows:

The values of the permission mask bits are specified as follows.

Symbolic name	Value	Description
EmptyMask	0x00000000	Grant no permissions.
FullMask	0xFFFFFFFF	Grant all permissions.

The **list (1)** and document permissions (0x0000XXXX) are specified as follows.

Symbolic name	Value	Description
ViewListItems	0x00000001	Allow viewing of list items in lists, documents in document libraries, and

Symbolic name	Value	Description
		web discussion comments.
AddListItems	0x00000002	Allow addition of list items to lists, documents to document libraries, and web discussion comments.
EditListItems	0x00000004	Allow editing of list items in lists, documents in document libraries, web discussion comments, and to customize **Web Part Page** in document libraries.
DeleteListItems	0x00000008	Allow deletion of list items from lists, documents from document libraries, and web discussion comments.
CancelCheckout	0x00000100	Allow discard or check-in of a document that is checked out to another user.
ManagePersonalViews	0x00000200	Allow creation, change, and deletion of personal views of lists.
Manage List Permissions	0x00000400	Allow creation and modification of permissions on lists created.
ManageLists	0x00000800	Allow creation and deletion of lists, addition or removal of **fields** to the schema of a list (1), and addition or removal of **personal views** of a list.

The web level permissions (0xXXXX0000) are specified as follows.

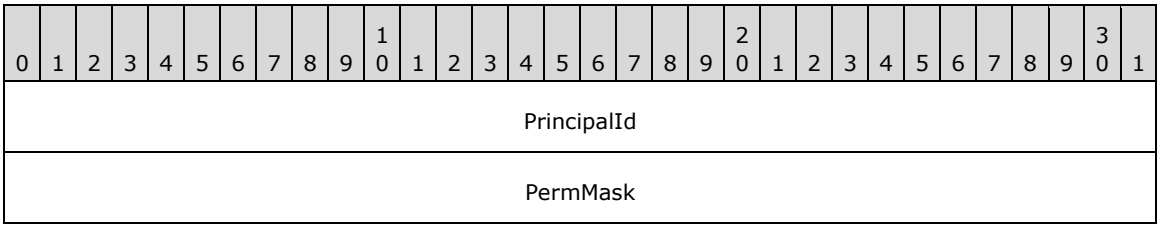
Symbolic name	Value	Description
Open	0x00010000	Allow access to the items contained within a **site**, list (1), or folder.
ViewPages	0x00020000	Allow viewing of pages in a site.
AddAndCustomizePages	0x00040000	Allow addition, modification, or deletion of HTML pages or Web Part Pages, and editing of the site using an editor compatible with WSS.
ApplyThemeAndBorder	0x00080000	Allow application of a theme or borders to the entire site.
ApplyStyleSheets	0x00100000	Allow application of a style sheet (.css file) to the site.
ViewUsageData	0x00200000	Allow viewing of reports on site usage.
CreateSSCSite	0x00400000	Allow creation of a site using Self-Service Site Creation, an implementation-specific capability of WSS.
ManageSubwebs	0x00800000	Allow creation of a **subsite** within the site or site collection.
CreatePersonalGroups	0x01000000	Allow creation of a group of users that can be used anywhere within the site collection.
ManageRoles	0x02000000	Allow creation and modification of **permission levels** on the site and assigning permissions to users and site groups.
BrowseDirectories	0x04000000	Allow enumeration of documents and folders in a site using FrontPage Server Extensions Remote Protocol (see [MS-FPSE]) and WebDAV interfaces.
BrowseUserInfo	0x08000000	Allow viewing the information about all users of the site.
AddDelPrivateWebParts	0x10000000	Allow addition or removal of personal Web Parts on a Web Part Page.
UpdatePersonalWebParts	0x20000000	Allow updating of Web Parts to display personalized information.

Symbolic name	Value	Description
ManageWeb	0x40000000	Allow all administration tasks for the site as well as manage content.

2.2.3 Binary Structures

2.2.3.1 WSS ACE

WSS ACE is an **ACE structure** specifying the individual access rights of a **principal (1)**.

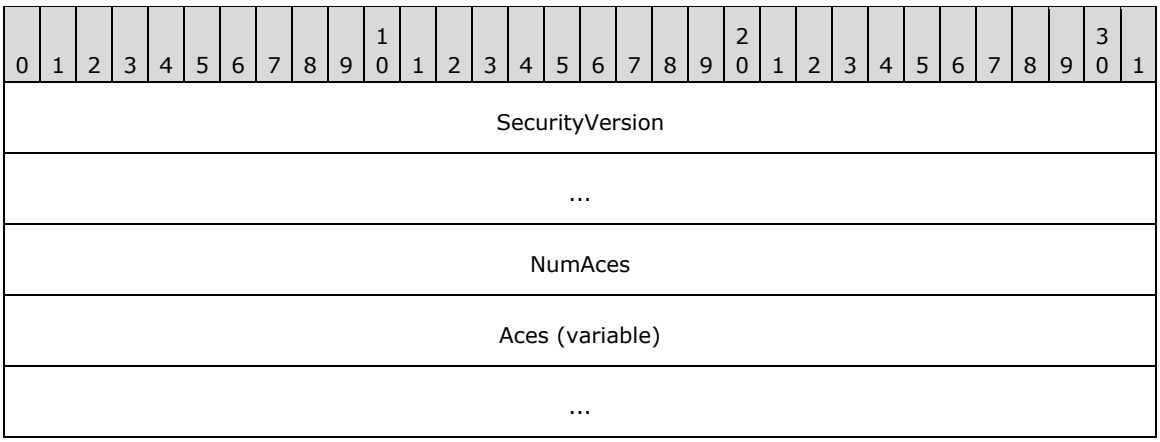


PrincipalId (4 bytes): A 4-byte integer specifying the **user identifier** of the principal (1).

PermMask (4 bytes): A 4-byte integer containing the **list (1)** of rights that are granted to the principal (1).

2.2.3.2 WSS ACL Format

The **WSS ACL Format** structure contains an array of WSS ACE (section 2.2.3.1).



SecurityVersion (8 bytes): An 8-byte integer specifying the site collection's security version value that was used to compute the ACL.

NumAces (4 bytes): A 4-byte integer specifying the count of ACEs within ACL.

Aces (variable): An array of ACEs for each **principal (1)** in this ACL.

2.2.4 Result Sets

The following result sets are used by this protocol.

2.2.4.1 Account Status Result Set

The **Account Status Result Set** returns account information for the non-deleted users matching the specified **login name** or email address. The **Account Status Result Set** is defined using T-SQL syntax, as follows.

```
tp_Login     nvarchar(255),
tp_Email     nvarchar(255),
tp_SystemId  varbinary(128);
```

tp_Login: The login name for the **principal (1)**.

tp_Email: The **email address** for the principal (1).

tp_SystemId: The **SystemID** (section 2.2.1.20) for the principal (1).

2.2.4.2 ACL and Permission Result Set

The **ACL and Permission Result Set** contains information about the permissions associated with a scope in effect for a **site**. **The ACL and Permission Result Set** is defined using T-SQL syntax, as follows.

```
AnonymousPermMask   int,
Acl                 image;
```

AnonymousPermMask: A **WSS Rights Mask** (section 2.2.2.10) indicating the rights granted to an **anonymous user** or a user who has no specific rights on this document.

Acl: The binary serialization of the **WSS ACL Format** (section 2.2.3.2) ACL for the scope in effect.

2.2.4.3 Attachment Document Information Result Set

The **Attachment Document Information Result Set** returns the **Document Identifier** (section 2.2.1.4) and the **leaf name** information of the document. **Attachment Document Information Result Set** is defined using T-SQL syntax, as follows.

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

Id: The **Document Identifier** (section 2.2.1.4) of the document.

LeafName: The leaf name of the document location.

2.2.4.4 Attachment Item Information Result Set

The **Attachment Item Information Result Set** returns the **item identifier** and the leaf name information of the document. **Attachment Item Information Result Set** is defined using T-SQL syntax, as follows.

```
A         int,
LeafName  nvarchar(128);
```

A: The item identifier of the item for which this document is an attachment.

LeafName: The leaf name of the document location.

2.2.4.5 Attachment State Result Set

The **Attachment State Result Set** contains information about the attachment state of the requested document. The **Attachment State Result Set** is defined using T-SQL syntax, as follows.

```
{IsAttachment}         bit,
{NeedManageListRight}  bit;
```

{IsAttachment}: Specifies whether the document is associated with an attachment. This value MUST be 1 if the document is an attachment, a list item attachment folder, or the list attachment folder itself. Otherwise, this value MUST be zero. See **Attachments Flag** (section 2.2.2.1) for more information.

{NeedManageListRight}: Specifies whether operations on this document require the user to have the ManageLists bit of the **WSS Rights Mask** (section 2.2.2.10) set. This value MUST be zero if the document is not stored in a **list (1)**.

2.2.4.6 Backward Link Result Set

The **Backward Link Result Set** contains the backward links for the moved (renamed) documents. If the **Backward Link Result Set** is returned, it MUST return one row for each backward link to any of the renamed documents. The **Backward Link Result Set** is defined using T-SQL syntax, as follows.

```
DocUrl nvarchar(260);
```

DocUrl: A **store-relative URL** specifying the backward link for the renamed document.

2.2.4.7 Contained Document Metadata Result Set

The **Contained Document Metadata Result Set** contains the metadata information for the documents contained within the specified document. **Contained Document Metadata Result Set** is defined using T-SQL syntax, as follows.

```
Id                          uniqueidentifier,
{FullUrl}                   nvarchar(385),
Type                        tinyint,
MetaInfoTimeLastModified    datetime,
MetaInfo                    image,
{Size}                      int,
TimeCreated                 datetime,
TimeLastModified            datetime,
Version                     int,
DocFlags                    tinyint,
{ListType}                  int,
tp Name                     nvarchar(38),
tp Title                    nvarchar(255),
CacheParseId                int,
{GhostDirName}              int,
{GhostLeafName}             int,
{tp Login}                  nvarchar(255),
{CheckOutDate}              datetime,
{CheckOutExpires}           datetime,
VirusStatus                 int,
VirusInfo                   nvarchar(255),
SetupPath                   nvarchar(255),
NextToLastTimeModified      datetime,
```

```
UIVersion                   int;
```

Id: The **Document Identifier** (section 2.2.1.4) of the contained document.

{FullUrl}: The **store-relative form** URL for this document.

Type: The **Document Store Type** (section 2.2.2.3) of this document.

MetaInfoTimeLastModified: A datetime with a timestamp in **Coordinated Universal Time (UTC)** format specifying the last time the **Metainfo** value of this document was changed. This value can be NULL.

MetaInfo: A **metadict** for this document. This value can be NULL and MUST be NULL if the **MetaInfoTimeLastModified** value is not more recent than the *@ClientTimeStamp* parameter.

{Size}: The number of bytes in the **document stream** of this document. This value can be NULL.

TimeCreated: A time stamp in UTC format specifying when this document was created.

TimeLastModified: A time stamp in UTC format. The value specifies when the document was last saved. This corresponds to the actual time when the document was last modified.

Version: A counter incremented any time a change is made to this document, and used for internal conflict detection. This value can be NULL.

DocFlags: A **Doc Flags** (section 2.2.2.2) value specifying information about the document. This value can be NULL.

{ListType}: A combination of the **List Base Type** (section 2.2.1.9) and **List Server Template** (section 2.2.1.12) values of the **list (1)** containing this document, consisting of the **List Server Template** value multiplied by 256 and added to the value of the **List Base Type**. This value can be NULL.

tp_Name: The identifier of the list (1) that contains this document. This value can be NULL.

tp_Title: This contains the **display name** of the list (1). This value can be NULL.

CacheParseId: This parameter MUST be NULL.

{GhostDirName}: This parameter MUST be NULL.

{GhostLeafName}: This parameter MUST be NULL.

{tp_Login}: If this document is currently checked out, this is the login name of the user to whom it is checked out. In all other cases, this MUST be NULL.

{CheckOutDate}: A time stamp in UTC format specifying when this document was checked out. This value can be NULL.

{CheckOutExpires}: A time stamp in UTC format specifying when the **short-term lock** for this document will expire. This value can be NULL if no short-term lock has been placed on the document. This value MUST be NULL if a user has the document checked out.

VirusStatus: A **Virus Status** (section 2.2.1.26) value specifying the current virus state of this document. This value can be NULL if this document has not been processed by a virus scanner.

VirusInfo: A string containing a provider-specific message returned by the virus scanner when it last processed this document. This value can be NULL if this document has not been processed by a virus scanner.

SetupPath: For a document that has ever been **ghosted**, this contains the **setup path** fragment relative to the base setup path where the content **stream** of this document can be found. Otherwise, this parameter MUST be NULL.

NextToLastTimeModified: The value of **TimeLastModified** from the previous time when the document was last saved. This value can be NULL.

UIVersion: A UI version number associated with the document. **UIVersion** defaults to 512, which corresponds to a **displayed version** of 1.0.

2.2.4.8 Deleted Documents Result Set

The **Deleted Documents Result Set** contains information about the deleted documents. The **Deleted Documents Result Set** is defined using T-SQL syntax, as follows.

```
{Url}     nvarchar(260),
{Type}    tinyint;
```

{Url}: The store-relative form URL of the deleted document.

{Type}: The **Document Store Type** (section 2.2.2.3) of the deleted document. If the **Document Store Type** is 0 (File) then this column MUST NOT be named. Otherwise, this column is named.

2.2.4.9 Dirty Result Set

The **Dirty Result Set** contains the information about the **dirty** state of the requested document. The **Dirty Result Set** is defined using T-SQL syntax, as follows.

```
Dirty bit;
```

Dirty: This parameter MUST be 1 if this document requires dependency update processing. Otherwise, it MUST be zero.

2.2.4.10 Document Content Stream Result Set

The **Document Content Stream Result Set** contains the document's binary **stream** and associated metadata. The **Document Content Stream Result Set** is defined using T-SQL syntax, as follows.

```
{Content}        image,
{Size}           int,
SetupPath        nvarchar(255),
Dirty            bit,
Version          int,
Id               uniqueidentifier;
```

{Content}: The document's content stream. For a ghosted document, this MUST be NULL. Otherwise, if the content is larger than the value specified in the *@ChunkSize* parameter, a single zero byte will be returned.

{Size}: The size of the document, in bytes.

SetupPath: Specifies the path from where a ghosted document was originally installed. This value MUST be NULL if the document was never **ghosted**.

Dirty: Set to 1 if this document requires dependency update processing. Otherwise, it is set to 0. If the document does not have a content stream, the value is implementation-dependent and MUST be ignored.

Version: An integer value tracking the **document version** in a linearly increasing, implementation-specific, version numbering system.

Id: The **Document Identifier** (section 2.2.1.4) of the document.

2.2.4.11 Document Information and Content (Read) Result Set

The **Document Information and Content (Read) Result Set** contains information about the content of the document stream for the **current version** of the requested document. The **Document Information and Content (Read) Result Set** is defined using T-SQL syntax, as follows.

```
{Size}          int,
SetupPath       nvarchar(255),
{Content}       image,
Dirty           bit,
DocFlags        tinyint,
DoclibRowId     int,
VirusVendorId   int,
VirusStatus     int,
VirusInfo       nvarchar(255),
Version         int,
Id              uniqueidentifier,
Version         int;
```

{Size}: The size of the requested document, in bytes.

SetupPath: Specifies the path from where a ghosted document was originally installed. This value MUST be NULL if the document was never **ghosted**.

{Content}: The document stream content of the document. For a ghosted document, content MUST be NULL. If the content is larger than the value specified in the *@ChunkSize* parameter, a single zero byte MUST be returned, and the front-end Web server can request individual chunks of content in a subsequent request.

Dirty: A bit set to 1, indicating that this document requires dependency update processing before its document stream is returned to an external agent. This field's value is implementation-dependent if this document does not have a document stream, and MUST be ignored.

DocFlags: A **Doc Flags** (section 2.2.2.2) value specifying information about the document.

DoclibRowId: The row identifier for the document within the containing document library or **list (1)**, if applicable.

VirusVendorId: The identifier of the virus scanner that processed this document. This value MUST be NULL if this document has not been processed by a virus scanner.

VirusStatus: An enumerated type specifying the current virus checks status of this document. This value MUST be NULL if the document has not been processed by a virus scanner. See **Virus Status** (section 2.2.1.26) in the Flags section for a list of valid values.

VirusInfo: A string containing a provider-specific message returned by the virus scanner when it last processed the document. This value MUST be NULL if this document has not been processed by a virus scanner.

Version: An internal version counter that is incremented any time a change is made to this document and used for internal conflict detection. This value MUST be set to the current internal version counter value for the document.

Id: The **Document Identifier** (section 2.2.1.4) of this document.

Version: An internal version counter that is incremented any time a change is made to this document and used for internal conflict detection. This value MUST be equal to the previous Version column.

2.2.4.12 Document Information and Content (Update) Result Set

The **Document Information and Content (Update) Result Set** contains information about the content of the **document stream** for the **current version** of the requested document. The **Document Information and Content (Update) Result Set** is defined using T-SQL syntax, as follows.

```
{Size}          int,
SetupPath       ntext,
{Content}       image,
Dirty           bit,
DocFlags        tinyint,
{DoclibRowId}   int,
{VirusVendorID} int,
{VirusStatus}   int,
{VirusInfo}     int,
Version         int,
Id              uniqueidentifier,
Version         int;
```

{Size}: The size of the requested document, in bytes.

SetupPath: For a document that is now or once was **ghosted**, this parameter MUST contain the setup path fragment where the content **stream** of the document can be found. This value MUST be NULL if the document was never ghosted.

{Content}: The document stream content of the document. For a ghosted document, this MUST be NULL. If the content is larger than the value specified in the *@ChunkSize* parameter, a single zero byte MUST be returned, and the front-end Web server can request individual chunks of content in a subsequent request.

Dirty: A bit set to 1, indicating that this document requires dependency update processing before its document stream is returned to an external agent. This field's value is implementation-dependent if this document does not have a document stream, and MUST be ignored.

DocFlags: A **Doc Flags** (section 2.2.2.2) value describing the document.

{DoclibRowId}: The identifier of the row in the document library that represents this document. This value MUST be NULL.

{VirusVendorID}: The identifier of the virus scanner that processed this document. This value MUST be NULL.

{VirusStatus}: A **Virus Status** (section 2.2.1.26) value specifying the current virus scan status of this document. This value MUST be NULL.

{VirusInfo}: A string containing a provider specific message returned by the virus scanner when it last processed the document. This value MUST be NULL.

Version: An internal version counter that is incremented any time a change is made to this document and used for internal conflict detection. This value MUST be set to the current internal version counter value for the document.

Id: The **Document Identifier** (section 2.2.1.4) of this document.

Version: An internal version counter that is incremented any time a change is made to this document and used for internal conflict detection. This value MUST be equal to the previous **Version** column.

2.2.4.13 Document Metadata Result Set

The **Document Metadata Result Set** returns the metadata for the document. The **Document Metadata Result Set** is defined using T-SQL syntax, as follows.

```
Id                          uniqueidentifier,
{FullUrl}                   nvarchar(385),
Type                        tinyint,
MetaInfoTimeLastModified    datetime,
MetaInfo                    image,
{Size}                      int,
TimeCreated                 datetime,
TimeLastModified            datetime,
Version                     int,
DocFlags                    tinyint,
{ListType}                  int,
{tp_Name}                   int,
{ListTitle}                 nvarchar(255),
{CacheParseId}              uniqueidentifier,
{GhostDirName}              int,
{GhostLeafName}             int,
{tp_Login}                  nvarchar(255),
{CheckoutDate}              datetime,
{CheckoutExpires}           datetime,
VirusStatus                 int,
VirusInfo                   nvarchar(255),
SetupPath                   nvarchar(255),
NextToLastTimeModified      datetime,
UIVersion                   int;
```

Id: The **Document Identifier** (section 2.2.1.4) of the requested document.

{FullUrl}: The full store-relative form URL for the document being requested.

Type: An integer identifier specifying the document's **Document Store Type** (section 2.2.2.3).

MetaInfoTimeLastModified: A time stamp in UTC format specifying the last time the **MetaInfo** value of this document was changed, which is useful for providing minimal metadata returns to clients. This value can be NULL.

MetaInfo: A **metadict** for the document. The metadict format is specified in [MS-FPSE] section 2.2.2.2.11. This value MUST be NULL if the document does not exist.

{Size}: The number of bytes in the document stream of the document. This value can be NULL.

TimeCreated: A time stamp in **UTC** format that specifies when this document was created.

TimeLastModified: A time stamp in UTC format that specifies when the document was last saved. This value does not necessarily correspond to the actual time when the document was last modified.

Version: A counter incremented any time a change is made to this document and used for internal conflict detection. This is an internal document version separate from the user-visible versioning system reflected in UIVersion.

DocFlags: A **Doc Flags** (section 2.2.2.2) value describing the document. This value can be NULL.

{ListType}: A packed combination of the **List Base Type** (section 2.2.1.9) and **List Server Template** (section 2.2.1.12) values of the associated **list (1)** for the document. This value MUST be NULL.

{tp_Name}: The identifier of the list (1) that contains the document. This value MUST be NULL.

{ListTitle}: Specifies the title of the list (1), if this document is the root folder of a list. This value MUST be NULL.

{CacheParseId}: Used for concurrency detection if two different requests attempt to perform dependency update or resetting the dirty bits on a document at the same time. If *@CheckCacheParseId* is set to 1, this field MUST reflect the value stored for the document. Otherwise, it MUST return NULL.

{GhostDirName}: A placeholder for a **directory name**. This value MUST be NULL.

{GhostLeafName}: A placeholder for a leaf name. This value MUST be NULL.

{tp_Login}: If this document exists and is currently checked out, this is the login name of the user to whom it is checked out. In all other cases, this MUST be NULL.

{CheckoutDate}: A time stamp in UTC format specifying when this document was checked out. This also includes short-term lock. This value can be NULL.

{CheckoutExpires}: A time stamp in UTC format that specifies when the short-term lock for this document will expire. If this date is in the past, the document SHOULD be treated as unlocked. This value can be NULL if no short-term lock has been placed on the document. This value MUST be NULL if a user has the document checked out.

VirusStatus: An enumerated type specifying the current virus state of this document. This value MAY be NULL if it has not been processed by a virus scanner. Valid values are listed in the **Virus Status** (section 2.2.1.26) simple data type.

VirusInfo: A string containing a provider-specific message returned by the virus scanner when it last processed the document. This value can be NULL if it has not been processed by a virus scanner.

SetupPath: Specifies the path from where a ghosted document was originally installed. This value MUST be NULL if the document was never **ghosted**.

NextToLastTimeModified: The value of **TimeLastModified** from the previous time when the document was last saved. If the **NextToLastTimeModified** value matches the **TimeLastModified** value when the rename occurred, and the client has a document that it has successfully fixed up, the client can safely submit the document to the server despite what appears to be an intervening edit to the document. This value can be NULL.

UIVersion: The **UI version** number for the document. This value MUST be NULL in the case of a document that does not exist.

2.2.4.14 Document Version Information and Content (Read) Result Set

The **Document Version Information and Content (Read) Result Set** contains information about the content of the document stream for the requested version of the document. The **Document Version Information and Content (Read) Result Set** is defined using T-SQL syntax, as follows.

```
Size            int,
{SetupPath}     nvarchar(255),
{Content}       image,
{Dirty}         int,
```

```
DocFlags          tinyint,
{DoclibRowId}     int,
VirusVendorId     int,
VirusStatus       int,
VirusInfo         nvarchar(255),
{Version}         int,
Id                uniqueidentifier,
{Version}         int;
```

Size: The size of the document stream, in bytes. This parameter can be set to NULL or to 0 for items such as sites, folders, document libraries, and lists.

{SetupPath}: Specifies the path from where a ghosted document was originally installed. This value MUST be NULL if the document was never **ghosted**.

{Content}: The document stream content of the document. For a ghosted document, content MUST be NULL. If the content is larger than the value specified in the *@ChunkSize* parameter, a single zero byte MUST be returned, and the front-end Web server can request individual chunks of content in a subsequent request.

{Dirty}: Set to 1 if the document requires dependency update processing. Otherwise, it will be 0.

DocFlags: A **Doc Flags** (section 2.2.2.2) value specifying information about the document.

{DoclibRowId}: The row identifier for the document within the containing document library or list, if applicable.

VirusVendorId: The identifier of the virus scanner that processed this document. This value MUST be NULL if this document has not been processed by a virus scanner.

VirusStatus: An enumerated type specifying the current virus checks status of this document. This value MUST be NULL if the document has not been processed by a virus scanner. See **Virus Status** (section 2.2.1.26) in the Flags section for a list of valid values.

VirusInfo: A string containing a provider-specific message returned by the virus scanner when it last processed the document. This value MUST be NULL if this document has not been processed by a virus scanner.

{Version}: An internal version counter incremented any time a change is made to this document, used for internal conflict detection. This value MUST be NULL.

Id: The **Document Identifier** (section 2.2.1.4) of this document.

{Version}: A current UI version of the document that is being requested.

2.2.4.15 Document Version Information and Content Result Set

The **Document Version Information and Content Result Set** contains information about the content of the document stream for the requested version of the document. The **Document Version Information and Content Result Set** is defined using T-SQL syntax, as follows.

```
Size              int,
{SetupPath}       int,
{Content}         image,
{Dirty}           bit,
DocFlags          tinyint,
{DoclibRowId}     int,
{VirusVendorID}   int,
{VirusStatus}     int,
{VirusInfo}       int,
```

```
{Version}        int,
{DocId}          uniqueidentifier,
{Version}        int;
```

Size: The size of the requested document, in bytes.

{SetupPath}: For a document that is now or once was **ghosted**, this parameter MUST contain the setup path fragment, where the content **stream** of the document can be found. This value MUST be NULL if the document was never ghosted.

{Content}: The document stream content of the document. For a ghosted document, this MUST be NULL. If the content is larger than the value specified in the *@ChunkSize* parameter, a single zero byte MUST be returned, and the front-end Web server can request individual chunks of content in a subsequent request.

{Dirty}: A bit set to 1, indicating that this document requires dependency update processing before its document stream is returned to an external agent. This field's value is implementation-dependent if this document does not have a document stream, and MUST be ignored.

DocFlags: A **Doc Flags** (section 2.2.2.2) value describing the document.

{DoclibRowId}: The identifier of the row in the document library that represents this document. This value MUST be NULL.

{VirusVendorID}: The identifier of the virus scanner that processed this document. This value MUST be NULL.

{VirusStatus}: A **Virus Status** (section 2.2.1.26) value specifying the current virus scan status of this document. This value MUST be NULL.

{VirusInfo}: A string containing a provider-specific message returned by the virus scanner when it last processed the document. This value MUST be NULL.

{Version}: An internal version counter that is incremented any time a change is made to this document and used for internal conflict detection. This value MUST be set to the current internal version counter value for the document.

{DocId}: The **Document Identifier** (section 2.2.1.4) of this document.

{Version}: An internal version counter that is incremented any time a change is made to this document and used for internal conflict detection. This value MUST be equal to the previous version column.

2.2.4.16 Document Version Metadata Result Set

The **Document Version Metadata Result Set** contains the metadata about the requested version of the document. The **Document Version Metadata Result Set** is defined using T-SQL syntax, as follows.

```
Id                          uniqueidentifier,
{FullUrl}                   nvarchar(256),
Type                        tinyint,
MetaInfoTimeLastModified    datetime,
MetaInfo                    image,
Size                        int,
TimeCreated                 datetime,
TimeCreated                 datetime,
Version                     int,
DocFlags                    tinyint,
{ListType}                  int,
```

```
{tp_Name}                   int,
{ListTitle}                 nvarchar(255),
{CacheParseId}              int,
{GhostDirName}              int,
{GhostLeafName}             int,
{tp Login}                  int,
{CheckoutDate}              datetime,
{CheckoutExpires}           datetime,
VirusStatus                 int,
VirusInfo                   nvarchar(255),
SetupPath                   nvarchar(255),
NextToLastTimeModified      datetime,
Version                     int;
```

Id: The **Document Identifier** (section 2.2.1.4) of the requested document.

{FullUrl}: The full store-relative form URL for the document being requested.

Type: The **Document Store Type** (section 2.2.2.3) of this document.

MetaInfoTimeLastModified: A time stamp in UTC format specifying the last time the **MetaInfo** value of this document was changed, which is useful for providing minimal metadata returns to clients. This value can be NULL.

MetaInfo: A metadict for this document version. The metadict format is specified in [MS-FPSE]. This value can be NULL.

Size: The number of bytes in the document stream of the document version requested. This value can be NULL.

TimeCreated: A time stamp in UTC format specifying when this document was created.

TimeCreated: A time stamp in UTC format specifying when the document was last saved. This corresponds to the **TimeCreated** or **TimeLastModified** of the document version requested.

Version: A counter that is incremented any time a change is made to this document and used for internal conflict detection.

DocFlags: A **Doc Flags** (section 2.2.2.2) value describing this document. This value can be NULL.

{ListType}: A packed combination of the **List Base Type** (section 2.2.1.9) and **List Server Template** (section 2.2.1.12) values of the associated **list (1)** for the document. This value MUST be NULL.

{tp_Name}: The identifier of the list (1) that contains the document. This value MUST be NULL.

{ListTitle}: Specifies the title of the list, if this document is the root folder of a list (1). This value MUST be NULL.

{CacheParseId}: Used for concurrency detection if two different requests attempt to perform dependency update or resetting the dirty bits on a document at the same time. If *@CacheParse* is set to 1, this field MUST reflect the value stored for the document. Otherwise, it MUST return NULL.

{GhostDirName}: A placeholder for a directory name. This value MUST be NULL.

{GhostLeafName}: A placeholder for a leaf name. This value MUST be NULL.

{tp_Login}: If this document exists and is currently checked out, this is the login name of the user to whom it is checked out. In all other cases, this MUST be NULL.

{CheckoutDate}: A time stamp in UTC format specifying when this document was checked out. This value can be NULL.

{CheckoutExpires}: A time stamp in UTC format specifying when the short-term lock for this document will expire. If this date is in the past, the document SHOULD be treated as unlocked. This value can be NULL if no short-term lock has been placed on the document. This value MUST be NULL if a user has the document checked out.

VirusStatus: An enumerated type specifying the current virus state of this document version. This value can be NULL if it has not been processed by a virus scanner. Valid values are listed in the **Virus Status** (section 2.2.1.26) section.

VirusInfo: A string containing a provider-specific message returned by the virus scanner when it last processed the document. This value can be NULL if it has not been processed by a virus scanner.

SetupPath: For a document that is now or once was **ghosted**, this contains the setup path fragment relative to the base setup. This value can be NULL.

NextToLastTimeModified: The value of **TimeLastModified of Docs Table** (section 2.2.5.1) from the previous time when the document was last saved. If the **NextToLastTimeModified** value matches the **TimeLastModified** value when the rename occurred and the client has a document that it has successfully fixed up, the client can safely submit the document to the server despite what appears to be an intervening edit to the document. This value can be NULL.

Version: The UI version number for the document. This value can be produced as the integer counter described earlier (the first Version field). This value MUST match *@Version*.

2.2.4.17 Document Versions Result Set

The **Document Versions Result Set** returns version information for a specified document. **Document Versions Result Set** is defined using T-SQL syntax, as follows.

```
TimeCreated      datetime,
Version          int,
Size             int,
MetaInfo         image;
```

TimeCreated: The date and time this version was created or last modified or long-term checked out.

Version: A counter that is incremented when a new document is created and used for internal conflict detection.

Size: The number of bytes in the document stream.

MetaInfo: A metadict for the document. The metadict format is specified in [MS-FPSE] section 2.2.2.2.11.

2.2.4.18 Domain Group Result Set

The **Domain Group Result Set** returns the **SystemID** (section 2.2.1.20) of the specified **member**. **Domain Group Result Set** is defined using T-SQL syntax, as follows.

```
tp_SystemId varbinary(128);
```

tp_SystemId: The **SystemID** (section 2.2.1.20) of the specified member.

2.2.4.19 Empty List Result Set

The **Empty List Result Set** is defined using T-SQL syntax, as follows.

```
{No column name} int;
```

{No column name}: Not Applicable.

2.2.4.20 Fields Information Result Set

The **Fields Information Result Set** is defined using T-SQL syntax, as follows.

```
tp_Fields ntext;
```

tp_Fields: Contains an XML fragment representation of the **field definitions**.

2.2.4.21 Globals Result Set

The **Globals Result Set** returns the settings that apply across WSS deployment. The **Globals Result Set** is defined using T-SQL syntax, as follows.

```
GlobalId                uniqueidentifier,
SchemaVersion           nvarchar(64),
UseHostHeader           bit,
UseNtAuthForDatabase    bit,
SmtpServiceId           uniqueidentifier,
MailCodePage            int,
FromAddress             nvarchar(255),
ReplyToAddress          nvarchar(255),
LastModified            datetime,
LastModifiedUser        nvarchar(255),
LastModifiedServer      nvarchar(128),
Status                  int,
Version                 varbinary(8),
Properties              ntext;
```

GlobalId: The **Global Identifier** (section 2.2.1.5) of the global settings record.

SchemaVersion: The schema version of the **configuration database**.

UseHostHeader: A bit set to 1, if the hoster header mode is used for WSS.

UseNtAuthForDatabase: A bit set to 1, if **Integrated Windows authentication** is being used for database access. Otherwise, if SQL authentication is being used, this bit is set to 0.

SmtpServiceId: The GUID for the Simple Mail Transfer Protocol (SMTP) Service specified in the global settings and is an implementation-specific capability.

MailCodePage: An integer that specifies the language code page that is used for email messages.

FromAddress: The email address identified in the From field of the SMTP settings specified in the global settings and is an implementation-specific capability.

ReplyToAddress: The email address that was placed in the **Reply To** field of the SMTP settings specified in the global settings and is an implementation-specific capability.

LastModified: The time of the last modification to the global settings record.

LastModifiedUser: The **user name** of the person who last modified the global settings record.

LastModifiedServer: The name of the server affected by the modifications.

Status: The value of the configuration status.

Version: The version value that is incremented when the **UseHostHeader** or SMTP Server Settings change.

Properties: Miscellaneous properties for the deployment.

2.2.4.22 Group Member Result Set

The **Group Member Result Set** is defined using T-SQL syntax, as follows.

```
GroupID         int,
tp_Id           int,
tp_SystemID     varbinary(128),
tp_DomainGroup  bit;
```

GroupID: The GUID of the **group** for the group assignment, which MUST be unique.

tp_Id: The **User Identifier** (section 2.2.1.24) of a principal (1).

tp_SystemID: The **SystemID** (section 2.2.1.20) of the principal (1).

tp_DomainGroup: A bit flag that specifies whether the account is a **domain group** account. This bit is set to 1 if the **principal (2)** is a domain group. Otherwise, the bit is set to 0, specifying that the principal is a user.

2.2.4.23 Group Membership Token Result Set

The **Group Membership Token Result Set** returns the identifier of the site group. **Group Membership Token Result Set** is defined using T-SQL syntax, as follows.

```
GID int;
```

GID: The identifier of the site group to which the user belongs.

2.2.4.24 HTTP Document Metadata Result Set

The **HTTP Document Metadata Result Set** returns the core document metadata, including the effective ACL and anonymous permission mask. This result set MUST be returned if the specified document exists. The **HTTP Document Metadata Result Set** is defined using T-SQL syntax, as follows.

```
{Size}                          int,
{DocFlags}                      tinyint,
{FullUrl}                       nvarchar(260),
{WebId}                         uniqueidentifier,
{FirstUniqueWebId}              uniqueidentifier,
{SecurityProvider}              uniqueidentifier,
{Dirty}                         bit,
{TimeLastWritten}               datetime,
{CharSet}                       int,
{Version}                       int,
{DocId}                         uniqueidentifier,
{LeafName}                      nvarchar(128),
InDocLibrary                    int,
IsAttachment                    int,
NeedManageListRight             int,
{SiteFlags}                     int,
```

```
Acl                                 image,
{AnonymousPermMask}                 int,
{ListIdForPermissionCheck}          int,
{PermCheckedAgainstUniqueList}      int,
{VirusVendorID}                     int,
{VirusStatus}                       int,
{VirusInfo}                         nvarchar(255),
{ContentModifiedSince}              bit;
```

{Size}: The size, in bytes, of the document.

{DocFlags}: A **Doc Flags** (section 2.2.2.2) value describing the document.

{FullUrl}: The complete store-relative form URL for the **site** containing the requested document.

{WebId}: The **Site Identifier** (section 2.2.1.19) of the site containing the document.

{FirstUniqueWebId}: The **Site Identifier** of the shallowest site in the site hierarchy containing this document that has the same **security scope** as the specified document.

{SecurityProvider}: The **Component Object Model (COM) CLSID** of the **security provider** for this site. This MUST be NULL for sites using the native security implementation.

{Dirty}: A bit specifying that this document requires dependency update processing before its **stream** is returned to an external agent. If this document does not have a content stream, the value is implementation-dependent and MUST be ignored.

{TimeLastWritten}: The time, in UTC format, when the document content was last modified.

{CharSet}: A **character set** associated with the document; it MAY be NULL.

{Version}: An integer incremented any time a change is made to this document, used for internal conflict detection.

{DocId}: The **Document Identifier** (section 2.2.1.4) of the requested document.

{LeafName}: The leaf name of the requested document.

InDocLibrary: If 1, the document is in a document library. Otherwise, it is not. If *@ListIdForPermissionCheck* is NULL then this is 0.

IsAttachment: If 1, the document is an attachment. Otherwise, it is not. If *@ListIdForPermissionCheck* is NULL then this is 0.

NeedManageListRight: If 1, then the user is required to have the "Manage List" right in order to read the document. Otherwise, the user is not required to have that right. If *@ListIdForPermissionCheck* is NULL then this is 0.

{SiteFlags}: A **Site Collection Flags** (section 2.2.2.7) value describing the configuration of the site collection containing the document.

Acl: The binary serialization of the **WSS ACL Format** (section 2.2.3.2) ACL that is in effect for the document.

{AnonymousPermMask}: A **WSS Rights Mask** (section 2.2.2.10) that indicates the rights granted to a user that is anonymous, or has no specific rights, to this document.

{ListIdForPermissionCheck}: The identifier for the **list (1)** containing the document. This MUST be NULL if this document is not in a list (1) or if *@ListIdForPermissionCheck* is NULL.

{PermCheckedAgainstUniqueList}: If **Lists.tp_acl** is NULL, then the value MUST be zero. Otherwise, the value MUST be 1. If *@ListIdForPermissionCheck* is NULL, then the value MUST be zero.

{VirusVendorID}: The identifier of the virus scanner that processed this document. This value MUST be NULL if the document has not been processed by a virus scanner.

{VirusStatus}: The **Virus Status** (section 2.2.1.26) of the document. This value MUST be NULL if the requested document does not exist or if it has not been processed by a virus scanner.

{VirusInfo}: A string containing a provider-specific message that is returned by the virus scanner when it last processed the document. This value MUST be NULL if the requested document does not exist or if it has not been processed by a virus scanner.

{ContentModifiedSince}: A bit indicating whether the document has been modified, depending on the validation type requested. Set to 1 if any of the following are true: the document is a dynamic document type; the document requires a dependency update; the validation type is "None" (0); the validation type is "E-tag" (1) and the value of *@ClientVersion* disagrees with the document version in the store, or the value of *@ClientId* disagrees with the **Document Identifier** (section 2.2.1.4) in the store; the validation type is "Last modified" (2) and the last modification date of the document's copy in the store is more recent than specified in *@IfModifiedSince*. In all other cases, **{ContentModifiedSince}** is set to 0.

2.2.4.25 Individual URL Security Result Set

The **Individual URL Security Result Set** contains security information about a document. The **Individual URL Security Result Set** is defined using T-SQL syntax, as follows.

```
tp_Id                   uniqueidentifier,
tp_Acl                  image,
tp_AnonymousPermMask    int,
{IsAttachment}          bit,
{NeedManageListRight}   bit,
{BaseType}              int,
{ExcludedType}          int;
```

tp_Id: The **List Identifier** (section 2.2.1.10) of the **list (1)** or document library containing the document. This can be NULL if the list (1) or document library containing the document is not present.

tp_Acl: The **WSS ACL Format** (section 2.2.3.2) ACL for the scope that is associated with the document.

tp_AnonymousPermMask: The **WSS Rights Mask** (section 2.2.2.10) that applies to an anonymous user or a user with no assigned rights, in the scope associated with the document.

{IsAttachment}: A bit flag specifying whether the document is an attachment or an attachment folder. This value MUST be 1 if the document is an attachment or attachment folder. Otherwise, it MUST be zero.

{NeedManageListRight}: Specifies whether operations on this document require the user to have the ManageLists bit of the **WSS Rights Mask** set. This value MUST be zero if the document is not stored in a list (1).

{BaseType}: The **List Base Type** (section 2.2.1.9) of the list (1) or document library containing the document. This can be NULL if the document is not in a list (1) or document library.

{ExcludedType}: Contains an enumerated value specifying whether the document is within a special folder type in the containing list (1) or document library. The following values are valid.

Value	Description
0	The document location is not contained in a special folder.
1	The document is, or is contained within, a forms folder: a folder named "Forms" within a document library.
2	The document is, or is contained within, a web image folder: a folder named "_w" within a document library.
3	The document is, or is contained within, a thumbnail folder: a folder named "_t" within a document library.
4	The document location is at the root folder of this list (1) or document library.

2.2.4.26 Item Update Result Set

The **Item Update Result Set** returns status information about the list item update. The **Item Update Result Set** is defined using T-SQL syntax as follows.

```
{ReturnCode}     int,
{RowsAffected}   int,
{Version}        int,
{Author}         int;
```

{ReturnCode}: The return code value returned by **proc_UpdateListItem** (section 3.1.5.99).

{RowsAffected}: The count of the **UserData Table** (section 2.2.5.6) rows updated by **proc_UpdateListItem**. It is an integer value, giving the number of rows affected after the update.

{Version}: An integer value that contains an implementation-specific, internal version counter used in tracking modifications to the list item.

{Author}: An integer value that contains the **User Identifier** (section 2.2.1.24) of the user who created the list item.

2.2.4.27 Link Info Result Set

The **Link Info Result Set** MUST be ordered by the DocId column. **Link Info Result Set** is defined using T-SQL syntax, as follows.

```
DocId              uniqueidentifier,
LinkDirName        nvarchar(256),
LinkLeafName       nvarchar(128),
LinkType           int,
LinkSecurity       tinyint,
LinkDynamic        tinyint,
LinkServerRel      bit,
LinkStatus         tinyint,
PointsToDir        bit,
WebPartId          int,
LinkNumber         int;
```

DocId: The **Document Identifier** (section 2.2.1.4) of the specified document

LinkDirName: Contains the directory name of the link target.

LinkLeafName: Contains the leaf name of the link target.

LinkType: The **LinkType Type** (section 2.2.1.8) value of the link. This value MUST be -1 for a backward link.

LinkSecurity: The **LinkSecurity Type** (section 2.2.1.7) value of the **forward link**, which specifies whether the scheme of the link is HTTP or HTTPS. This value MUST be NULL for a backward link.

LinkDynamic: The **LinkDynamic Type** (section 2.2.1.6) value of the forward link, which specifies whether the link is to a special type of target. This value MUST be NULL for a backward link.

LinkServerRel: A bit flag that specifies whether the link URL is server-relative or not. A value of 1 specifies a **server-relative URL**. This value MUST be NULL for a backward link.

LinkStatus: The **Document Store Type** (section 2.2.2.3) value of the document that is targeted by a forward link. This value MUST be NULL for a backward link. If the forward link target is a document that does not exist or if this link refers to a target that exists outside the specified site collection or if it refers to a location that could not be verified, this value MUST be NULL.

PointsToDir: A bit flag specifying whether the target of the forward link was a directory and has been modified to target a **Welcome page**. This value MUST be NULL for a backward link.

WebPartId: This parameter MUST be NULL.

LinkNumber: This parameter MUST be NULL.

2.2.4.28 Link Info Single Doc Fixup Result Set

The **Link Info Single Doc Fixup Result Set** contains information about forward links within the requested document. The **Link Info Single Doc Fixup Result Set** is defined using T-SQL syntax, as follows.

```
{NULL}          int,
LinkDirName     nvarchar(256),
LinkLeafName    nvarchar(128),
LinkType        tinyint,
LinkSecurity    tinyint,
LinkDynamic     tinyint,
LinkServerRel   bit,
LinkStatus      tinyint,
PointsToDir     bit,
WebPartId       uniqueidentifier,
LinkNumber      int;
```

{NULL}: The value always will be NULL.

LinkDirName: Contains the directory name of the link.

LinkLeafName: Contains the leaf name of the link.

LinkType: The **LinkType Type** (section 2.2.1.8) value of the link in the specified document.

LinkSecurity: A **LinkSecurity Type** (section 2.2.1.7) value specifying whether the scheme of the link is HTTP or HTTPS.

LinkDynamic: A **LinkDynamic Type** (section 2.2.1.6) value that specifies whether this link is one of several special link types.

LinkServerRel: A bit flag used to indicate whether a link URL is server-relative. When **LinkServerRel** is set to 1, the link URL is server-relative. Otherwise, the URL is not server-relative.

LinkStatus: A **Document Store Type** (section 2.2.2.3) indicating the type of document the link points to. If the link is a forward link for a document that doesn't exist, this MUST be NULL. This value MUST be NULL if this link entry refers to a location that exists outside the site collection where the document is stored, or if it refers to a location that could not be verified. See section 2.2.2.3 for a list of valid values.

PointsToDir: If the link pointed to a directory where a welcome page existed, then the link is automatically changed to be the URL to the welcome page itself. This bit MUST be set to 1 if this operation has been performed so that it can be distinguished from an explicit link to the welcome page.

WebPartId: The GUID of the Web Part to which the link belongs, if this link corresponds to a Web Part within a **Web Part zone**.

LinkNumber: An ordinal value denoting the relative order of the link within the source of the document or Web Part being processed.

2.2.4.29 Link Info Single Doc Result Set

The **Link Info Single Doc Result Set** returns information about each forward link from the document and backward link to the document. The **Link Info Single Doc Result Set** is defined using T-SQL syntax, as follows.

```
{NULL}          int,
LinkDirName     nvarchar(256),
LinkLeafName    nvarchar(128),
LinkType        int,
LinkSecurity    tinyint,
LinkDynamic     tinyint,
LinkServerRel   bit,
LinkStatus      tinyint,
PointsToDir     bit,
WebPartId       int,
LinkNumber      int;
```

{NULL}: This parameter MUST be NULL.

LinkDirName: The directory name of the link target.

LinkLeafName: The leaf name of the link target.

LinkType: The **LinkType Type** (section 2.2.1.8) value of the link. This value MUST be -1 for a backward link.

LinkSecurity: The **LinkSecurity Type** (section 2.2.1.7) value of the forward link that specifies whether the scheme of the link is HTTP or HTTPS. This value MUST be NULL for a backward link.

LinkDynamic: The **LinkDynamic Type** (section 2.2.1.6) value of the forward link that specifies whether the link is to a special type of target. This value MUST be NULL for a backward link.

LinkServerRel: A bit flag that specifies whether the link URL is **server-relative URL** or not. A value of 1 specifies a server-relative URL. This value MUST be NULL for a backward link.

LinkStatus: The **Document Store Type** (section 2.2.2.3) value of the document targeted by a forward link. This value MUST be NULL for a backward link. If the forward link target is a document that does not exist, or if this link refers to a target that exists outside the specified site collection, or if it refers to a location that could not be verified, this value MUST be NULL.

PointsToDir: A bit flag specifying whether the target of the forward link was a directory and has been modified to target a welcome page. This value MUST be NULL for a backward link. For a forward link,

if the link target is a directory where a welcome page is specified, the link MUST be changed to the URL of the welcome page and **PointsToDir** MUST be set to 1 so that the link can be distinguished from an explicit link to the welcome page. Otherwise, this value MUST be zero.

WebPartId: This parameter MUST be NULL.

LinkNumber: This parameter MUST be NULL.

2.2.4.30 List Access Result Set

The **List Access Result Set** is defined using T-SQL syntax, as follows.

```
tp_Flags int;
```

tp_Flags: A **List Flags** (section 2.2.2.4) value describing this **list (1)**.

2.2.4.31 List Information Result Set

The **List Information Result Set** returns information about Lists contained in the specified **site**. The **List Information Result Set** is defined using T-SQL syntax, as follows.

```
tp_DocTemplateUrl      nvarchar(386),
tp_DefaultViewUrl      nvarchar(386),
tp_ID                  uniqueidentifier,
tp_Title               nvarchar(255),
tp_Description         ntext,
tp_ImageUrl            nvarchar(255),
tp_Name                nvarchar(38),
tp_BaseType            int,
tp_ServerTemplate      int,
tp_Created             datetime,
tp_Modified            datetime,
tp_LastDeleted         datetime,
tp_Version             int,
tp_Direction           int,
tp_ThumbnailSize       int,
tp_WebImageWidth       int,
tp_WebImageHeight      int,
tp_Flags               int,
tp_ItemCount           int,
tp_AnonymousPermMask   int,
tp_RootFolder          nvarchar(260),
tp_ReadSecurity        int,
tp_WriteSecurity       int,
tp_Author              int,
tp_EventSinkAssembly   nvarchar(255),
tp_EventSinkClass      nvarchar(255),
tp_EventSinkData       nvarchar(255),
tp_EmailInsertsFolder  nvarchar(255),
tp_ACL                 image;
```

tp_DocTemplateUrl: Contains the store-relative form URL of the **document template** that is associated with the **list (1)**, if any, or NULL if none.

tp_DefaultViewUrl: Contains the store-relative form URL of the **default view** of the list (1).

tp_ID: Contains the **List Identifier** (section 2.2.1.10) for the list (1).

tp_Title: Contains the display name, a short user-provided text description to identify the list (1).

tp_Description: Contains user-provided text describing the list (1).

tp_ImageUrl: Contains the store-relative form URL that holds an image associated with the list (1).

tp_Name: Contains a string representation of the **tp_ID** GUID, delimited by braces.

tp_BaseType: Contains the **List Base Type** (section 2.2.1.9) value from which the list (1) is derived.

tp_ServerTemplate: Contains the value of the **list template** that defines the base structure of the list (1). This value can either be in the **List Server Template** (section 2.2.1.12) enumeration or can be a custom value as defined in the disk-based template (a template known to all web frontend clients) of the list (1). A custom list template value SHOULD be unique and more than 10000.

tp_Created: Contains a time stamp, in UTC format, specifying when the list (1) was created.

tp_Modified: Contains a time stamp, in UTC format, specifying when the list (1) was last modified.

tp_LastDeleted: Contains a time stamp, in UTC format, specifying when a list item was last deleted from the list (1). If no list item has been deleted, contains the value of **tp_Created**.

tp_Version: Contains an implementation-specific, internal version counter used in tracking modifications to the list's settings.

tp_Direction: Contains a value specifying the direction of text flow for front-end Web server elements. The following values are valid.

Value	Description
0	No explicit direction is specified.
1	Text flow is left to right.
2	Text flow is right to left.

tp_ThumbnailSize: The width, in pixels, that is specified for use when creating thumbnail images of list items within this list (1). This value MUST be NULL for lists that do not have a **List Server Template** value of 109 (Image Library Template). Thumbnails and web images are front-end Web server-generated images of documents, and they are implementation-specific capabilities.

tp_WebImageWidth: The width, in pixels, that is specified for use when creating web images of list items within this list (1). This value MUST be NULL for lists that do not have a **List Server Template** value of 109 (Image Library Template).

tp_WebImageHeight: The height, in pixels, specified for use when creating web images of list items within this list (1). This value MUST be NULL for lists that do not have a **List Server Template** value of 109 (Image Library Template).

tp_Flags: Contains a **List Flags** (section 2.2.2.4) value describing list properties.

tp_ItemCount: Contains a count of list items in the list (1).

tp_AnonymousPermMask: Contains the **WSS Rights Mask** (section 2.2.2.10) that applies to an anonymous user in the scope of the list (1).

tp_RootFolder: The store-relative form URL of the folder in which the list (1) is stored. This value MUST be returned if the root folder is requested. Otherwise, this MUST be NULL.

tp_ReadSecurity: Contains a value identifying the security policy for read access on list items. The following values are valid.

Value	Description
1	Users with read permissions can read all list items.
2	Users with read permissions can only read their own items.

tp_WriteSecurity: Contains a value specifying the security policy in use for write access to list items. The following values are valid.

Value	Description
1	Users with write permissions have write access to all list items.
2	Users with write permissions have write access to their own list items only.
4	Users have no write access to any list items.

tp_Author: Contains the **User Identifier** (section 2.2.1.24) of the user who created the list (1).

tp_EventSinkAssembly: Contains the **assembly name** of the **event sink** handler if an event sink handler is registered for the list (1), or NULL if none exists.

tp_EventSinkClass: Contains the assembly class identifier of the event sink handler if an event sink handler is registered for the list (1), or NULL if none exists.

tp_EventSinkData: Contains Unicode string data specific to the implementation of the event sink associated with this list (1), or NULL if none exists.

tp_EmailInsertsFolder: Contains a directory name specifying the directory on the configured email inserts server that SHOULD be inspected for new email messages to be processed for this list (1). It can be NULL if the list is not email-enabled.

tp_ACL: When *@FAclInfo* is set to 1, this MUST contain a binary image of the ACL for this list (1) if one is defined. Otherwise, it MUST be NULL.

2.2.4.32 List Metadata Result Set

The **List Metadata Result Set** contains the metadata associated with the **list (1)** containing the requested document. The **List Metadata Result Set** is defined using T-SQL syntax, as follows.

```
tp_ID                       uniqueidentifier,
tp_Title                    nvarchar(255),
tp_Modified                 datetime,
tp_Created                  datetime,
tp_LastDeleted              datetime,
tp_Version                  int,
tp_BaseType                 int,
tp_ServerTemplate           int,
DirName                     nvarchar(256),
LeafName                    nvarchar(128),
DirName                     nvarchar(256),
LeafName                    nvarchar(128),
tp_ReadSecurity             int,
tp_WriteSecurity            int,
tp_Description              ntext,
tp_Fields                   ntext,
tp_Direction                int,
tp_AnonymousPermMask        int,
tp_Flags                    int,
tp_ThumbnailSize            int,
tp_WebImageWidth            int,
tp_WebImageHeight           int,
```

```
tp_ImageUrl                  nvarchar(255),
tp_ItemCount                 int,
tp_Author                    int,
tp_ACL                       image,
tp_EventSinkAssembly         nvarchar(255),
tp_EventSinkClass            nvarchar(255),
tp_EventSinkData             nvarchar(255),
tp_EmailInsertsFolder        nvarchar(255),
tp_EmailInsertsLastSyncTime  nvarchar(50);
```

tp_ID: The **List Identifier** (section 2.2.1.10) of the list (1).

tp_Title: The title of the list (1) for display in the user interface.

tp_Modified: A time stamp in UTC format specifying when the list item within the list (1) was last modified.

tp_Created: A time stamp in UTC format specifying when the list (1) was created.

tp_LastDeleted: A time stamp in UTC format specifying when an item was last deleted from the list (1).

tp_Version: A counter incremented any time a change is made to the list (1), used for internal conflict detection.

tp_BaseType: The **List Base Type** (section 2.2.1.9) value from which the list (1) is derived.

tp_ServerTemplate: The **List Server Template** (section 2.2.1.12) enumeration value of the list template that defines the base structure of the list (1).

DirName: The directory name of the location that contains the list (1).

LeafName: The leaf name of the location that contains the list (1).

DirName: The directory name of the default template of the list (1).

LeafName: The leaf name of the default template of the list (1).

tp_ReadSecurity: Special restrictions that can be placed on list item access. The following values are valid.

Value	Description
1	No special restrictions.
2	Users will see only their own list items. The front-end Web server MUST NOT display list items to users without the ManageLists right unless the list item was created by that user (for example, tp_Author = *@UserId*).

tp_WriteSecurity: Special restrictions that can be placed on list item update. The following values are valid.

Value	Description
1	No special restrictions.
2	Users will see only their own list items. The front-end Web server MUST NOT allow users without the ManageLists right to update a list item unless the list item was created by that user (for example, tp_Author = *@UserId*).
4	Users will not update any list items in the list (1). The front-end Web server MUST NOT allow users

Value	Description
	without the ManageLists right to add or update list items in the list (1).

tp_Description: The description of this list (1) for display in the front-end Web server.

tp_Fields: Contains an XML fragment representation of the field definitions. This field can be NULL if the list (1) has not been customized.

tp_Direction: An enumerated value specifying the direction of text flow for front-end Web server elements presented by this list (1). The following values are valid.

Value	Description
0	No explicit direction is specified.
1	Text flow is left to right.
2	Text flow is right to left.

tp_AnonymousPermMask: A **WSS Rights Mask** (section 2.2.2.10) that indicates the rights granted to a user that is anonymous, or has no specific rights, on this list (1).

tp_Flags: A **List Flags** (section 2.2.2.4) value describing this list (1).

tp_ThumbnailSize: The width, in pixels, specified for use when creating thumbnail images of list items within this list (1). This value MUST be NULL for lists that do not have a **List Server Template** value of 109 (Image Library Template). Thumbnails and web images are front-end Web server generated images of documents that are implementation-specific capabilities.

tp_WebImageWidth: The width, in pixels, that is specified for use when creating web images of list items within this list (1). This value MUST be NULL for lists that do not have a **List Server Template** value of 109 (Image Library Template).

tp_WebImageHeight: The height, in pixels, that is specified for use when creating web images of list items within this list (1). This value MUST be NULL for lists that do not have a **List Server Template** value of 109 (Image Library Template).

tp_ImageUrl: The URL of the image that is used to represent this list (1).

tp_ItemCount: The number of list items that are stored within this list (1).

tp_Author: The identifier of the user who created the list (1).

tp_ACL: The permissions for this list (1). If **tp_ACL** is NULL, then the permissions of the web to which this list belongs apply to this list (1).

tp_EventSinkAssembly: The name of the .NET assembly that contains the class definition that implements the **event sink** associated with this list (1). This value MUST default to NULL if no event sink has been associated with the list (1).

tp_EventSinkClass: The name of the class definition that implements the event sink associated with this list (1). This value MUST default to NULL if no event sink has been associated with the list (1).

tp_EventSinkData: Additional data persisted on behalf of the event sink implementation, to be passed to the event sink associated with this list (1). This value MUST default to NULL if no event sink has been associated with the list (1).

tp_EmailInsertsFolder: Contains a directory name specifying the directory on the configured email inserts server that SHOULD be inspected for new email messages to be processed for this list (1). If

the **List Flags** for this list do not have the value 0x00010000 set, this field MUST be ignored. This value MUST default to NULL if no email inserts server has been associated with the list (1).

tp_EmailInsertsLastSyncTime: This field is a UTC time stamp encoded as a Unicode string in yyyy-mm-dd hh:mi:ss.mmm format, specifying the last time the location specified in the **tp_EmailInsertsFolder** column was inspected for new list items. If the **List Flags** value for this list (1) does not have the value 0x00010000 set, this field MUST be ignored. This value MUST default to NULL if no email inserts server has been associated with the list (1).

2.2.4.33 List Web Parts Result Set

The **List Web Parts Result Set** contains information about the Web Parts related to the lists. The **List Web Parts Result Set** is defined using T-SQL syntax, as follows.

```
tp_ListID          uniqueidentifier,
tp_Type            tinyint,
tp_ID              uniqueidentifier,
tp_Flags           int,
tp_DisplayName     nvarchar(255),
tp_PageUrl         nvarchar(385),
tp_BaseViewId      tinyint;
```

tp_ListID: The **List identifier** (section 2.2.1.10) of the **list (1)** containing the Web Part.

tp_Type: The **Page Type** (section 2.2.1.14) of the Web Part.

tp_ID: The **Web Part Identifier** (section 2.2.1.27) of the Web Part.

tp_Flags: A **View Flags** (section 2.2.2.9) value specifying view-related settings for this Web Part.

tp_DisplayName: The display name specified for the Web Part, if any. This value can be an **empty string**.

tp_PageUrl: The store-relative form URL to the **site** that contains this Web Part.

tp_BaseViewId: The **View Identifier** (section 2.2.1.25) for the view where this Web Part is used. This value can be NULL.

2.2.4.34 List Webpart Result Set

The **List Webpart Result Set** contains information about the Web Parts related to the lists associated with a specified document. The **List Webpart Result Set** is defined using T-SQL syntax, as follows.

```
tp_ListID          uniqueidentifier,
tp_Type            tinyint,
tp_ID              uniqueidentifier,
tp_Flags           int,
tp_DisplayName     nvarchar(255),
tp_PageUrl         nvarchar(385),
tp_BaseViewId      tinyint,
tp_View            ntext;
```

tp_ListID: The list identifier (section 2.2.1.10) of the **list (1)** containing the Web Part.

tp_Type: The **Page Type** (section 2.2.1.14) of the Web Part.

tp_ID: The **Web Part Identifier** (section 2.2.1.27) of the Web Part.

tp_Flags: A **View Flags** (section 2.2.2.9) value specifying view-related settings for this Web Part.

tp_DisplayName: The display name specified for the Web Part, if any. This value can be an **empty string**.

tp_PageUrl: The store-relative form URL to the **site** that contains this Web Part.

tp_BaseViewId: The **View Identifier** (section 2.2.1.25) for the view where this Web Part is used.

tp_View: An ntext value that contains implementation-specific XML used when processing this Web Part. This field can be NULL.

2.2.4.35 Login Result Set

The **Login Result Set** returns login information. The **Login Result Set** is defined using T-SQL syntax, as follows.

```
tp_Login nvarchar(255);
```

tp_Login: The login name for the user.

2.2.4.36 Multiple Document Metadata Result Set

The **Multiple Document Metadata Result Set** returns the metadata for the specified documents. The **Multiple Document Metadata Result Set** is defined using T-SQL syntax, as follows.

```
DocId uniqueidentifier,
{FullUrl}                   nvarchar(385),
Type                        tinyint,
MetaInfoTimeLastModified    datetime,
MetaInfo                    image,
{Size}                      int,
TimeCreated                 datetime,
TimeLastModified            datetime,
Version                     int,
DocFlags                    tinyint,
{ListType}                  int,
tp_Name                     nvarchar(38),
{ListTitle}                 nvarchar(255),
{CacheParseId}              int,
GhostDirName                nvarchar(256),
GhostLeafName               nvarchar(128),
{tp_Login}                  nvarchar(255),
{CheckoutDate}              datetime,
{CheckoutExpires}           datetime,
VirusStatus                 int,
VirusInfo                   nvarchar(255),
SetupPath                   nvarchar(255),
NextToLastTimeModified      datetime,
UIVersion                   int;
```

DocId: The GUID of the requested document.

{FullUrl}: The full store-relative form URL for the document being requested.

Type: The **Document Store Type** (section 2.2.2.3) of this document.

MetaInfoTimeLastModified: A time stamp in UTC format specifying the last time the **MetaInfo** value of this document was changed, which is useful for providing minimal metadata returns to clients. This value can be NULL.

MetaInfo: A metadict for this document.

{Size}: The number of bytes in the document stream of the document. This value can be NULL.

TimeCreated: A time stamp in UTC format that specifies when this document was created.

TimeLastModified: A time stamp in UTC format that specifies when the document was last saved. This value corresponds to the actual time when the document was last modified.

Version: A counter that is incremented any time a change is made to this document and used for internal conflict detection.

DocFlags: A **Doc Flags** (section 2.2.2.2) value describing the document. This value can be NULL.

{ListType}: If the document is the root folder of the **list (1)**, this contains a packed combination of the **List Base Type** (section 2.2.1.9) and **List Server Template** (section 2.2.1.12) values of the associated list for this document, where the **List Server Template** value is multiplied by 256 and added to the value of the **List Base Type**. This value MUST be NULL if the document does not exist or if it is not in a List.

tp_Name: If this document is the root folder of the list (1), this contains the identifier of the list, formatted as a **List Identifier** (section 2.2.1.10) string. This value can be NULL.

{ListTitle}: If this document is the root folder of a list (1), this contains the title of the list for display in the front-end Web servers. This value can be NULL.

{CacheParseId}: Used for concurrency detection if two different requests attempt to perform dependency update or resetting the dirty bits on a document at the same time. This value MUST be NULL.

GhostDirName: A placeholder for a **directory name**.

GhostLeafName: A placeholder for a leaf name. This value can be NULL.

{tp_Login}: If this document exists and is currently checked out, this is the login name of the user to whom it is checked out. In all other cases, this value is NULL.

{CheckoutDate}: A time stamp in UTC format specifying when this document was checked out. This value can be NULL.

{CheckoutExpires}: A time stamp in UTC format that specifies when the short-term lock for this document will expire. If this date is in the past, the document SHOULD be treated as unlocked. This value can be NULL if no short-term lock has been placed on the document. This value MUST be NULL if a user has the document checked out.

VirusStatus: An enumerated type specifying the current virus state of this document. This value can be NULL if it has not been processed by a virus scanner. Valid values are listed in the **Virus Status** (section 2.2.1.26) section.

VirusInfo: A string containing a provider-specific message returned by the virus scanner when it last processed the document. This value can be NULL if it has not been processed by a virus scanner.

SetupPath: For a document that is now or once was **ghosted**, this contains the setup path fragment relative to the base setup. This value can be NULL.

NextToLastTimeModified: The value of **TimeLastModified** from the previous time when the document was last saved. If the **NextToLastTimeModified** value matches the **TimeLastModified** value when the rename occurred, and the client has a document that it has successfully fixed up, the client can safely submit the document to the server despite what appears to be an intervening edit to the document. This value can be NULL.

UIVersion: The UI version number for the document. This value MUST be NULL in the case of a document that does not exist.

2.2.4.37 Null Individual URL Security Result Set

The **Null Individual URL Security Result Set** indicates that a specified document is not found. The **Null Individual URL Security Result Set** is defined using T-SQL syntax, as follows.

```
{tp Id}                  int,
{tp_Acl}                 int,
{tp AnonymousPermMask}   int,
{IsAttachment}           bit,
{NeedManageListRight}    bit,
{BaseType}               int,
{ExcludedType}           int;
```

{tp_Id}: This parameter MUST be NULL.

{tp_Acl}: This parameter MUST be NULL.

{tp_AnonymousPermMask}: This parameter MUST be NULL.

{IsAttachment}: This parameter MUST be zero.

{NeedManageListRight}: This parameter MUST be zero.

{BaseType}: This parameter MUST be NULL.

{ExcludedType}: This parameter MUST be NULL.

2.2.4.38 Principal Display Information Result Set

The **Principal Display Information Result Set** returns information about a site group or web group. The **Principal Display Information Result Set** is defined using T-SQL syntax, as follows.

```
IsUser                   bit,
IsSiteGroup              bit,
UserID                   int,
UserSID                  int,
UserName                 int,
UserLogin                int,
UserEmail                int,
UserNotes                int,
UserSiteAdmin            int,
UserDomainGroup          int,
GroupID                  int,
GroupName                nvarchar(255),
GroupDescription         nvarchar(512),
GroupOwnerID             int,
GroupOwnerIsUser         bit,
GroupType                tinyint;
```

IsUser: This parameter MUST be zero.

IsSiteGroup: 1 if this row represents a site group, else 0 if this row represents a web group.

UserID: This parameter MUST be NULL.

UserSID: This parameter MUST be NULL.

UserName: This parameter MUST be NULL.

UserLogin: This parameter MUST be NULL.

UserEmail: This parameter MUST be NULL.

UserNotes: This parameter MUST be NULL.

UserSiteAdmin: This parameter MUST be NULL.

UserDomainGroup: This parameter MUST be NULL.

GroupID: The identifier of the site group if **IsSiteGroup** is 1, else the identifier of the web group.

GroupName: The title of the site group if **IsSiteGroup** is 1, else the title of the web group.

GroupDescription: The description of the site group if **IsSiteGroup** is 1, else the description of the web group.

GroupOwnerID: The identifier of owner of the site group if **IsSiteGroup** is 1, else 0.

GroupOwnerIsUser: Value that specifies whether the owner is a user of the site group if **IsSiteGroup** is 1, else 0.

GroupType: 0 if **IsSiteGroup** is 1, else the type value that indicates the built-in site group template of web group.

2.2.4.39 Principal User Information Result Set

The **Principal User Information Result Set** returns information about a **principal (1)**. The **Principal User Information Result Set** is defined using T-SQL syntax, as follows.

```
tp_ID           int,
tp_SystemID     varbinary(128),
tp_Title        nvarchar(255),
tp_Login        nvarchar(255),
tp_Email        nvarchar(255),
tp_Notes        nvarchar(1023),
tp_SiteAdmin    bit,
tp_DomainGroup  bit;
```

tp_ID: The **User Identifier** (section 2.2.1.24) of the principal (1).

tp_SystemID: The **SystemID** (section 2.2.1.20) of the principal (1).

tp_Title: The display name of the principal (1).

tp_Login: The login name of the principal (1).

tp_Email: The email address of the principal (1).

tp_Notes: Contains notes associated with the principal (1).

tp_SiteAdmin: Set to 1 if the principal (1)l is a **site collection administrator**. Otherwise, the value is set to 0.

tp_DomainGroup: Set to 1 if the principal (1) is a domain group. Otherwise, the value is set to 0.

2.2.4.40 Rename Result Set

The **Rename Result Set** returns basic information about the old and new URLs for all moved (renamed) documents. When renaming a container object, affected items in the container are included in the **Rename Result Set**. If the **Rename Result Set** is returned, it MUST return one row for each

document that was renamed during the operation. The **Rename Result Set** is defined using T-SQL syntax, as follows.

```
OldDirName       nvarchar(256),
OldLeafName      nvarchar(128),
NewDirName       nvarchar(256),
NewLeafName      nvarchar(128),
Type             int;
```

OldDirName: The directory name of the document before rename.

OldLeafName: The leaf name of the document before rename.

NewDirName: The directory name of the document after rename.

NewLeafName: The leaf name of the document after rename.

Type: The **Document Store Type** (section 2.2.2.3) of the renamed document.

2.2.4.41 Request Access Email Result Set

The **Request Access Email Result Set** contains the request access email address for a specified **site**. The **Request Access Email Result Set** is defined using T-SQL syntax, as follows.

```
RequestAccessEmail nvarchar(255);
```

RequestAccessEmail: Contains the request access email address of the specified site.

2.2.4.42 Server Information Result Set

The **Server Information Result Set** returns records for the physical servers that make up the WSS deployment. The **Server Information Result Set** is defined using T-SQL syntax, as follows.

```
ServerId              uniqueidentifier,
Name                  nvarchar(128),
Address               nvarchar(128),
LastModified          datetime,
LastModifiedUser      nvarchar(255),
LastModifiedServer    nvarchar(128),
Status                int,
Version               varbinary(8),
Properties            ntext;
```

ServerId: The **Server Identifier** (section 2.2.1.16) simple type of the server.

Name: The name of the server.

Address: The address of the server.

LastModified: The time of last modification to the server record.

LastModifiedUser: The user name of the person who last modified the server record.

LastModifiedServer: The name of the server that performed the modifications.

Status: The value of the configuration status.

Version: The version value that is incremented when a new computer is added to the server farm.

Properties: Consists of additional information about the server.

2.2.4.43 Server Time Result Set

The **Server Time Result Set** returns the current time from the back-end database server in UTC. The **Server Time Result Set** is defined using T-SQL syntax, as follows.

```
{CurrentTime} datetime,
```

{CurrentTime}: The current time from the database server, in UTC format.

2.2.4.44 Single Doc Link Information Result Set

The **Single Doc Link Information Result Set** returns information about each forward link from, and each backward link to, a specified document or document version. The **Single Doc Link Information Result Set** is defined using T-SQL syntax, as follows.

```
DocId               uniqueidentifier,
LinkDirName         nvarchar(256),
LinkLeafName        nvarchar(128),
LinkType            int,
LinkSecurity        tinyint,
LinkDynamic         tinyint,
LinkServerRel       bit,
LinkStatus          tinyint,
PointsToDir         bit,
WebPartId           int,
LinkNumber          int;
```

DocId: The **Document Identifier** (section 2.2.1.4) of the specified document.

LinkDirName: Contains the directory name of the link target.

LinkLeafName: Contains the leaf name of the link target.

LinkType: The **LinkType Type** (section 2.2.1.8) value of the link in the specified document. This value MUST be -1 for a backward link.

LinkSecurity: The **LinkSecurity Type** (section 2.2.1.7) value of the forward link that specifies whether the scheme of the link is HTTP or HTTPS. This value MUST be NULL for a backward link.

LinkDynamic: The **LinkDynamic Type** (section 2.2.1.6) value of the forward link, which specifies whether the link is to a special type of target. This value MUST be NULL for a backward link.

LinkServerRel: A bit flag which specifies whether the link URL is server-relative URL or not. A value of 1 specifies a Server-Relative URL. This value MUST be NULL for a backward link.

LinkStatus: The **Document Store Type** (section 2.2.2.3) value of the document targeted by a forward link. This value MUST be NULL for a backward link. If the forward link target is a document that does not exist, or if this link refers to a target that exists outside the specified site collection, or if it refers to a location that could not be verified, this value MUST be NULL.

PointsToDir: A bit flag specifying whether the target of the forward link was a directory and has been modified to target a welcome page. This value MUST be NULL for a backward link. For a forward link, if the link target is a directory where a welcome page is specified, the link MUST be changed to the URL of the welcome page and **PointsToDir** MUST be set to 1 so that the link can be distinguished from an explicit link to the welcome page. Otherwise, this value MUST be zero.

WebPartId: This parameter MUST be NULL.

LinkNumber: This parameter MUST be NULL.

2.2.4.45 Site Acl Result Set

The **Site Acl Result Set** contains information about any **access control list (ACL)** for the specified **site**. The **Site Acl Result Set** is defined using T-SQL syntax as follows.

```
tp_WebId              uniqueidentifier,
tp_Id                 uniqueidentifier,
tp_Acl                image,
tp_AnonymousPermMask  int;
```

tp_WebId: The **Site Identifier** (section 2.2.1.19) of the site that contains the **list (1)**.

tp_Id: The GUID for the list (1).

tp_Acl: The permissions for the list (1).

tp_AnonymousPermMask: Contains the **WSS Rights Mask** (section 2.2.2.10) that applies to an anonymous user in the scope of the list (1).

2.2.4.46 Site Category Result Set

The **Site Category Result Set** contains information about categories associated with the specified **site**. The **Site Category Result Set** is defined using T-SQL syntax as follows.

```
Category nvarchar(128);
```

Category: The user-friendly name of a category defined for content in the site.

2.2.4.47 Site Collection Flags Result Set

The **Site Collection Flags Result Set** returns the **Site Collection Flags** (section 2.2.2.7) for a specified site collection. The **Site Collection Flags Result Set** is defined using T-SQL syntax, as follows.

```
{SiteCollectionFlags} int;
```

{SiteCollectionFlags}: A **Site Collection Flags** (section 2.2.2.7) value describing the site collection.

2.2.4.48 Site Group Existence Result Set

The **Site Group Existence Result Set** returns information for a given site group. The **Site Group Existence Result Set** is defined using T-SQL syntax, as follows.

```
Id              int,
{IsWebIdNull}   int;
```

Id: An integer value. Whenever a site group is added to a site collection, its value is incremented by one.

{IsWebIdNull}: Returns 1 if the GUID of the website to which the **member** belongs is NULL, else returns 0.

2.2.4.49 Site Group Information Result Set

The **Site Group Information Result Set** returns the available site group information for the specified site group. The **Site Group Information Result Set** is defined using T-SQL syntax, in the **Sec_SiteGroupsView** view (section 2.2.5.3).

2.2.4.50 Site Group Result Set

The **Site Group Result Set** returns information about a site group. The **Site Group Result Set** is defined using T-SQL syntax, in the **Sec_SiteGroupsView** view (section 2.2.5.3).

2.2.4.51 Site Metadata Result Set

The **Site Metadata Result Set** contains metadata for a **site** or sites. The **Site Metadata Result Set** is defined using T-SQL syntax, as follows.

```
Id                          uniqueidentifier,
Title                       nvarchar(255),
Description                 ntext,
MetaInfoVersion             int,
WebTemplate                 int,
Language                    int,
Locale                      int,
Collation                   smallint,
TimeZone                    smallint,
Time24                      bit,
CalendarType                smallint,
AdjustHijriDays             smallint,
ProvisionConfig             smallint,
Flags                       int,
Author                      int,
AlternateCSSUrl             nvarchar(260),
CustomJSUrl                 nvarchar(260),
AlternateHeaderUrl          nvarchar(260),
SecurityProvider            uniqueidentifier,
AnonymousPermMask           int,
{SiteFlags}                 int,
{SitePortalURL}             nvarchar(260),
{SitePortalName}            nvarchar(255),
MeetingCount                smallint,
DefTheme                    nvarchar(64),
CachedNav                   image,
CachedNavDirty              int,
FirstUniqueAncestorWebId    uniqueidentifier,
DbNow                       datetime,
ACL                         image,
FullUrl                     nvarchar(256),
Title                       nvarchar(255),
tp_Id                       int,
tp_SiteAdmin                bit,
tp_Login                    nvarchar(255),
tp Email                    nvarchar(255),
tp_Title                    nvarchar(255),

STSToken                    varbinary(4000),
{SiteSecurityVersion}       bigint,
{SiteHashKey}               varbinary(16);
```

Id: The **Site Identifier** (section 2.2.1.19) for the site.

Title: The title of the site for display in the front-end Web server.

Description: The description of the site for display in the front-end Web server.

MetaInfoVersion: A counter that is incremented any time a change is made to the metainfo for this site and used for internal conflict detection.

WebTemplate: The identifier for the template that is used in the **site definition** to define the base structure of this site.

Language: The **language code identifier (LCID)** used to determine the current UI culture of the site. Determines which language resources to use to display messages on the front-end Web server.

Locale: The LCID associated with the site that is used to determine the current culture for regional language-specific data formatting, such as currency or date and time settings.

Collation: The **Collation Order Enumeration** (section 2.2.1.3) of the site that indicates an additional sorting order that SHOULD be processed by the back-end database server. The collation method is an implementation-specific capability of the front-end Web server and back-end database server.

TimeZone: The **Time Zone Identifier** (section 2.2.1.21) for the time zone to be used when displaying time values for this site.

Time24: If set to 1, use a 24-hour time format when displaying time values for this site. Otherwise, a 12-hour time format can be used.

CalendarType: The **Calendar Type** (section 2.2.1.1) that is to be used when processing date values for this site.

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.

ProvisionConfig: An identifier of the site template used to provision this site. The following values are valid.

Value	Description
-1	This site has not had any template provision.
0	This site has Team Site or Basic Meeting workspace template applied.
1	This site has Blank Site or Blank Meeting Workspace template applied.
2	This site has Document Workspace or Decision Meeting Workspace template applied.
3	This site has Social Meeting Workspace template applied.
4	This site has Multipage Meeting Workspace template applied.

Flags: A **Site Property Flags** (section 2.2.2.8) value describing the configuration of this site.

Author: The **User Identifier** (section 2.2.1.24) of the user who is listed as creating this site.

AlternateCSSUrl: The URL for a custom CSS sheet file registered on the site for use in pages of the site.

CustomJSUrl: The URL for a custom JavaScript file registered on the site for use in pages of the site.

AlternateHeaderUrl: The URL for a custom header HTML page registered on the site for use in pages of the site when rendered on the front-end Web server.

SecurityProvider: COM class identifier (CLSID) of the external **security provider** for this site. This is NULL for sites using the native security implementation.

AnonymousPermMask: The **WSS Rights Mask** (section 2.2.2.10) that indicates the rights granted to a user that is anonymous, or has no specific rights, to this site.

{SiteFlags}: A **Site Collection Flags** (section 2.2.2.7) value that indicates the settings for the site collection that contains this site.

{SitePortalURL}: The URL for a different site registered on the site for use in navigation structures as a parent location.

{SitePortalName}: The **display name** of a different site that is registered on the site for use in navigation structures as a parent location.

MeetingCount: If this site is a meeting workspace, this value indicates the number of meetings that are configured.

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

CachedNav: An implementation-specific serialization of **navigation structure** information to display in front-end Web server elements associated with this site.

CachedNavDirty: An integer value that indicates whether the navigation MUST be recomputed.

FirstUniqueAncestorWebId: The **Site Identifier** (section 2.2.1.19) of the closest site in this site's ancestor chain that does not inherit security settings from its **parent site**.

DbNow: The current time, in UTC format, on the back-end database server.

ACL: The binary serialization of the **WSS ACL Format** (section 2.2.3.2), ACL for the site.

FullUrl: The store-relative form URL of the parent site.

Title: The title of the parent site.

tp_Id: The **User Identifier** (section 2.2.1.24) of the principal (1).

tp_SiteAdmin: A bit set to 1 if the **principal (1)** is a site administrator for the site collection.

tp_Login: The login name for the principal (1).

tp_Email: The email address for the principal (1).

tp_Title: The display name for the principal (1).

STSToken: The token for all of the site groups to which this user belongs, either directly or indirectly.

{SiteSecurityVersion}: The current security information version of the site collection.

{SiteHashKey}: Contains binary information that is used when generating digital signatures of information associated with this site collection.

2.2.4.52 Site Metainfo Result Set

The **Site Metainfo Result Set** returns metainfo about a **site**. The **Site Metainfo Result Set** is defined using T-SQL syntax, as follows.

```
MetaInfo image;
```

MetaInfo: The site metainfo in the form of a metadict. This value MAY be NULL.

2.2.4.53 Site URL Result Set

The **Site URL Result Set** MUST contain the store-relative form URL of the **site**. The **Site URL Result Set** is defined using T-SQL syntax, as follows.

```
{WebUrl} nvarchar(256);
```

{WebUrl}: The store-relative form URL of the site.

2.2.4.54 Subsite List Result Set

The **Subsite List Result Set** contains an ordered list of store-relative form URLs for all subsites within the site collection. The **Subsite List Result Set** is defined using T-SQL syntax, as follows.

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

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

2.2.4.55 User Count Result Set

The **User Count Result Set** returns the number of users registered with this site collection when configured in **Active Directory account creation mode**. When not in Active Directory account creation mode, UsersCount will always be 1. The **User Count Result Set** is defined using T-SQL syntax, as follows.

```
UsersCount      int,
UserQuota       int;
```

UsersCount: Contains the integer value for the number of users registered with the specified site collection when configured in Active Directory account creation mode. In any other configuration, **UsersCount** MUST return 1.

UserQuota: The maximum number of users that the site collection can contain. A value of 0 indicates that no limit is set.

2.2.4.56 User Display Information Result Set

The **User Display Information Result Set** returns information about a user. The **User Display Information Result Set** is defined using T-SQL syntax, as follows.

```
IsUser              bit,
IsSiteGroup         bit,
UserID              int,
UserSID             varbinary(128),

UserName            nvarchar(255),
UserLogin           nvarchar(255),
UserEmail           nvarchar(255),
UserNotes           nvarchar(1023),
UserSiteAdmin       bit,
UserDomainGroup     bit,
GroupID             int,
GroupName           int,
```

```
GroupDescription     int,
GroupOwnerID         int,
GroupOwnerIsUser     int,
GroupType            int;
```

IsUser: This parameter MUST be 1.

IsSiteGroup: This parameter MUST be zero.

UserID: The identifier of the user.

UserSID: The **SystemID** (section 2.2.1.20) of the user.

UserName: The user's name as used for display.

UserLogin: The user's login name.

UserEmail: The user's email address.

UserNotes: A longer description text associated with the **security principal**.

UserSiteAdmin: A bit that is set to 1 if the **principal (1)** is a site administrator for the site collection.

UserDomainGroup: A bit flag that specifies whether the account is a domain group account. A bit that is set to 1 if the principal (1) is a domain group. Otherwise, the bit is set to 0, specifying that the principal is a user.

GroupID: This parameter MUST be NULL.

GroupName: This parameter MUST be NULL.

GroupDescription: This parameter MUST be NULL.

GroupOwnerID: This parameter MUST be NULL.

GroupOwnerIsUser: This parameter MUST be NULL.

GroupType: This parameter MUST be NULL.

2.2.4.57 User ID Result Set

The **User ID Result Set** returns the **User Identifier** (section 2.2.1.24) for the user who is being removed from site collection. **User ID Result Set** is defined using T-SQL syntax, as follows.

```
{UserId} int;
```

{UserId}: The **User Identifier** for the user who is being removed from site collection.

2.2.4.58 User Identifier Result Set

The **User Identifier Result Set** returns the Unique Identifier of the **principal (1)**. The **User Identifier Result Set** is defined using T-SQL syntax, as follows.

```
{UserId} int;
```

{UserId}: A **UserId** is a 4-byte integer value used to uniquely identify a principal (1) within a site collection.

2.2.4.59 User Information Result Set

The **User Information Result Set** returns information about a specified user. The **User Information Result Set** is defined using T-SQL syntax, as follows.

```
tp_Id                 int,
tp_SiteAdmin          bit,
tp_Login              nvarchar(255),
tp_Email              nvarchar(255),
tp_Title              nvarchar(255),
STSToken              varbinary(4000),
SiteSecurityVersion   bigint;
```

tp_Id: The identifier of the specified user.

tp_SiteAdmin: Indicates whether the specified user is an administrator on the site collection. This is set to 1 if the **principal (1)** is a site administrator for the site collection.

tp_Login: The login name of the specified user.

tp_Email: The email address of the specified user.

tp_Title: The display name of the specified user.

STSToken: Contains a unique string for the user (SID) and a list of attributes for the user, specifying the site group membership of the specified user.

SiteSecurityVersion: The current security information version of the site collection containing this document.

2.2.4.60 Users Web Groups Result Set

The **Users Web Groups Result Set** returns all the site groups for all users in a **site**. The **Users Web Groups Result Set** is defined using T-SQL syntax, as follows.

```
ID                    int,
Title                 nvarchar(255),
Description           nvarchar(512),
WebID                 uniqueidentifier,
Owner                 int,
OwnerIsUser           bit,
OwnerGlobal           bit,
Type                  tinyint,
Global                bit,
SiteId                uniqueidentifier,
UserID                int,
UserSID               int,
UserName              int,
UserLogin             int,
UserEmail             int,

UserNotes             int,
UserSiteAdmin         int,
UserDomainGroup       int,
GroupID               int,
GroupName             int,
GroupDescription      int,
GroupSiteID           int,
GroupOwner            int,
```

```
GroupOwnerIsUser        int,
MemberID                int;
```

ID: The identifier for the site group. This parameter MUST NOT be NULL.

Title: The user-friendly display name for the site group. This parameter MUST NOT be NULL.

Description: The description of the site group.

WebID: The **Site Identifier** (section 2.2.1.19) of the site group. This parameter MUST NOT be NULL.

Owner: The **User Identifier** (section 2.2.1.24) or the **Site Collection Identifier** (section 2.2.1.17) of the site group owner. This parameter MUST be zero.

OwnerIsUser: A bit flag specifying whether the user is a site group owner or a site group **member**. This parameter MUST be zero.

OwnerGlobal: A bit flag specifying whether the view contains ownership information for a user or **domain group**, or for a site group. This parameter MUST be zero.

Type: An integer value that indicates the built-in site group template.

Global: This parameter MUST be zero.

SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection. This parameter MUST NOT be NULL.

UserID: This parameter MUST be NULL.

UserSID: This parameter MUST be NULL.

UserName: This parameter MUST be NULL.

UserLogin: This parameter MUST be NULL.

UserEmail: This parameter MUST be NULL.

UserNotes: This parameter MUST be NULL.

UserSiteAdmin: This parameter MUST be NULL.

UserDomainGroup: This parameter MUST be NULL.

GroupID: This parameter MUST be NULL.

GroupName: This parameter MUST be NULL.

GroupDescription: This parameter MUST be NULL.

GroupSiteID: This parameter MUST be NULL.

GroupOwner: This parameter MUST be NULL.

GroupOwnerIsUser: This parameter MUST be NULL.

MemberID: The **User Identifier** (section 2.2.1.24) of the user added to the site group.

2.2.4.61 Web Group Information Result Set

The **Web Group Information Result Set** returns information about a site group. The **Web Group Information Result Set** is defined using T-SQL syntax, in the **Sec_WebGroupsView** view (section 2.2.5.4).

2.2.4.62 Web Part Info Result Set

The **Web Part Info Result Set** returns information about Web Parts. The **Web Part Info Result Set** is defined using T-SQL syntax, as follows.

```
tp_ID                   uniqueidentifier,
tp_Source               ntext,
tp_AllUsersProperties   image;
```

tp_ID: The **Web Part Identifier** (section 2.2.1.27) of the Web Part.

tp_Source: Properties of the Web Part.

tp_AllUsersProperties: Properties specified for all users. This value can be NULL.

2.2.4.63 Web Parts Metadata (Nonpersonalized) Result Set

The **Web Parts Metadata (Nonpersonalized) Result Set** contains the core metadata about the Web Parts that appear on a document. The **Web Parts Metadata (Nonpersonalized) Result Set** is defined using T-SQL syntax, as follows.

```
tp_ID                   uniqueidentifier,
tp_ZoneID               nvarchar(64),
tp_WebPartTypeId        uniqueidentifier,
tp IsIncluded           bit,
tp FrameState           tinyint,
tp AllUsersProperties   image,
tp_PerUserProperties    image,
tp_PartOrder            int,
{tp_Flags}              int,
{AllUsers}              int,
tp_Cache                image,
{Pre tp Cache}          int,
{tp_ListId}             nvarchar(38),
tp_Type                 tinyint,
tp_Source               ntext,
tp_View                 ntext;
```

tp_ID: The **Web Part Identifier** (section 2.2.1.27) of the Web Part. This value MUST NOT be NULL.

tp_ZoneID: The name of a Web Part zone. This value can be NULL.

tp_WebPartTypeId: A 16-byte value uniquely identifying the type of the Web Part. This value MUST NOT be NULL.

tp_IsIncluded: If 1, this indicates that the Web Part is visible. This value MUST NOT be NULL.

tp_FrameState: A value that indicates the frame state of the Web Part. The following values are valid.

Value	Description
0	Normal. The Web Part is displayed in its normal state, with title, content, and placement within the page.
1	Minimized. The Web Part is collapsed so that only the title portion of the frame appears.

tp_AllUsersProperties: Properties specified for all users. This value can be NULL.

tp_PerUserProperties: Properties specified on a per-user basis. This value can be NULL.

tp_PartOrder: An ordinal number that indicates the location of the Web Part in relation to other Web Parts in the same zone. This value can be NULL.

{tp_Flags}: A **View Flags** (section 2.2.2.9) value that specifies view-related settings for this Web Part. This value MUST NOT be NULL.

{AllUsers}: A flag that indicates whether customization or personalization is in effect on this Web Parts page. This value MUST be equal to 1, indicating customization.

tp_Cache: A private data cache for the Web Part. This value can be NULL.

{Pre_tp_Cache}: A private data cache for the Web Part. This value MUST be NULL.

{tp_ListId}: The **List Identifier** (section 2.2.1.10) of the **list (1)** to which this Web Part refers, enclosed in braces. If not referencing a list, this value MUST be NULL.

tp_Type: The **Page Type** (section 2.2.1.14) of this Web Part. This value can be NULL.

tp_Source: Properties of the Web Part as specified by a Windows SharePoint Services-compatible HTML editor. This value can be NULL.

tp_View: An ntext value containing implementation-specific XML used when processing this Web Part. If this Web Part is not a view, this MUST be NULL.

2.2.4.64 Web Parts Metadata (Personalized) Result Set

The **Web Parts Metadata (Personalized) Result Set** contains the core metadata about the Web Parts that appear on the specified document, personalized for a specified user. The **Web Parts Metadata (Personalized) Result Set** is defined using T-SQL syntax, as follows.

```
tp_ID                       uniqueidentifier,
{tp_ZoneID}                 nvarchar(64),
tp_WebPartTypeId            uniqueidentifier,
{tp_IsIncluded}             bit,
{tp_FrameState}             tinyint,
tp_AllUsersProperties       image,
{tp_PerUserProperties}      image,
{tp_PartOrder}              int,
{tp_Flags}                  int,
{AllUsers}                  int,
tp_Cache                    image,
tp_Cache                    image,
{tp_ListId}                 nvarchar(38),
tp_Type                     tinyint,
tp_Source                   ntext,
tp_View                     ntext;
```

tp_ID: The **Web Part Identifier** (section 2.2.1.27) of the Web Part. This value MUST NOT be NULL.

{tp_ZoneID}: The name of a Web Part zone. This value can be NULL.

tp_WebPartTypeId: A 16-byte value uniquely identifying the type of the Web Part. This MUST NOT be NULL.

{tp_IsIncluded}: If 1, this indicates that the Web Part is visible. This value MUST be 1 or 0.

{tp_FrameState}: A value that indicates the frame state of the Web Part. The following values are valid.

Value	Description
0	Normal: The Web Part is displayed in its normal state, with title, content, and placement within the page.
1	Minimized: The Web Part is collapsed so that only the title portion of the frame appears.

tp_AllUsersProperties: Properties specified for all users. This value can be NULL.

{tp_PerUserProperties}: Properties specified for per-user basis. This value can be NULL.

{tp_PartOrder}: An ordinal number that indicates the location of the Web Part in relation to other Web Parts in the same zone. This value can be NULL.

{tp_Flags}: A **View Flags** (section 2.2.2.9) value that specifies view-related settings for this Web Part. This value can be equal to 0.

{AllUsers}: A flag that indicates whether customization or personalization is in effect on this Web Parts page. The following values are valid.

Value	Description
1	User not specified; changes apply to all users of this Web Part instance. Same indicates customization mode.
2	Personalization is in effect; changes apply to the **tp_UserId** in the Personalization Table.
3	Personalization is in effect; changes apply to the **tp_UserId** in the Web Parts Table.

tp_Cache: A private data cache for the Web Part. This value can be NULL.

tp_Cache: A private data cache for **published** documents. This value can be NULL.

{tp_ListId}: The **List Identifier** (section 2.2.1.10) of the **list (1)** to which this Web Part refers, enclosed in braces. If not referencing a list (1), this value MUST be NULL.

tp_Type: The **Page Type** (section 2.2.1.14) of this Web Part. This value can be NULL.

tp_Source: Properties of the Web Part, as specified by a Windows SharePoint Services-compatible HTML editor. This value can be NULL.

tp_View: This parameter MUST be NULL.

2.2.4.65 Web Url Result Set

The **Web Url Result Set** returns the site URL of a site collection. The **Web Url Result Set** is defined using T-SQL syntax, as follows.

```
{WebUrl} nvarchar(256);
```

{WebUrl}: A string containing the site URL.

2.2.4.66 Welcome Pages Result Set

The **Welcome Pages Result Set** returns the list of configured default welcome page names. The **Welcome Pages Result Set** is defined using T-SQL syntax, as follows.

```
LeafName nvarchar(128);
```

LeafName: A string containing the configured welcome page name in leaf name format (for example, "default.aspx", "default.htm").

2.2.4.67 Zone ID Result Set

The **Zone ID Result Set** returns a list of Web Part zone names. The **Zone ID Result Set** is defined using T-SQL syntax, as follows.

```
tp_ZoneID nvarchar(64);
```

tp_ZoneID: The name of a Web Part zone.

2.2.5 Tables and Views

2.2.5.1 Docs Table

The **Docs Table** stores data for all documents in the SharePoint Store. The **Docs Table** is defined using T-SQL syntax as follows.

```
TABLE Docs(
Id                              uniqueidentifier  NOT NULL,
SiteId                          uniqueidentifier  NOT NULL,
DirName                         nvarchar(256)     NOT NULL,
LeafName                        nvarchar(128)     NOT NULL,
WebId                           uniqueidentifier  NOT NULL,
ListId                          uniqueidentifier  NULL,
DoclibRowId                     int               NULL,
Type                            tinyint           NOT NULL,
Size                            int               NULL,
MetaInfoSize                    int               NULL,
Version                         int               NULL,
UIVersion                       int               NOT NULL DEFAULT 512,
Dirty                           bit               NULL,
CacheParseId                    uniqueidentifier  NULL,
DocFlags                        int               NULL,
ThicketFlag                     bit               NULL DEFAULT 0,
CharSet                         int               NULL,
TimeCreated                     datetime          NOT NULL,
TimeLastModified                datetime          NOT NULL,
NextToLastTimeModified          datetime          NULL,
MetaInfoTimeLastModified        datetime          NULL,
TimeLastWritten                 datetime          NULL,
SetupPath                       nvarchar(255)     NULL,
CheckoutUserId                  int               NULL,
CheckoutDate                    datetime          NULL,
CheckoutExpires                 datetime          NULL,
CheckoutSize                    int               NULL,
VersionCreatedSinceSTCheckout   bit               NOT NULL DEFAULT 0,
LTCheckoutUserId                int               NULL,
VirusVendorID                   int               NULL,
VirusStatus                     int               NULL,

VirusInfo                       nvarchar(255)     NULL,
```

```
    MetaInfo                    image           NULL,
    Content                     image           NULL,
    CheckoutContent             image           NULL
    );
```

Id: The **Document Identifier** (section 2.2.1.4) of the document.

SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the document.

DirName: The directory name of the document location.

LeafName: The leaf name of the document location.

WebId: The **Site Identifier** (section 2.2.1.19) of the site containing the document.

ListId: The **List Identifier** (section 2.2.1.10) of the **list (1)** containing the document, if any, or NULL if the document is not contained in a list.

DoclibRowId: The row identifier for the document within the containing document library or list (1), if applicable.

Type: An integer identifier specifying the document's **Document Store Type** (section 2.2.2.3).

Size: The size of the document stream, in bytes. This parameter can be set to NULL or to 0 for items such as sites, folders, document libraries, and lists.

MetaInfoSize: The size, in bytes, of the document metadata info.

Version: A counter that is incremented any time a change is made to this document and used for internal conflict detection. This is an internal document version that is separate from the user-visible versioning system reflected in **UIVersion**.

UIVersion: A UI version number associated with the document. **UIVersion** defaults to 512, which corresponds to a displayed version of 1.0.

Dirty: Set to 1 if the document requires dependency update processing. Otherwise, it MUST be zero.

CacheParseId: A SharePoint implementation-specific identifier for an internal dependency updates process. Used to manage bulk updates to documents when processed for dependency fix-up.

DocFlags: A **Doc Flags** (section 2.2.2.2) value specifying information about the document.

ThicketFlag: If set to 1, indicates that the document is an auxiliary thicket file, one of a set of files supporting a **thicket**.

CharSet: An optional **character set** associated with the document. The valid values are defined in **CharSet Enumeration** (section 2.2.1.2). This value can be NULL, indicating that no character set is specified for the document.

TimeCreated: A time stamp in UTC format specifying when this document was created.

TimeLastModified: A time stamp in UTC format. The value specifies when the document was last saved. This corresponds to the actual time when the document was last modified.

NextToLastTimeModified: The value of **TimeLastModified** from the previous time that the document was saved.

MetaInfoTimeLastModified: A time stamp in UTC format specifying when the metadata information for the document was last changed.

TimeLastWritten: A time stamp in UTC format specifying when any changes were made to the document stream.

SetupPath: Contains the path of the location from where a ghosted document was originally installed. This value MUST be NULL if the document was never **ghosted**.

CheckoutUserId: If the document is checked out, this parameter contains the **User Identifier** (section 2.2.1.24) of the user who has the document checked out. Otherwise, this parameter is NULL.

CheckoutDate: A time stamp in UTC format indicating when this document was checked out.

CheckoutExpires: A time stamp in UTC format indicating when the short-term lock for this document will expire.

CheckoutSize: The size of the file when it was checked out.

VersionCreatedSinceSTCheckout: If this parameter is 1, the document version has been incremented since the document last had a short-term lock established. This is used to prevent more than one new version of the document from being created while a short-term lock is established.

LTCheckoutUserId: If the document is currently checked out, this parameter is a calculated column containing the value of **CheckoutUserId**. Otherwise, this parameter is NULL.

VirusVendorID: The identifier of the virus scanner that processed this document. This value MUST be NULL if this document has not been processed by a virus scanner.

VirusStatus: An enumerated type specifying the current virus checks status of this document. This value MUST be NULL if the document has not been processed by a virus scanner. See **Virus Status** (section 2.2.1.26) in the Flags section for a list of valid values.

VirusInfo: A string containing a provider-specific message that was returned by the virus scanner when it last processed the document. This value MUST be NULL if the document has not been processed by a virus scanner.

MetaInfo: A metadict for the document. The metadict format is specified in [MS-FPSE] section 2.2.2.2.11. This value MUST be NULL if the document does not exist.

Content: A binary image holding content of the file.

CheckoutContent: A binary image holding the new content of the file, which will be overwritten to the Content field when the file is checked in. This is NULL if the document is not checked out.

2.2.5.2 Lists Table

The **Lists Table** stores information about lists that are on the SharePoint **site**. The **Lists Table** is defined using T-SQL syntax, as follows.

```
TABLE Lists(
tp_WebId                    uniqueidentifier   NOT NULL,
tp_ID                       uniqueidentifier   NOT NULL,
tp_Title                    nvarchar(255)      NOT NULL,
tp_Created                  datetime           NOT NULL,
tp_Modified                 datetime           NOT NULL,
tp_LastDeleted              datetime           NOT NULL,
tp_DeleteCount              int                NOT NULL DEFAULT (0),
tp_LastSecurityChange       datetime           NOT NULL DEFAULT (getutcdate()),
tp_Version                  int                NOT NULL,
tp_Author                   int                NULL,
tp_BaseType                 int                NOT NULL,

tp_ServerTemplate           int                NOT NULL,
tp_RootFolder               uniqueidentifier   NOT NULL,
```

```
tp_Template                 uniqueidentifier   NULL,
tp_ImageUrl                 nvarchar(255)      NOT NULL,
tp_ReadSecurity             int                NOT NULL,
tp_WriteSecurity            int                NOT NULL,
tp_AnonymousPermMask        [dbo].[tPermMask]  NULL,
tp Subscribed               bit                NOT NULL DEFAULT (0),
tp_Direction                int                NULL,
tp_Flags                    int                NOT NULL,
tp_ThumbnailSize            int                NULL,
tp_WebImageWidth            int                NULL,
tp WebImageHeight           int                NULL,
tp_ItemCount                int                NOT NULL,
tp_NextAvailableId          int                NOT NULL,
tp_Description              ntext              NOT NULL,
tp_EmailInsertsFolder       nvarchar(255)      NULL,
tp_EmailInsertsLastSyncTime nvarchar(50)       NULL,
tp ACL                      image              NULL,
tp_EventSinkAssembly        nvarchar(255)      NULL,
tp EventSinkClass           nvarchar(255)      NULL,
tp_EventSinkData            nvarchar(255)      NULL,
tp_Fields                   ntext              NULL
);
```

tp_WebId: The Site Identifier (section 2.2.1.19) of the site containing the **list (1)**.

tp_ID: A GUID for the list (1). This value MUST be unique.

tp_Title: The display name of the list (1).

tp_Created: A date and time value, in UTC format, specifying when this list (1) was created.

tp_Modified: A date and time value, in UTC format, specifying when a list item within the list was last modified.

tp_LastDeleted: A time value specifying when an item was last deleted from the list (1). Used to optimize the incremental search.

tp_DeleteCount: This value MUST be ignored. The default value is zero.

tp_LastSecurityChange: The date and time when security settings on the list (1) were last changed.

tp_Version: A value that indicates the version of the data. A counter that is incremented any time a change is made to the list (1) and used for internal conflict detection.

tp_Author: The **User Identifier** (section 2.2.1.24) for the user who created the list (1).

tp_BaseType: The **List Base Type** (section 2.2.1.9) of the list (1).

tp_ServerTemplate: The **List Server Template** (section 2.2.1.12) that was used to create the list (1).

tp_RootFolder: The directory in which the list (1) is stored. The GUID is the identifier of the directory in the **Docs Table** (section 2.2.5.1).

tp_Template: The identifier of a file in the **Docs Table**. This file is a template used for creating new items in a list.

tp_ImageUrl: The image associated with the list type. This image is displayed on the Documents and Lists page.

tp_ReadSecurity: Special restrictions that can be placed on list item access. The following values are valid.

Value	Description
1	No special restrictions.
2	Users will see only their own list items.

tp_WriteSecurity: Special restrictions that can be placed on list item update. The following values are valid.

Value	Description
1	No special restrictions.
2	Users will see only their own list items.
4	Users will not update any list items in the list (1).

tp_AnonymousPermMask: The value that is used to restrict access to the list (1). Identifies the rights granted to anonymous users.

tp_Subscribed: The value that indicates whether **alerts** are specified for a list (1).

tp_Direction: The value indicating whether the list direction is left-to-right or right-to-left. The following values are valid.

Value	Description
0	No explicit direction is specified.
1	Text flow is left to right.
2	Text flow is right to left.

tp_Flags: A **List Flags** (section 2.2.2.4) value describing this list (1).

tp_ThumbnailSize: The option that specifies the size of thumbnail images.

tp_WebImageWidth: The option that specifies image width in the browser.

tp_WebImageHeight: The option that specifies image height in the browser.

tp_ItemCount: The number of items in the list (1).

tp_NextAvailableId: The next available identifier for an item that is added to a list (1).

tp_Description: The description of the list (1).

tp_EmailInsertsFolder: The public folder that is examined for email inserts into this document library.

tp_EmailInsertsLastSyncTime: The last time the list (1) was synchronized with the public folder. It is used to prevent reading the same email messages repeatedly.

tp_ACL: The binary serialization of the **WSS ACL Format** (section 2.2.3.2) in effect for the document.

tp_EventSinkAssembly: The managed code assembly that is used to handle document library **events (1)**.

tp_EventSinkClass: The class name inside the managed code assembly that handles document library events (1).

tp_EventSinkData: The **property bag** for use by the event handler.

tp_Fields: The fields in the list (1). This field can be NULL if the list has not been customized.

2.2.5.3 Sec_SiteGroupsView

The **Sec_SiteGroupsView** is a view into site group information with one site group per row. Site groups available through this view are owned by a user or domain group, or by a site group. If a user or domain group owns the site group, the **OwnerIsUser** bit is set to 1, and the **GroupID**, **GroupName**, **GroupDescription**, **GroupSiteID**, **GroupOwner**, and **GroupOwnerIsUser** fields are set to NULL.

If the site group is owned by a site group, the **OwnerIsUser** bit is set to 0, and the **UserID**, **UserSID**, **UserName**, **UserLogin**, **UserEmail**, **UserNotes**, **UserSiteAdmin**, and **UserDomainGroup** are set to NULL. The **Sec_SiteGroupsView** is defined using T-SQL syntax, as follows.

```
VIEW Sec_SiteGroupsView(
ID                                  int,
Title                               nvarchar(255),
Description                         nvarchar(512),
SiteWebId                           uniqueidentifier,
Owner                               int,
OwnerIsUser                         bit,
OwnerGlobal                         bit,
Type                                tinyint,
Global                              bit,
SiteID                              uniqueidentifier,
UserID                              int,
UserSID                             varbinary(512),
UserName                            nvarchar(255),
UserLogin                           nvarchar(255),
UserEmail                           nvarchar(255),
UserNotes                           nvarchar(1023),
UserSiteAdmin                       bit,
UserDomainGroup                     bit,
GroupID                             int,
GroupName                           nvarchar(255),
GroupDescription                    nvarchar(512),
GroupSiteID                         uniqueidentifier,
GroupOwner                          int,
GroupOwnerIsUser                    bit
);
```

ID: The identifier for the sitegroup. This parameter MUST NOT be NULL.

Title: The user-friendly display name for the site group. This parameter MUST NOT be NULL.

Description: The site group description.

SiteWebId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection that contains the site group information. This value is the same as the value returned in SiteId. For non-user-owned **groups**, this column name will be WebId. This parameter MUST NOT be NULL.

Owner: The **User Identifier** (section 2.2.1.24) or **Site Group Identifier** (section 2.2.1.18) of the site group owner. This parameter MUST NOT be NULL.

OwnerIsUser: A bit flag specifying whether the site group owner is a user or a sitegroup. When the value in the owner field is a **User Identifier** for a user or a domain group, the **OwnerIsUser** flag MUST be set to 1. When the value in the owner field is a **Site Group Identifier**, the **OwnerIsUser** flag MUST be set to 0. This parameter MUST NOT be NULL.

OwnerGlobal: A bit flag specifying whether the view contains ownership information for a user or domain group, or for a site group.

When the owner is a site group, the **OwnerGlobal** flag MUST be set to 1, and the **GroupID**, **GroupName**, **GroupDescription**, **GroupSiteID**, **GroupOwner**, and **GroupOwnerIsUser** fields MUST be populated with information from the site group that owns this site group, and the **UserID**, **UserSID**, **UserName**, **UserLogin**, **UserEmail**, **UserNotes**, **UserSiteAdmin**, and **UserDomainGroup** MUST be NULL.

When the owner is a user or domain group, the **OwnerGlobal** flag MUST be set to 0, the **UserID**, **UserSID**, **UserName**, **UserLogin**, **UserEmail**, **UserNotes**, **UserSiteAdmin**, and **UserDomainGroup** MUST be populated with information from the user or domain group that own this site group, and the **GroupID**, **GroupName**, **GroupDescription**, **GroupSiteID**, **GroupOwner**, and **GroupOwnerIsUser** MUST be NULL.

Type: This parameter MUST be zero.

Global: This parameter MUST be 1.

SiteID: The **Site Collection Identifier** (section 2.2.1.17) of the site collection. This parameter MUST NOT be NULL.

UserID: The **User Identifier** (section 2.2.1.24) of the **owner** of the sitegroup. This MUST be NULL when the site group owner is a site group.

UserSID: The **SystemID** (section 2.2.1.20) of the owner of the site group. This MUST be NULL when the site group owner is a site group.

UserName: The user-friendly name of the owner of the site group. This MUST be NULL when the site group owner is a site group.

UserLogin: The login name of the owner of the site group. This MUST be NULL when the site group owner is a site group.

UserEmail: The email address of the owner of the site group. This MUST be NULL when the site group owner is a site group.

UserNotes: The notes associated with the owner of the site group. This MUST be NULL when the site group owner is a site group.

UserSiteAdmin: A bit flag specifying whether the site group owner is a site administrator. When the site group owner is a site administrator, the **UserSiteAdmin** flag MUST be set to 1. If the user or domain group owner of the site group is not a site administrator, the **UserSiteAdmin** flag MUST be set to 0. This MUST be NULL when the site group owner is a site group.

UserDomainGroup: A bit flag specifying whether the site group owner is a domain group. When the site group owner is a domain group, the flag MUST be set to 1. When the sitegroup owner is a user, the flag MUST be set to 0. This flag MUST be NULL when the sitegroup owner is a sitegroup.

GroupID: The **Site Group Identifier** of the owner of this site group. This MUST be NULL when the site group owner is a user or domain group.

GroupName: The user-friendly name of the owner of this site group. This MUST be NULL when the site group owner is a user or domain group.

GroupDescription: The description of the owner of this site group. This MUST be NULL when the site group owner is a user or domain group.

GroupSiteID: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the owner of this site group. This MUST be NULL when the sitegroup owner is a user or domain group.

GroupOwner: The user identifier or site group identifier of the owner of the site group that owns this site group. This MUST be NULL when the site group owner is a user or domain group, because only an owner that is a site group can have a group owner.

GroupOwnerIsUser: A bit flag specifying whether the site group that owns this sitegroup is in turn owned by a user or domain group, or by a site group. When the owner of the site group that owns this site group is a user or domain group, the **GroupOwnerIsUser** flag MUST be set to 1. This flag MUST be set to 0 when the site group owner is owned by a site group. This MUST be NULL when the site group owner is a user or domain group.

2.2.5.4 Sec_WebGroupsView

The **Sec_WebGroupsView** is a view into site group information with one site group per row. Site groups available through this view are owned by a user or domain group, or by a site group. The **Sec_WebGroupsView** is defined using T-SQL syntax, as follows.

```
VIEW Sec WebGroupsView(
ID                                  int,
Title                               nvarchar(255),
Description                         nvarchar(512),
WebId                               uniqueidentifier,
Owner                               int,
OwnerIsUser                         bit,
OwnerGlobal                         bit,
Type                                tinyint,
Global                              bit,
SiteID                              uniqueidentifier,
UserID                              int,
UserSID                             int,
UserName                            int,
UserLogin                           int,
UserEmail                           int,
UserNotes                           int,
UserSiteAdmin                       int,
UserDomainGroup                     int,
GroupID                             int,
GroupName                           int,
GroupDescription                    int,
GroupSiteID                         int,
GroupOwner                          int,
GroupOwnerIsUser                    int
);
```

ID: The identifier for the site group. This parameter MUST NOT be NULL.

Title: The user-friendly display name for the site group. This parameter MUST NOT be NULL.

Description: The site group description.

WebId: The **Site Identifier** (section 2.2.1.19) of the site group. This parameter MUST NOT be NULL.

Owner: The **User Identifier** (section 2.2.1.24) or the **Site Collection Identifier** (section 2.2.1.17) of the site group owner. This parameter MUST be zero.

OwnerIsUser: A bit flag specifying whether the user is a site group owner or a site group **member**. This parameter MUST be zero.

OwnerGlobal: A bit flag specifying whether the view contains ownership information for a user or domain group, or for a site group. This parameter MUST be zero.

Type: An integer value that indicates the built-in site group template.

Global: This value MUST be zero.

SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection. This parameter MUST NOT be NULL.

UserID: This parameter MUST be NULL.

UserSID: This parameter MUST be NULL.

UserName: This parameter MUST be NULL.

UserLogin: This parameter MUST be NULL.

UserEmail: This parameter MUST be NULL.

UserNotes: This parameter MUST be NULL.

UserSiteAdmin: This parameter MUST be NULL.

GroupId: This parameter MUST be NULL.

GroupName: This parameter MUST be NULL.

GroupDescription: This parameter MUST be NULL.

GroupSiteId: This parameter MUST be NULL.

GroupOwner: This parameter MUST be NULL.

GroupOwnerIsUser: This parameter MUST be NULL.

2.2.5.5 Sites Table

The **Sites Table** stores information about site collections. The **Sites Table** is defined using T-SQL syntax, as follows.

```
TABLE Sites(
FullUrl             nvarchar(260)     NOT NULL,
Id                  uniqueidentifier  NOT NULL,
NextUserOrGroupId   int               NOT NULL,
OwnerID             int               NOT NULL,
SecondaryContactID  int               NULL,
Subscribed          bit               NOT NULL DEFAULT (0),
TimeCreated         datetime          NOT NULL,
UsersCount          int               NULL DEFAULT (1),
BWUsed              bigint            NULL DEFAULT (0),
DiskUsed            bigint            NULL DEFAULT (0),
QuotaTemplateID     smallint          NULL DEFAULT (0),
DiskQuota           bigint            NULL DEFAULT (0),
UserQuota           int               NULL DEFAULT (0),
DiskWarning         bigint            NULL DEFAULT (0),
DiskWarned          datetime          NULL,
BitFlags            int               NULL DEFAULT (0),
SecurityVersion     bigint            NULL DEFAULT (0),
CertificationDate   datetime          NULL,
DeadWebNotifyCount  smallint          NULL DEFAULT (0),
PortalURL           nvarchar(260)     NULL,
PortalName          nvarchar(255)     NULL,
LastContentChange   datetime          NOT NULL DEFAULT (getutcdate()),
LastSecurityChange  datetime          NOT NULL DEFAULT (getutcdate()),
HashKey             binary(16)        NULL,
EmailEnabled        bit               NULL DEFAULT (0)
```

```
);
```

FullUrl: The absolute URL of the site collection.

Id: The **Site Collection Identifier** (section 2.2.1.17) of the site collection. Used only during upgrade.

NextUserOrGroupId: An integer value that is incremented when a new user or site group is added to the site collection. This value indicates the value of the next user or **Site Group Identifier** (section 2.2.1.18) to be used.

OwnerID: The **User Identifier** (section 2.2.1.24) of the user who owns the site collection.

SecondaryContactID: The **User Identifier** of the user who is the secondary contact of the site collection.

Subscribed: A bit set to 1 to indicate that some part of the site collection has been subscribed for implementation-specific notifications.

TimeCreated: The date and time, in UTC format, when the site collection was created.

UsersCount: The number of users in the site collection.

BWUsed: The number of sites in the site collection that are actively used. This value serves an implementation-specific, usage-reporting **feature**.

DiskUsed: The size of disk space used to store content in the site collection, in bytes.

QuotaTemplateID: An identifier of a quota template used to set the disk quota for the site collection.

DiskQuota: The maximum size, in bytes, of disk space that can be allocated by the site collection. A value of 0 indicates that no limit is set.

UserQuota: The maximum number of users that the site collection can contain. A value of 0 indicates that no limit is set.

DiskWarning: The size, in bytes, of disk space that can be allocated on the site collection before the disk warning email is sent. A value of 0 indicates that no warning email will be sent.

DiskWarned: A date and time value, in UTC format, set to the last time a warning was sent to the site collection owner about disk usage, if any.

BitFlags: A **Site Collection Flags** (section 2.2.2.7) value describing the site collection.

SecurityVersion: A version number that is incremented when changes are made to the site collection's permissions.

CertificationDate: The date and time, in UTC format, when the site collection was last confirmed by its owner as being used.

DeadWebNotifyCount: The number of times that a notification was sent to the site collection owner that the site collection will be deleted if the owner does not certify it as being used.

PortalURL: The URL of an external website designated as the sites' portal. This is an implementation-specific navigation links feature.

PortalName: The name of the external website referenced in the PortalURL column. Used for display in implementation-specific navigation features.

LastContentChange: The date and time value, in UTC format, when the content of the site collection was last changed.

LastSecurityChange: The date and time value, in UTC format, when the permissions on the site collection were last changed.

HashKey: Binary data containing an algorithmic **hash** of the URL for a **site** and a lookup table that is used to route requests to the correct database server.

EmailEnabled: A value that indicates whether a document library is email enabled. Set to 1 if enabled; otherwise, set to 0. Email integration is an implementation-specific capability. This value is used to optimize performance.

2.2.5.6 UserData Table

The **UserData Table** stores the data that is displayed in the lists. The **UserData Table** is defined using T-SQL syntax, as follows.

```
TABLE UserData(
tp_Id                int               NOT NULL,
tp_ListId            uniqueidentifier  NOT NULL,
tp_SiteId            uniqueidentifier  NOT NULL,
tp_Version           int               NOT NULL,
tp_Author            int               NULL,
tp_Editor            int               NULL,
tp_Modified          datetime          NULL,
tp_Created           datetime          NULL,
tp_Ordering          varchar(512)      NULL,
tp_HasAttachment     bit               NOT NULL DEFAULT ((0)),
tp_ModerationStatus  int               NOT NULL DEFAULT ((0)),
tp_IsCurrent         bit               NOT NULL DEFAULT ((1)),
tp_ItemOrder         float             NULL,
tp_InstanceID        int               NULL,
tp_GUID              uniqueidentifier  NOT NULL DEFAULT (newid()),
tp_Size              int               NOT NULL DEFAULT ((0)),
nvarchar1            nvarchar(255)     NULL,
nvarchar2            nvarchar(255)     NULL,
nvarchar3            nvarchar(255)     NULL,
nvarchar4            nvarchar(255)     NULL,
nvarchar5            nvarchar(255)     NULL,
nvarchar6            nvarchar(255)     NULL,
nvarchar7            nvarchar(255)     NULL,
nvarchar8            nvarchar(255)     NULL,
nvarchar9            nvarchar(255)     NULL,
nvarchar10           nvarchar(255)     NULL,
nvarchar11           nvarchar(255)     NULL,
nvarchar12           nvarchar(255)     NULL,
nvarchar13           nvarchar(255)     NULL,
nvarchar14           nvarchar(255)     NULL,
nvarchar15           nvarchar(255)     NULL,
nvarchar16           nvarchar(255)     NULL,
nvarchar17           nvarchar(255)     NULL,
nvarchar18           nvarchar(255)     NULL,
nvarchar19           nvarchar(255)     NULL,
nvarchar20           nvarchar(255)     NULL,
nvarchar21           nvarchar(255)     NULL,
nvarchar22           nvarchar(255)     NULL,
nvarchar23           nvarchar(255)     NULL,
nvarchar24           nvarchar(255)     NULL,
nvarchar25           nvarchar(255)     NULL,
nvarchar26           nvarchar(255)     NULL,
nvarchar27           nvarchar(255)     NULL,
nvarchar28           nvarchar(255)     NULL,
nvarchar29           nvarchar(255)     NULL,
nvarchar30           nvarchar(255)     NULL,
nvarchar31           nvarchar(255)     NULL,
nvarchar32           nvarchar(255)     NULL,

nvarchar33           nvarchar(255)     NULL,
```

```
    nvarchar34          nvarchar(255)     NULL,
    nvarchar35          nvarchar(255)     NULL,
    nvarchar36          nvarchar(255)     NULL,
    nvarchar37          nvarchar(255)     NULL,
    nvarchar38          nvarchar(255)     NULL,
    nvarchar39          nvarchar(255)     NULL,
    nvarchar40          nvarchar(255)     NULL,
    nvarchar41          nvarchar(255)     NULL,
    nvarchar42          nvarchar(255)     NULL,
    nvarchar43          nvarchar(255)     NULL,
    nvarchar44          nvarchar(255)     NULL,
    nvarchar45          nvarchar(255)     NULL,
    nvarchar46          nvarchar(255)     NULL,
    nvarchar47          nvarchar(255)     NULL,
    nvarchar48          nvarchar(255)     NULL,
    nvarchar49          nvarchar(255)     NULL,
    nvarchar50          nvarchar(255)     NULL,
    nvarchar51          nvarchar(255)     NULL,
    nvarchar52          nvarchar(255)     NULL,
    nvarchar53          nvarchar(255)     NULL,
    nvarchar54          nvarchar(255)     NULL,
    nvarchar55          nvarchar(255)     NULL,
    nvarchar56          nvarchar(255)     NULL,
    nvarchar57          nvarchar(255)     NULL,
    nvarchar58          nvarchar(255)     NULL,
    nvarchar59          nvarchar(255)     NULL,
    nvarchar60          nvarchar(255)     NULL,
    nvarchar61          nvarchar(255)     NULL,
    nvarchar62          nvarchar(255)     NULL,
    nvarchar63          nvarchar(255)     NULL,
    nvarchar64          nvarchar(255)     NULL,
    int1                int               NULL,
    int2                int               NULL,
    int3                int               NULL,
    int4                int               NULL,
    int5                int               NULL,
    int6                int               NULL,
    int7                int               NULL,
    int8                int               NULL,
    int9                int               NULL,
    int10               int               NULL,
    int11               int               NULL,
    int12               int               NULL,
    int13               int               NULL,
    int14               int               NULL,
    int15               int               NULL,
    int16               int               NULL,
    float1              float             NULL,
    float2              float             NULL,
    float3              float             NULL,
    float4              float             NULL,
    float5              float             NULL,
    float6              float             NULL,
    float7              float             NULL,
    float8              float             NULL,
    float9              float             NULL,
    float10             float             NULL,
    float11             float             NULL,
    float12             float             NULL,
    float13             float             NULL,
    float14             float             NULL,
    float15             float             NULL,
    float16             float             NULL,
    float17             float             NULL,
    float18             float             NULL,
    float19             float             NULL,
    float20             float             NULL,
    float21             float             NULL,
```

```
    float22             float           NULL,
    float23             float           NULL,
    float24             float           NULL,
    float25             float           NULL,
    float26             float           NULL,
    float27             float           NULL,
    float28             float           NULL,
    float29             float           NULL,
    float30             float           NULL,
    float31             float           NULL,
    float32             float           NULL,
    datetime1           datetime        NULL,
    datetime2           datetime        NULL,
    datetime3           datetime        NULL,
    datetime4           datetime        NULL,
    datetime5           datetime        NULL,
    datetime6           datetime        NULL,
    datetime7           datetime        NULL,
    datetime8           datetime        NULL,
    datetime9           datetime        NULL,
    datetime10          datetime        NULL,
    datetime11          datetime        NULL,
    datetime12          datetime        NULL,
    datetime13          datetime        NULL,
    datetime14          datetime        NULL,
    datetime15          datetime        NULL,
    datetime16          datetime        NULL,
    bit1                bit             NULL,
    bit2                bit             NULL,
    bit3                bit             NULL,
    bit4                bit             NULL,
    bit5                bit             NULL,
    bit6                bit             NULL,
    bit7                bit             NULL,
    bit8                bit             NULL,
    bit9                bit             NULL,
    bit10               bit             NULL,
    bit11               bit             NULL,
    bit12               bit             NULL,
    bit13               bit             NULL,
    bit14               bit             NULL,
    bit15               bit             NULL,
    bit16               bit             NULL,
    uniqueidentifier1   uniqueidentifier NULL,
    ntext1              ntext           NULL,
    ntext2              ntext           NULL,
    ntext3              ntext           NULL,
    ntext4              ntext           NULL,
    ntext5              ntext           NULL,
    ntext6              ntext           NULL,
    ntext7              ntext           NULL,
    ntext8              ntext           NULL,
    ntext9              ntext           NULL,
    ntext10             ntext           NULL,
    ntext11             ntext           NULL,
    ntext12             ntext           NULL,
    ntext13             ntext           NULL,
    ntext14             ntext           NULL,
    ntext15             ntext           NULL,
    ntext16             ntext           NULL,
    ntext17             ntext           NULL,
    ntext18             ntext           NULL,
    ntext19             ntext           NULL,
    ntext20             ntext           NULL,
    ntext21             ntext           NULL,
    ntext22             ntext           NULL,
    ntext23             ntext           NULL,
    ntext24             ntext           NULL,
```

```
ntext25             ntext           NULL,
ntext26             ntext           NULL,
ntext27             ntext           NULL,
ntext28             ntext           NULL,
ntext29             ntext           NULL,
ntext30             ntext           NULL,
ntext31             ntext           NULL,
ntext32             ntext           NULL,
sql_variant1        sql_variant     NULL,
sql_variant2        sql_variant     NULL,
sql_variant3        sql_variant     NULL,
sql_variant4        sql_variant     NULL,
sql_variant5        sql_variant     NULL,
sql_variant6        sql_variant     NULL,
sql_variant7        sql_variant     NULL,
sql_variant8        sql_variant     NULL
);
```

tp_Id: The identifier for the list item, uniquely identifying it within the UserData table.

tp_ListId: The **List Identifier** (section 2.2.1.10) of the **list (1)** or document library containing the list item.

tp_SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the list item.

tp_Version: A value that indicates the version of the data. A counter that is incremented any time a change is made to the list item and used for internal conflict detection.

tp_Author: The **User Identifier** (section 2.2.1.24) for the user who created the list item.

tp_Editor: The **User Identifier** for the user who last edited the list item.

tp_Modified: A date and time value, in UTC format, specifying when this list item was last modified.

tp_Created: A date and time value, in UTC format, specifying when this list item was created.

tp_Ordering: A field that specifies the order for threading web discussion comments.

tp_HasAttachment: A value that specifies whether the item has an associated attachment. A bit that is set to 1 if the list item has an attachment associated with it. Otherwise, it is set to 0.

tp_ModerationStatus: A value that specifies the status of a list (1) if it is moderated (section 2.2.1.13). 0 is approved; 2 is pending; 1 is rejected.

tp_IsCurrent: A value indicating whether the item is the current item in the issue tracking list. A bit that is set to 1 if this is a current version of this list item. Otherwise, it is set to 0.

tp_ItemOrder: Sort criterion for an ordered list (1). A value used to calculate the relative order in which to view the list item when displayed with other list items from the same list (1).

tp_InstanceID: An identifier for an occurrence in a **Meeting Workspace site** that has a recurring **event (1)** or multiple single events (1) linked to it.

tp_GUID: A **List Item Identifier** (section 2.2.1.11) value uniquely identifying the list item.

tp_Size: The sum of the size, in bytes, of the content of application-schema columns in the list item. This does not include the size of the **stream** for list items that have an associated stream.

For the following columns, a suffix of' '#' indicates that they are duplicated a number of times. The number of times each column is duplicated varies and is indicated for each column in the description of that column.

nvarchar#: Columns for application-defined fields that hold values of type nvarchar. The 64 columns are named nvarchar1 to nvarchar64. If the column does not contain data, this value MUST be NULL.

int#: Columns for application-defined fields that hold values of type int. The 16 columns are named int1 to int16. If the column does not contain data, this value MUST be NULL.

float#: Columns for application-defined fields that hold values of type float. The 32 columns are named float1 to float32. If the column does not contain data, this value MUST be NULL.

datetime#: Columns for application-defined fields that hold values of type datetime. The 16 columns are named datetime1 to datetime16. If the column does not contain data, this value MUST be NULL.

bit#: Columns for application-defined fields that hold values of type bit. The 16 columns are named bit1 to bit16. If the column does not contain data, this value MUST be NULL.

uniqueidentifier1: A column for an application-defined field of type uniqueidentifier. If the column does not contain data, this value MUST be NULL.

ntext#: Columns for application-defined fields that hold values of type ntext. The 32 columns are named ntext1 to ntext32. If the column does not contain data, this value MUST be NULL.

sql_variant#: Columns for application-defined fields that hold values of type **sql_variant**. The eight columns are named sql_variant1 to sql_variant8. If the column does not contain data, this value MUST be NULL.

2.2.5.7 UserInfo Table

The **UserInfo table** stores information about each user that is added to the **site**. The **UserInfo table** is defined using T-SQL syntax, as follows.

```
TABLE UserInfo(
tp_SiteID       uniqueidentifier  NOT NULL,
tp_ID           int               NOT NULL,
tp_DomainGroup  bit               NOT NULL,
tp_GUID         uniqueidentifier  NOT NULL DEFAULT (newid()),
tp_SystemID     tSystemID         NOT NULL,
tp_Deleted      int               NOT NULL,
tp_SiteAdmin    bit               NOT NULL,
tp_Login        nvarchar (255)    NOT NULL,
tp_Title        nvarchar (255)    NOT NULL,
tp_Email        nvarchar (255)    NOT NULL,
tp_Notes        nvarchar (1023)   NOT NULL
);
```

tp_SiteID: The **Site Collection Identifier** (section 2.2.1.17) of the site collection that is associated with the **principal (1)**.

tp_ID: The **User Identifier** (section 2.2.1.24) of the principal (1). This value MUST be unique.

tp_DomainGroup: A bit flag that specifies whether the account is a domain group account. A bit set to 1 if the principal (1) is a domain group. Otherwise, the bit is set to 0, specifying that the principal (1) is a user.

tp_GUID: A GUID for indexing with the UserInfo table.

tp_SystemID: The **SystemID** (section 2.2.1.20) of the principal (1).

tp_Deleted: A value specifying whether the principal (1) is marked as deleted. If **tp_Deleted** is 0, the principal (1) is not deleted. If **tp_Deleted** is not 0, the principal (1) is marked as deleted, and the value is the **User Identifier** that was associated with this principal.

tp_SiteAdmin: A bit set to 1 if the principal (1) is a site administrator for the site collection.

tp_Login: The login name for the principal (1).

tp_Title: The display name for the principal (1).

tp_Email: The email address for the principal (1).

tp_Notes: A descriptive text about the principal.

2.2.6 XML Structures

The syntax of the definitions in this section uses XML Schema as defined in [XMLSCHEMA1] and [XMLSCHEMA2].

2.2.6.1 Namespaces

None.

2.2.6.2 Simple Types

2.2.6.2.1 FALSE_Case_Insensitive_Else_Anything

The **FALSE_Case_Insensitive_Else_Anything** type is used to specify a **Boolean** value.

```
<xs:simpleType name="FALSE_Case_Insensitive_Else_Anything">
<xs:restriction base="xs:string">
<xs:pattern value="[Ff][Aa][Ll][Ss][Ee]|.*" />
</xs:restriction>
</xs:simpleType>
```

2.2.6.2.2 FieldAggregationAttribute

The **FieldAggregationAttribute** type is used to specify how values are promoted from an XML or site template document.

```
<xs:simpleType name="FieldAggregationAttribute">
<xs:restriction base="xs:string">
<xs:enumeration value="average" />
<xs:enumeration value="count" />
<xs:enumeration value="first" />
<xs:enumeration value="last" />
<xs:enumeration value="max" />
<xs:enumeration value="merge" />
<xs:enumeration value="min" />
<xs:enumeration value="sum" />
</xs:restriction>
</xs:simpleType>
```

The meanings of the values are specified in the following table. In the table, a matching node designates an element or an attribute that matches a specified **XML Path Language (XPath)** query. The value of an element node is the concatenation of all character data directly contained by the element. The meanings of the values are specified in the following table.

Value	Meaning
average	The average of the matching nodes, whose values can be interpreted as a floating point numbers. This

Value	Meaning
	is an **empty string** if no values match.
count	Count of the matching nodes.
first	The value of the first matching node.
last	The value of the last matching node.
max	The value of the largest matching node, interpreted as a floating point number.
merge	The value of the nodes, in order, delimited by a newline character.
min	The value of the smallest matching node, interpreted as a floating point number.
sum	The sum of the nodes that match and whose values can be interpreted as floating point numbers.

2.2.6.2.3 FieldInternalType

The **FieldInternalType** type is used to specify a field internal type.

Note: In the following XML, the value attribute is broken into multiple lines only for readability.

```
<xs:simpleType name="FieldInternalType">
<xs:restriction base="xs:string">
<xs:pattern value="[iI][nN][tT][eE][gG][eE][rR]|[tT][eE][xX]
[tT]|[nN][oO][tT][eE]|[dD][aA][tT][eE][tT][iI][mM][eE]|[cC][oO][uU][nN]
[tT][eE][rR]|[cC][hH][oO][iI][cC][eE]|[lL][oO][oO][kK][uU][pP]|[bB][oO]
[oO][lL][eE][aA][nN]|[nN][uU][mM][bB][eE][rR]|[cC][uU][rR][rR][eE][nN]
[cC][yY]|[uU][rR][lL]|[cC][oO][mM][pP][uU][tT][eE][dD]|[tT][hH][rR][eE]
[aA][dD][iI][nN][gG]|[gG][uU][iI][dD]|[mM][uU][lL][tT][iI][cC][hH][oO]
[iI][cC][eE]|[gG][rR][iI][dD][cC][hH][oO][iI][cC][eE]|[cC][aA][lL][cC]
[uU][lL][aA][tT][eE][dD]|[fF][iI][lL][eE]|[aA][tT][tT][aA][cC][hH][mM]
[eE][nN][tT][sS]|[uU][sS][eE][rR]|[rR][eE][cC][uU][rR][rR][eE][nN][cC]
[eE]|[cC][rR][oO][sS][sS][pP][rR][oO][jJ][eE][cC][tT][lL][iI][nN][kK]|
[mM][oO][dD][sS][tT][aA][tT]|[eE][rR][rR][oO][rR]" />
</xs:restriction>
</xs:simpleType>
```

While the XSD pattern is normative, the values are:

- **Integer**
- **Text**
- **Note**
- **DateTime**
- **Counter**
- **Choice**
- **Lookup**
- **Boolean**
- **Number**
- **Url**

- **Computed**
- **Threading**
- **GUID**
- **MultiChoice**
- **GridChoice**
- **Calculated**
- **File**
- **Attachments**
- **User**
- **Recurrence**
- **CrossProjectLink**
- **ModStat**
- **Error**

2.2.6.2.4 FieldRefType

The **FieldRefType** type is used to specify the type of relationship given by a field reference definition within the field.

```
<xs:simpleType name="FieldRefType">
<xs:restriction base="xs:string">
<xs:enumeration value="Recurrence" />
<xs:enumeration value="EventType" />
<xs:enumeration value="UID" />
<xs:enumeration value="RecurrenceId" />
<xs:enumeration value="EventCancel" />
<xs:enumeration value="StartDate" />
<xs:enumeration value="EndDate" />
<xs:enumeration value="RecurData" />
<xs:enumeration value="Duration" />
<xs:enumeration value="TimeZone" />
<xs:enumeration value="XMLTZone" />
<xs:enumeration value="CPLink" />
<xs:enumeration value="LinkURL" />
<xs:enumeration value="MasterSeriesItemID" />
</xs:restriction>
</xs:simpleType>
```

The meanings of the values are specified in the following table.

Value	Meaning
Recurrence	Specifies a recurring event.
EventType	Specifies an event type.
UID	Specifies a **unique identifier (UID)** of the event.
RecurrenceId	Specifies a recurrence Id for the event.

Value	Meaning
EventCancel	Specifies a cancelled event.
StartDate	Specifies a start date for the event.
EndDate	Specifies an end date for the event.
RecurData	Specifies recurrence data for the event.
Duration	Specifies event duration.
TimeZone	Specifies a time zone for the event.
XMLTZone	Specifies a time zone in XML format.
CPLink	Specifies a cross program link.
LinkURL	Specifies a link **Uniform Resource Locator (URL)** for the event.
MasterSeriesItemID	Specified the ID of the master recurrence record.

2.2.6.2.5 IMEMode

The **IMEMode** type is used to specify a bias for an **Input Method Editor (IME)**.

```
<xs:simpleType name="IMEMode">
<xs:restriction base="xs:string">
<xs:enumeration value="auto" />
<xs:enumeration value="active" />
<xs:enumeration value="inactive" />
<xs:enumeration value="disabled" />
</xs:restriction>
</xs:simpleType>
```

The meanings of the values are specified in the following table.

Value	Meaning
auto	Default. The IME is not affected. This value has the same effect as not specifying the ime-mode attribute (as described in [CSS3UI]).
active	All characters are entered through the IME. Users can still deactivate the IME.
inactive	All characters are entered without the IME. Users can still activate the IME.
disabled	The IME is completely disabled. Users cannot activate the IME if the control has focus.

2.2.6.2.6 IntPositive

The **IntPositive** type is used to specify a positive integer.

```
<xs:simpleType name="IntPositive">
<xs:restriction base="xs:int">
```

```
<xs:minInclusive value="1" />
</xs:restriction>
</xs:simpleType>
```

2.2.6.2.7 JoinType

The **JoinType** type specifies how to handle lookup fields for which the target of the lookup does not exist.

```
<xs:simpleType name="JoinType">
<xs:restriction base="xs:string">
<xs:enumeration value="INNER" />
<xs:enumeration value="LEFT OUTER" />
</xs:restriction>
</xs:simpleType>
```

The meanings of the values are specified in the following table.

Value	Meaning
INNER	Items whose target does not exist are omitted.
LEFT OUTER	Items whose target does not exist are included.

2.2.6.2.8 TextDirection

The **TextDirection** type is used to specify the preferred direction for displaying text.

```
<xs:simpleType name="TextDirection">
<xs:restriction base="xs:string">
<xs:enumeration value="none" /><xs:enumeration value="LTR" />
<xs:enumeration value="RTL" />
</xs:restriction>
</xs:simpleType>
```

The meanings of the values are specified in the following table.

Value	Meaning
ltr	Specifies a left-to-right text direction.
rtl	Specifies a right-to-left text direction.
none	Specifies no hint for a text direction, and that the direction will follow the context of the page.

2.2.6.2.9 TRUEFALSE

The **TRUEFALSE** type is used to specify a Boolean value.

```
<xs:simpleType name="TRUEFALSE">
<xs:restriction base="xs:string">
<xs:pattern value="[Tt][Rr][Uu][Ee]|[Ff][Aa][Ll][Ss][Ee]"/>
```

```
</xs:restriction>
</xs:simpleType>
```

2.2.6.2.10 UniqueIdentifierWithOrWithoutBraces

The **UniqueIdentifierWithOrWithoutBraces** type is used to specify a GUID.

```
<xs:simpleType name="UniqueIdentifierWithOrWithoutBraces">
<xs:restriction base="xs:string">
<xs:pattern value="\{?[0-9a-fA-F]{8}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{12}\}?"/>
</xs:restriction>
</xs:simpleType>
```

2.2.6.3 Complex Types

2.2.6.3.1 CHOICEDEFINITION Type

The **CHOICEDEFINITION** type contains a collection of choices for a **Choice** or **MultiChoice** field.

2.2.6.3.1.1 Schema

```
<xs:complexType name="CHOICEDEFINITION" mixed="true">
</xs:complexType>
```

2.2.6.3.1.2 Attributes

None

2.2.6.3.1.3 Child Elements

Content: Specifies values for choices in a **Choice** or **MultiChoice** field.

2.2.6.3.2 CHOICEDEFINITIONS Type

The **CHOICEDEFINITIONS** type contains a collection of choices for a **Choice** or **MultiChoice** field.

2.2.6.3.2.1 Schema

```
<xs:complexType name="CHOICEDEFINITIONS">
<xs:sequence>
<xs:element name="CHOICE" type="CHOICEDEFINITION" minOccurs="0" maxOccurs="unbounded" />
</xs:sequence>
</xs:complexType>
```

2.2.6.3.2.2 Attributes

None.

2.2.6.3.2.3 Child Elements

CHOICE: Specifies a choice in a **Choice** or **MultiChoice** field.

2.2.6.3.3 FieldDefinition Type

A **FieldDefinition** describes the structure and format of a field that is used within a **list (1)** or **content type**.

2.2.6.3.3.1 Schema

```
<xs:complexType name="FieldDefinition" mixed="true">
<xs:all>
<xs:element name="CHOICES" type="CHOICEDEFINITIONS" minOccurs="0" maxOccurs="1" />
<xs:element name="Customization" minOccurs="0" maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:any minOccurs="1" maxOccurs="1" namespace="##any" processContents="skip" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Default" type="xs:string" minOccurs="0" maxOccurs="1" />
<xs:element name="DefaultFormula" type="xs:string" minOccurs="0" maxOccurs="1" />
<xs:element name="DisplayBidiPattern" minOccurs="0" maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:any minOccurs="0" maxOccurs="unbounded" namespace="##any" processContents="skip" />
</xs:sequence>
<xs:anyAttribute processContents="skip" />
</xs:complexType>
</xs:element>
<xs:element name="DisplayPattern" type="CamlViewRoot" minOccurs="0" maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:any minOccurs="0" maxOccurs="unbounded" namespace="##any" processContents="skip" />
</xs:sequence>
<xs:anyAttribute processContents="skip" />
</xs:complexType>
</xs:element>
<xs:element name="FieldRefs" minOccurs="0" maxOccurs="1">
<xs:complexType mixed="true">
<xs:sequence>
<xs:element name="FieldRef" type="FieldRefDefinitionField" minOccurs="0"
maxOccurs="unbounded" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Formula" type="xs:string" minOccurs="0" maxOccurs="1" />
<xs:element name="MAPPINGS" type="MAPPINGDEFINITIONS" minOccurs="0" maxOccurs="1" />
</xs:all>

<xs:attribute name="AllowHyperlink" type="TRUEFALSE" default="FALSE" />
<xs:attribute name="AllowMultiVote" type="TRUEFALSE" default="FALSE" />
<xs:attribute name="AuthoringInfo" type="xs:string" default=""/>
<xs:attribute name="BaseType" type="FieldInternalType" default="Text" />
<xs:attribute name="BooleanFalse" type="TRUEFALSE" />
<xs:attribute name="BooleanTrue" type="TRUEFALSE" />
<xs:attribute name="CalType" type="xs:int" />
<xs:attribute name="CanToggleHidden" type="TRUEFALSE" default="FALSE" />
<xs:attribute name="ColName" type="xs:string" />
<xs:attribute name="ColName2" type="xs:string" />
<xs:attribute name="Default" type="xs:string" />
<xs:attribute name="Description" type="xs:string" />

<xs:attribute name="DisplayImage" type="xs:string" />
<xs:attribute name="DisplayName" type="xs:string" />
<xs:attribute name="DisplayNameSrcField" type="xs:string" />
<xs:attribute name="Div" type="xs:int" />

<xs:attribute name="Explicit" type="TRUEFALSE" />
<xs:attribute name="FieldRef" type="xs:string" />
<xs:attribute name="FillInChoice" type="TRUEFALSE" default="FALSE" />
<xs:attribute name="Filterable" type="TRUEFALSE" default="TRUE" />
<xs:attribute name="FilterableNoRecurrence" type="TRUEFALSE" default="FALSE" />
```

```
<xs:attribute name="ForcedDisplay" type="xs:string" />
<xs:attribute name="Format" type="xs:string" />
<xs:attribute name="FromBaseType" type="TRUEFALSE" default="FALSE" />
<xs:attribute name="GridEndNum" type="xs:int" />
<xs:attribute name="GridNATxt" type="xs:string" default="xs:string" />
<xs:attribute name="GridStartNum" type="IntPositive" />
<xs:attribute name="GridTxtRng1" type="xs:string" default=" xs:string" />
<xs:attribute name="GridTxtRng2" type="xs:string" default=" xs:string" />
<xs:attribute name="GridTxtRng3" type="xs:string" default=" xs:string" />
<xs:attribute name="Hidden" type="TRUEFALSE" default="FALSE" />

<xs:attribute name="ID" type="UniqueIdentifierWithOrWithoutBraces" />
<xs:attribute name="IMEMode" type="IMEMode " />

<xs:attribute name="JoinColName" type="xs:string" default="tp_ID" />
<xs:attribute name="JoinType" type="JoinType" default="LEFT OUTER" />
<xs:attribute name="Key" type="xs:string" />
<xs:attribute name="List" type="xs:string" />
<xs:attribute name="Max" type="xs:int" />
<xs:attribute name="maxLength" type="xs:int" />
<xs:attribute name="Min" type="xs:int" />
<xs:attribute name="Name" type="xs:string" />
<xs:attribute name="node" type="xs:string" />
<xs:attribute name="NullString" type="xs:string " />
<xs:attribute name="NumLines" type="xs:int" default="" />

<xs:attribute name="Percentage" type="TRUEFALSE" default="FALSE" />
<xs:attribute name="Presence" type="TRUEFALSE" />

<xs:attribute name="ReadOnly" type="TRUEFALSE" default="FALSE" />
<xs:attribute name="RefType" type="xs:string" />
<xs:attribute name="RenderXMLUsingPattern" type="TRUEFALSE" default="FALSE" />
<xs:attribute name="Required" type="TRUEFALSE" default="FALSE" />
<xs:attribute name="RichText" type="TRUEFALSE" default="FALSE" />


<xs:attribute name="Sealed" type="TRUEFALSE" default="FALSE" />

<xs:attribute name="ShowAddressBookButton" type="TRUEFALSE" default="FALSE" />
<xs:attribute name="ShowField" type="xs:string" />
<xs:attribute name="ShowInEditForm" type="TRUEFALSE" default="TRUE" />
<xs:attribute name="ShowInFileDlg" type="TRUEFALSE" />
<xs:attribute name="ShowInNewForm" type="TRUEFALSE" default="TRUE" />
<xs:attribute name="Sortable" type="TRUEFALSE" />
<xs:attribute name="StorageTZ" type="TRUEFALSE" />
<xs:attribute name="StripWS" type="TRUEFALSE " />
<xs:attribute name="SuppressNameDisplay" type="TRUEFALSE" />

<xs:attribute name="Type" type="xs:string" use="required"/>
<xs:attribute name="XName" type="xs:string" />
<xs:anyAttribute namespace="##other" processContents="lax" />
</xs:complexType>
```

2.2.6.3.3.2 Attributes

AllowHyperlink: For fields whose **Type** maps to the **Note** field internal type, and for which **RichText** is **TRUE**, a reader can use this attribute to determine if the editing UI allows insertion of hyperlinks.

AllowMultiVote: If **TRUE**, then multiple responses are allowed in a survey.

AuthoringInfo: Text describing the field to schema authors. A client or server presenting a schema-editing UI can display the text in this attribute.

BaseType: Allows a schema **author** to override the internal storage format for choice fields. A reader MUST ignore this attribute unless the field's internal type is **Choice**, **MultiChoice**, or **GridChoice**. A writer MUST NOT set this attribute's value to **Choice**, **MultiChoice**, or **GridChoice**.

BooleanFalse: An attribute indicating a false value.

BooleanTrue: An attribute indicating a true value.

CalType: A reader SHOULD take this value into consideration when rendering the field or performing date calculations on it. If that is also not present, it SHOULD fall back to the user's preference.

CanToggleHidden: If **TRUE**, a server MUST permit the user to change the value of the **Hidden** attribute. Otherwise, a server MUST prevent the **Hidden** attribute from changing.

ColName: A server SHOULD use this attribute to describe the physical storage location for the field.

ColName2: Same meaning as the **ColName** attribute, except that it describes a secondary physical storage location.

Default: Specifies the default description that is displayed for the **list (1)** in the user interface (UI).

Description: A textual description of the field to be displayed in the user interface for schema authoring.

DisplayImage: When specified, this attribute supplies the URL, relative to the server's /_layouts/images folder, to render when displaying the field. A reader MUST ignore this attribute except for **CrossProjectLink** fields and **Recurrence** fields. If not specified, a blank placeholder image is used instead.

DisplayName: The text to display in the user interface when referring to the field.

DisplayNameSrcField: When present, this attribute specifies the name of another field on the list (1) to which the **DisplayName** of the current field is synchronized.

Div: In fields whose field internal type is **Integer**, **Number**, or **Calculated**, a reader MUST interpret this as a floating point number by which the value is divided at render time. In fields of other internal types, a reader MUST ignore this attribute and a writer MUST NOT include this attribute.

Explicit: If set to **TRUE**, it is used to hide the recurrence and force it into the view.

FieldRef: When present on a **Lookup** field, specifies the name of another field on the list (1) from which to obtain the local value for the lookup. **Lookup** fields for which this attribute is not specified will supply their own storage for the local value.

FillInChoice: Specifies whether or not a **form** generated to let the user edit a choice field or multichoice field allows values other than those listed in the **CHOICES** child element. A reader MUST ignore this attribute except for the **Choice** fields and **MultiChoice** fields.

Filterable: Specifies whether items can be omitted or included from the results of a query based on the value of the field. TRUE if the field can be filtered; FALSE otherwise.

FilterableNoRecurrence: When **TRUE**, and the **Filterable** attribute is **FALSE**, this attribute specifies that items can be included or omitted from the results of a query in views that do not expand recurring **events (1)**.

ForcedDisplay: If present, specifies a value for the field to display in place of the field's real value.

Format: An optional attribute that specifies formatting for columns. The option can be set to **DateTime**, **DateOnly**,**TimeOnly**,**Telephone**, **ISO8601**, **ISO8601Basic**, **ISO8601Gregorian**, **DropDown**, **RadioButtons**, or **EventList**.

FromBaseType: If **TRUE**, this attribute indicates that a field is derived from base type.

GridEndNum: Specifies the largest choice possible in a **GridChoice** field.

GridNATxt: A textual value to render for the choice that indicates "not applicable" in a **GridChoice** field.

GridStartNum: Specifies the smallest choice possible in a **GridChoice** field.

GridTxtRng1: A textual value to render for the choice that indicates the value corresponding to that specified by the **GridStartNum** attribute in a **GridChoice** field.

GridTxtRng2: A textual value to render for the choice that indicates the middle value in a **GridChoice** field.

GridTxtRng3: A textual value to render for the choice that indicates the value corresponding to that specified by the **GridEndNum** attribute in a **GridChoice** field.

Hidden: If **TRUE**, this attribute indicates to render a field in views or forms.

ID: The GUID for the field.

IMEMode: If specified, indicates a value for the ime-mode attribute to be applied to an <input> tag when reading the field.

JoinColName: Specifies the physical storage location in the list (1) referred to by the **List** attribute to compare with the local value of a lookup field.

JoinType: Specifies how to treat items that have no corresponding matching item in the list (1) indicated by the **List** attribute. A reader MUST ignore this attribute except for **Lookup** fields.

Key: A string that stores unique value for identification of specified elements.

List: Specifies the foreign list for a **Lookup** field. A writer MUST include this attribute for **Lookup** fields. A reader MUST ignore this attribute for other kinds of fields.

Max: Specifies the maximum value for a **Number** field. A reader MUST ignore this attribute for other kinds of fields.

maxLength: Specifies the maximum number of characters allowed in a **Text** field.

Min: Specifies the minimum value for a **Number** field. A reader MUST ignore this attribute for other kinds of fields.

Name: A string that identifies the field within its list (1).

node: When present, this attribute specifies an XPath to be used to read or write the value of the field into an XML document.

NullString: Specifies the text that is displayed to represent an empty item.

NumLines: The number of lines to render when accepting input for a **Note** field. A reader MUST ignore this attribute for other fields.

Percentage: When specified on a **Number** field or a Calculated field that evaluates to a number, specifies that the field is rendered as a percentage. A reader MUST ignore this attribute for other fields.

Presence: Specifies whether a **User** field will be decorated with instant messaging presence information. A reader MUST ignore this attribute for fields that are not user fields.

ReadOnly: Determines whether or not the user is allowed to change the field through the user interface. If the value is **TRUE**, only programmatic changes are allowed.

RefType: Defines various reference types for **Event** identification, **Recurrence** pattern field and **CrossProjectLink** fields.

RenderXMLUsingPattern: Specifies whether or not a **Computed** field is rendered. A reader MUST ignore this attribute except for **Computed** fields.

Required: Specifies whether or not forms presented to accept data for the list item permit blank values for the field.

RichText: If **TRUE**, the attribute specifies that the **Note** field contains formatted text. A reader MUST ignore this attribute except for **Note** fields.

Sealed: Specifies how the server allows the field to be changed. If **TRUE**, the server MUST prevent changes to the field except for **DisplayName**, **Description**, and **Hidden**.

ShowAddressBookButton: Determines whether or not a field's edit control includes a facility for entering users from an address book. A reader MUST ignore this attribute except when rendering the field in the context of a form that supports this functionality.

ShowField: Specifies the field in the foreign list (1) that supplies the value of a **Lookup** field. A reader MUST ignore this attribute unless the field is a **Lookup** field.

ShowInEditForm: When **FALSE**, indicates that the field is not included in the form that is used to modify an item. When **TRUE**, indicates that the field's inclusion in such a form depends on implementation.

ShowInFileDlg: Valid only for fields within document library schemas. If **FALSE**, the field does not show up in the property dialog box for saving forms that appears when saving from client applications. For example, the **Title** field has this attribute because this is set directly in the document being saved to the document library.

ShowInNewForm: Similar to **ShowInEditForm**, except that it applies to the field's inclusion in a form designed to collect information about an **item** that is being created.

Sortable: When **FALSE**, this attribute specifies that query results are not allowed to be ordered with respect to this field.

StorageTZ: For **DateTime** fields, specifies the time zone in which the field is stored. If **TRUE**, UTC is indicated. Otherwise, the site's local time zone is indicated. The reader MUST ignore this attribute unless the field's type is **DateTime**.

StripWS: TRUE if white space is removed from the beginning and end of field values.

SuppressNameDisplay: TRUE to not display the name of the user in a **User** field.

Type: Specifies the rendering properties and internal type of the field.

Url: Contains the absolute URL for the **site**.

XName: A string that is used to correlate the field in an external schema.

2.2.6.3.3.3 Child Elements

Customization: Provides an arbitrary XML document to provide vendor extensibility.

CHOICES: A set of **CHOICE** string elements that represent a list of available choices for a field. The reader MUST ignore **CHOICES** if the **Type** attribute is not **Choice** or **MultiChoice**. The writer SHOULD omit **CHOICES** if the **Type** attribute is not **Choice** or **MultiChoice**.<1>

Default: The default value of instances of data for this field in a list item. The reader MUST ignore this element when the **DefaultFormula** element is present and not empty.

DefaultFormula: A formula used to calculate the default value for this field in a list item.

DisplayBidiPattern: The implementation-specific XML that specifies the rendering of a **Computed** field to be used for a **site** with a **locale** identifier that specifies a bidirectional read order. The reader MUST ignore **DisplayBidiPattern** if the **Type** attribute is **not** Computed. The writer SHOULD omit **DisplayBidiPattern** if the **Type** attribute is not **Computed**. <2>

DisplayPattern: The implementation-specific XML that specifies the rendering of a **Computed** field. A reader MUST ignore **DisplayPattern** if the **Type** attribute is not **Computed**. The writer SHOULD omit **DisplayPattern** if the **Type** attribute is not **Computed**.<3>

FieldRefs: Specifies the fields required to render **DisplayPattern** or **DisplayBidiPattern**. **DisplayPattern** and **DisplayBidiPattern** MUST NOT use fields that are not listed in this element.

MAPPINGS: A set of **MAPPING** string elements that represents a canonical, language-agnostic identifier for a corresponding **CHOICE** with the value specified by the **MAPPING** element. The reader MUST ignore **MAPPINGS** if the **Type** attribute is not **Choice** or **MultiChoice**. The writer SHOULD omit **MAPPINGS** if the **Type** attribute is not **Choice** or **MultiChoice**.<4>

2.2.6.3.4 FieldDefinitionDatabase Type

The **FieldDefinitionDatabase** type provides an overall container for **field definitions** in a Content Database.

2.2.6.3.4.1 Schema

```
<xs:complexType name="FieldDefinitionDatabase">
<xs:sequence>
<xs:choice>
<xs:element name="FieldRef" type="FieldRefDefinitionTP" />
<xs:element name="Field" type="FieldDefinitionTP" />
</xs:choice>
</xs:sequence>
</xs:complexType>
```

2.2.6.3.4.2 Attributes

None.

2.2.6.3.4.3 Child Elements

FieldRef: Specifies a reference to an existing field definition that is used in the current **list (1)**.

Field: Specifies either a definition of a new field to be used in this list (1), or a reference to an existing **field** that was deleted.

2.2.6.3.5 FieldDefinitionDatabaseWithVersion Type

The **FieldDefinitionDatabaseWithVersion Type** provides an overall container for field definitions in **tp_Fields**, as specified in the **Fields Information Result Set**.

2.2.6.3.5.1 Schema

```
<xs:complexType name="FieldDefinitionDatabaseWithVersion" mixed="true">
<xs:all>
<xs:element name="tp_Fields" type="FieldDefinitionDatabase" minOccurs="1" maxOccurs="1"/>
</xs:all>
</xs:complexType>
```

2.2.6.3.5.2 Attributes

None.

2.2.6.3.5.3 Child Elements

See **FieldDefinitionDatabase** (section 2.2.6.3.4).

2.2.6.3.6 FieldDefinitionTP Type

The **FieldDefinitionTP** type specifies a field and its associated components. **FieldDefinitionTP** has the same structure as **FieldDefinition**. However, if only the **ID** attribute of **FieldDefinitionTP** is specified, the element specifies a field definition that was deleted by a user on a front-end Web server.

2.2.6.3.6.1 Schema

```
<xs:complexType name="FieldDefinitionTP">
<xs:complexContent>
<xs:extension base="FieldDefinition">
<xs:attribute name="Type" type="xs:string" use="optional" />
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

2.2.6.3.6.2 Attributes

See **FieldDefinition** (section 2.2.6.3.3).

2.2.6.3.6.3 Child Elements

See **FieldDefinition** (section 2.2.6.3.3).

2.2.6.3.7 FieldRefDefinitionField Type

The **FieldRefDefinitionField** type specifies field definitions that are referenced within another field definition.

2.2.6.3.7.1 Schema

```
<xs:complexType name="FieldRefDefinitionField" mixed="true" >
<xs:attribute name="Name" type="xs:string" use="required" />
<xs:attribute name="ID" type="UniqueIdentifierWithOrWithoutBraces" use="optional" />
<xs:attribute name="ShowField" type="xs:string" use="optional" />
<xs:attribute name="RefType" type="FieldRefType" use="optional" />
<xs:attribute name="CreateURL" type="xs:string" use="optional" />
<xs:attribute name="Key" type="xs:string" use="optional" />
<xs:attribute name="DisplayName" type="xs:string" use="optional" />
</xs:complexType>
```

2.2.6.3.7.2 Attributes

Name: Specifies the **Name** attribute of the referenced field.

ID: Specifies the **ID** attribute of the referenced fields. When both **ID** and **Name** are specified, the reader MUST use **ID** first and fall back to **Name** if the **ID** does not match.

ShowField: Specifies an alternate value for the **ShowField** attribute on a lookup field when rendering in the context of a computed field.

RefType: Describes the type of reference of the field in an **events (1)** list. This MUST be a **FieldRefType**. In other cases, this attribute MUST NOT be present.

CreateURL: The URL to create a Meeting Workspace site. If the **RefType** is **LinkURL**, this attribute MUST be present. Otherwise, it MUST NOT be present.

Key: If the value of this attribute is set to "Primary", the server MUST give this field priority in the ordering of the items.

DisplayName: The reader MUST ignore this attribute.

2.2.6.3.7.3 Child Elements

Content: Describes the Meeting Workspace site created by the URL in **CreateURL**. If the **RefType** attribute is present and has the value **LinkURL**, then the element MUST have content. In other cases, the reader MUST ignore any content.

2.2.6.3.8 FieldRefDefinitionTP Type

The **FieldRefDefinitionTP** type specifies a reference to a field definition. The attributes specified here override existing values in the field definition.

2.2.6.3.8.1 Schema

```
<xs:complexType name="FieldRefDefinitionTP">
<xs:attribute name="Name" type="xs:string" use="required" />
<xs:attribute name="ColName" type="xs:string" use="optional" />
<xs:attribute name="ColName2" type="xs:string" use="optional" />
<xs:attribute name="ID" type="UniqueIdentifierWithOrWithoutBraces" use="optional" />
</xs:complexType>
```

2.2.6.3.8.2 Attributes

See **FieldDefinition** (section 2.2.6.3.3).

2.2.6.3.8.3 Child Elements

None.

2.2.6.3.9 MAPPINGDEFINITION Type

The **MAPPINGDEFINITION** type defines a canonical value for localizable **CHOICE** entries. Each **MAPPINGDEFINITION** MUST define in its contents a corresponding value from a choice.

2.2.6.3.9.1 Schema

```
<xs:complexType name="MAPPINGDEFINITION">
<xs:simpleContent>
<xs:extension base="xs:string">
<xs:attribute name="Value" type="xs:string" />
</xs:extension>
</xs:simpleContent>
</xs:complexType>
```

2.2.6.3.9.2 Attributes

Value: A string that contains a canonical, non-localizable value for a **CHOICE**.

2.2.6.3.9.3 Child Elements

Content: A string that MUST specify exactly the string of a corresponding **CHOICE**.

2.2.6.3.10 MAPPINGDEFINITIONS Type

The **MAPPINGDEFINITIONS** type is a container for one or more **MAPPINGS**. **MAPPINGS** MUST NOT be defined for fields other than **Choice** or **MultiChoice**. There MUST be either no **MAPPINGDEFINITION** elements defined for a **Choice** field, or exactly one **MAPPING** element for each corresponding **CHOICE** element.

2.2.6.3.10.1 Schema

```
<xs:complexType name="MAPPINGDEFINITIONS">
<xs:sequence>
<xs:element name="MAPPING" type="MAPPINGDEFINITION" minOccurs="0" maxOccurs="unbounded" />
</xs:sequence>
</xs:complexType>
```

2.2.6.3.10.2 Attributes

None.

2.2.6.3.10.3 Child Elements

MAPPING: A canonical value mapping for a **CHOICE**.

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.

3 Protocol Details

3.1 Server Details

3.1.1 Abstract Data Model

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

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

Configuration Objects: A set of information about farm configuration. The Configuration Objects are stored in a Configuration Database.

Site Collections: A set of information about all site collections in a content database. Site collection entries are identified by the **Site Collection Identifier** (section 2.2.1.17) and are also represented by either absolute URLs or store relative form URLs.

Sites: A set of information about all sites in a content database. Site entries are identified by site identifiers (section 2.2.1.19) and are also represented by store relative form URLs.

Lists: A set of information about all lists in a content database. List entries are identified by list identifiers (section 2.2.1.10) and are also represented by store relative form URLs.

List Items: A set of information about all list items in a content database. List item entries are identified by list item identifiers (section 2.2.1.11).

Documents: A set of information about all documents in a content database. Document entries are identified by their **Document Identifier** (section 2.2.1.4) and are also are represented by store relative form URLs.

Users: A set of information about all users in a content database. User entries are identified by User Identifiers (section 2.2.1.24).

Site Groups: A set of information about all site groups in a content database. Site group entries are identified by site group identifiers (section 2.2.1.18).

Roles: A set of information about all **roles** in a content database. Role entries are identified by Role Identifiers (section 2.2.1.15).

3.1.2 Timers

An execution timeout timer is set up on the back-end database server to govern the execution time for any requests. The amount of time is governed by a time-out value configured on the back-end database server for all connections.

3.1.3 Initialization

Authentication of the TDS connection to the back-end database server MUST occur before this protocol can be used.

The data structures, stored procedures, and actual data are persisted by the back-end database server within databases, so any operations to initialize the state of the database MUST occur before the back-end database server can use this protocol. This protocol requires that the data for site collections, sites, lists, and document libraries already exist within the back-end database server in a valid state.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

The T-SQL syntax for each stored procedure and result set, and the variables they are composed of, is defined using the T-SQL language specified in [TSQL-Ref]. In the T-SQL syntax, the variable name is followed by the type of the variable, which can optionally have a length value in brackets and can optionally have a default value indicated by an equal 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, because the front-end Web server can access any column with no defined name by ordinal position. Such names are designated in the text using braces in the form {name}. For interoperability, named columns in result sets are specified with what they SHOULD be named, and columns marked with braces SHOULD have no defined name. Front-end Web server implementations MUST NOT rely on any column name in a result set.

The logical sequence of returned values and result sets are indicated in each of the individual stored procedures defined in this section. The TDS protocol controls the actual order and structure of how the T-SQL language formatted information is transported over the wire.

All functions, result sets, and stored procedures are defined using T-SQL.

3.1.5.1 proc_AddDocument

The **proc_AddDocument** stored procedure is called to add a document to the back-end database server with the specified parameters. **proc_AddDocument** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_AddDocument (
@DocSiteId                    uniqueidentifier
,@DocWebId                    uniqueidentifier
,@UserId                      int
,@DocDirName                  nvarchar(256)
,@DocLeafName                 nvarchar(128) OUTPUT
,@NewDocId                    uniqueidentifier
,@DoclibId                    uniqueidentifier
,@NewDoclibRowId              int
,@DocContent                  image
,@DocMetaInfo                 image
,@DocSize                     int
,@DocMetainfoSize             int
,@DocDirty                    bit
,@DocFlags                    tinyint
,@DocIncomingCreatedDTM       datetime
,@DocIncomingDTM              datetime
,@GetWebListForNormalization  bit
,@PutFlags                    int
,@CreateParentDir             bit
,@UrlIsSuggestion             bit
,@ThicketMainFile             bit
,@CharSet                     int
,@AttachmentOp                int
```

```
,@VirusVendorID              int
,@VirusStatus                int
,@VirusInfo                  nvarchar(255)
,@@DocDTM                    datetime OUTPUT
,@fNoQuotaOrLockCheck        bit
,@ChunkSize                  int
,@DocTextptr                 varbinary(16) OUTPUT
);
```

@DocSiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection that will contain the document to be stored. This parameter MUST NOT be NULL.

@DocWebId: The **Site Identifier** (section 2.2.1.19) for the **site** that will contain the document to be stored. This parameter MUST NOT be NULL.

@UserId: The **User Identifier** (section 2.2.1.24) of the **current user** that is making the request to the front-end Web server. This value MUST refer to an existing **User Identifier** for the specified site collection.

@DocDirName: The directory name of the document to be stored. This parameter MUST NOT be NULL.

@DocLeafName: The leaf name of the document to be stored. If *@UrlIsSuggestion* is set to 1, this name can be replaced with a unique name and returned in this output parameter as the actual document leaf name. This parameter MUST NOT be NULL.

@NewDocId: The **Document Identifier** (section 2.2.1.4) of the document to be stored. This parameter MUST NOT be NULL and MUST be unique for a new document.

@DoclibId: The **List Identifier** (section 2.2.1.10) of the **list (1)** or document library in which the document is to be stored.

@NewDoclibRowId: The document library row identifier for the document to be stored. This parameter can be NULL.

@DocContent: The optional document stream of the document.

@DocMetaInfo: The metadata information for the document to be stored. This parameter can be NULL.

@DocSize: The final size, in bytes, of the document stream image to be stored. This parameter MUST be NULL if *@DocContent* is NULL.

@DocMetainfoSize: The size, in bytes, of the document's metadata info. This parameter can be NULL if *@DocMetaInfo* is NULL.

@DocDirty: A bit flag specifying whether the document has dependencies such as links to other items. If this parameter is set to 1, the document has dependencies that MUST subsequently be updated.

@DocFlags: A **Doc Flags** (section 2.2.2.2) value that contains options for adding the document.

@DocIncomingCreatedDTM: A time stamp, in UTC format, that specifies when the document was created.

@DocIncomingDTM: A time stamp, in UTC format, that specifies the document's last modification date.

@GetWebListForNormalization: A bit flag specifying whether to return the **Subsite List Result Set**. If this parameter is set to 1, **proc_AddDocument** MUST return a list of the subsites of the document's containing site in the **Subsite List Result Set** (section 2.2.4.54).

@PutFlags: A **Put Flags Type** (section 2.2.2.5) value that specifies the options for adding the document.

@CreateParentDir: A bit flag specifying whether to create the parent directory for the document to be added if it does not already exist. If this parameter is set to 1, the parent directory specified by *@DocDirName* MUST be created if it does not exist. If this parameter is set to 0, **proc_AddDocument** MUST fail if the parent directory does not exist.

@UrlIsSuggestion: A bit flag specifying whether the *@DocLeafName* provided can be changed to a unique value if it is not unique. If this parameter is set to 1, *@DocLeafName* MUST be updated to get a guaranteed unique URL. If this parameter is set to 0 and the URL is not unique, this procedure MUST fail.

@ThicketMainFile: A bit flag specifying whether the document is a **thicket main file**. If this parameter is set to 1, then the document is a thicket main file.

@CharSet: An optional parameter that specifies a **Windows code page** identifier for the **character set** to be associated with the document.

@AttachmentOp: An **Attachments Flag** (section 2.2.2.1) value that specifies the security checks to be performed by **proc_AddDocument** on this document's URL, based on whether it appears to be an attachment.

@VirusVendorID: An optional parameter that specifies the identifier of the virus scanner that processed this document.

@VirusStatus: A **Virus Status** (section 2.2.1.26) type that specifies the current virus check status of this document.

@VirusInfo: A string containing a provider-specific message returned by the virus scanner when it last processed the document. This value MUST be NULL if the document has not been processed by a virus scanner.

@@DocDTM: An output parameter for the time stamp of the last modification date of the document. This parameter MUST be set to the value of *@DocIncomingDTM* or to the current UTC date if *@DocIncomingDTM* is NULL.

@fNoQuotaOrLockCheck: A bit flag specifying whether to bypass the disk quota and disk lock check. If this parameter is set to 1, the checks are bypassed. If this parameter is set to 0, an explicit check will be made to see if the site is **locked** or if quota is reached.

@ChunkSize: The size, in bytes, of the portion of the document stream in *@DocContent*. If *@ChunkSize* is less than *@DocSize* and a document stream is submitted, *@DocTextptr* MUST be returned to complete filling of the document stream image later. This parameter MUST be NULL if *@DocContent* is NULL. Otherwise, it MUST NOT be NULL.

@DocTextptr: An output parameter containing a pointer set to the storage location of *@DocContent* if execution is successful, *@DocContent* is not NULL, and *@DocSize* is greater than *@ChunkSize*.

Return Values: The **proc_AddDocument** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.

Value	Description
3	The path specified for the document was not found.
5	Access denied error while attempting to create a directory.
80	The document already exists. The attempt to add the document failed. The attempt to add the document stream failed.
212	The disk storage was locked.
1816	Disk quota exceeded.

Result Sets: The **proc_AddDocument** stored procedure MUST return a **Subsite List Result Set** (section 2.2.4.54) if the input parameter *@GetWebListForNormalization* is set to 1 and execution has been successful up to the point of inserting the document. The **Subsite List Result Set** returns a list of URLs for the immed**iate child subsites of the site containing the newly added document. The Subsite List Result Set** MUST contain one row for each **subsite** found.

3.1.5.2 proc_AddListItem

The **proc_AddListItem** stored procedure is called to add a list item to a **list (1)** or document library. **proc_AddListItem** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_AddListItem(
      @SiteId                  uniqueidentifier,
      @WebId                   uniqueidentifier,
      @ListID                  uniqueidentifier,
      @ItemId                  int output,
      @UserId                  int,
      @Size                    int,
      @TimeNow                 datetime,
      @ItemCountDelta          int = 1,
      @ItemName                nvarchar(255) = NULL,
      @UseNvarchar1ItemName    bit = 1,
      @DocFullUrl              nvarchar(260) = NULL,
      @ServerTemplate          int = NULL,
      @IsNotUserDisplayed      bit = NULL,
      @BaseType                int = NULL,
      @CheckSchemaVersion      int = NULL,
      @OnRestore               bit = 0,
      @tp_Ordering             varchar(512) = NULL,
      @tp HasAttachment        bit = NULL,
      @tp_ModerationStatus     int = 0,
      @tp IsCurrent            bit = 1,
      @tp_ItemOrder            float = NULL,
      @tp_InstanceID           int = NULL,
      @tp_GUID                 uniqueidentifier = NULL,
      @tp ID                   int = NULL,
      @tp_Author               int = NULL,
      @tp Editor               int = NULL,
      @tp Modified             datetime = NULL,
      @tp_Created              datetime = NULL,
      @nvarchar1 nvarchar(255) = NULL,
      @nvarchar2 nvarchar(255) = NULL,
      @nvarchar3 nvarchar(255) = NULL,
      @nvarchar4 nvarchar(255) = NULL,
      @nvarchar5 nvarchar(255) = NULL,
      @nvarchar6 nvarchar(255) = NULL,
      @nvarchar7 nvarchar(255) = NULL,
      @nvarchar8 nvarchar(255) = NULL,
      @nvarchar9 nvarchar(255) = NULL,
      @nvarchar10 nvarchar(255) = NULL,
```

```
@nvarchar11 nvarchar(255) = NULL,
@nvarchar12 nvarchar(255) = NULL,
@nvarchar13 nvarchar(255) = NULL,
@nvarchar14 nvarchar(255) = NULL,
@nvarchar15 nvarchar(255) = NULL,
@nvarchar16 nvarchar(255) = NULL,
@nvarchar17 nvarchar(255) = NULL,
@nvarchar18 nvarchar(255) = NULL,
@nvarchar19 nvarchar(255) = NULL,
@nvarchar20 nvarchar(255) = NULL,
@nvarchar21 nvarchar(255) = NULL,
@nvarchar22 nvarchar(255) = NULL,
@nvarchar23 nvarchar(255) = NULL,
@nvarchar24 nvarchar(255) = NULL,
@nvarchar25 nvarchar(255) = NULL,
@nvarchar26 nvarchar(255) = NULL,
@nvarchar27 nvarchar(255) = NULL,
@nvarchar28 nvarchar(255) = NULL,
@nvarchar29 nvarchar(255) = NULL,
@nvarchar30 nvarchar(255) = NULL,
@nvarchar31 nvarchar(255) = NULL,
@nvarchar32 nvarchar(255) = NULL,
@nvarchar33 nvarchar(255) = NULL,
@nvarchar34 nvarchar(255) = NULL,
@nvarchar35 nvarchar(255) = NULL,
@nvarchar36 nvarchar(255) = NULL,
@nvarchar37 nvarchar(255) = NULL,
@nvarchar38 nvarchar(255) = NULL,
@nvarchar39 nvarchar(255) = NULL,
@nvarchar40 nvarchar(255) = NULL,
@nvarchar41 nvarchar(255) = NULL,
@nvarchar42 nvarchar(255) = NULL,
@nvarchar43 nvarchar(255) = NULL,
@nvarchar44 nvarchar(255) = NULL,
@nvarchar45 nvarchar(255) = NULL,
@nvarchar46 nvarchar(255) = NULL,
@nvarchar47 nvarchar(255) = NULL,
@nvarchar48 nvarchar(255) = NULL,
@nvarchar49 nvarchar(255) = NULL,
@nvarchar50 nvarchar(255) = NULL,
@nvarchar51 nvarchar(255) = NULL,
@nvarchar52 nvarchar(255) = NULL,
@nvarchar53 nvarchar(255) = NULL,
@nvarchar54 nvarchar(255) = NULL,
@nvarchar55 nvarchar(255) = NULL,
@nvarchar56 nvarchar(255) = NULL,
@nvarchar57 nvarchar(255) = NULL,
@nvarchar58 nvarchar(255) = NULL,
@nvarchar59 nvarchar(255) = NULL,
@nvarchar60 nvarchar(255) = NULL,
@nvarchar61 nvarchar(255) = NULL,
@nvarchar62 nvarchar(255) = NULL,
@nvarchar63 nvarchar(255) = NULL,
@nvarchar64 nvarchar(255) = NULL,
@int1 int = NULL,
@int2 int = NULL,
@int3 int = NULL,
@int4 int = NULL,
@int5 int = NULL,
@int6 int = NULL,
@int7 int = NULL,
@int8 int = NULL,
@int9 int = NULL,
@int10 int = NULL,
@int11 int = NULL,
@int12 int = NULL,
@int13 int = NULL,
@int14 int = NULL,
@int15 int = NULL,
```

```
@int16 int = NULL,
@float1 float = NULL,
@float2 float = NULL,
@float3 float = NULL,
@float4 float = NULL,
@float5 float = NULL,
@float6 float = NULL,
@float7 float = NULL,
@float8 float = NULL,
@float9 float = NULL,
@float10 float = NULL,
@float11 float = NULL,
@float12 float = NULL,
@float13 float = NULL,
@float14 float = NULL,
@float15 float = NULL,
@float16 float = NULL,
@float17 float = NULL,
@float18 float = NULL,
@float19 float = NULL,
@float20 float = NULL,
@float21 float = NULL,
@float22 float = NULL,
@float23 float = NULL,
@float24 float = NULL,
@float25 float = NULL,
@float26 float = NULL,
@float27 float = NULL,
@float28 float = NULL,
@float29 float = NULL,
@float30 float = NULL,
@float31 float = NULL,
@float32 float = NULL,
@datetime1 datetime = NULL,
@datetime2 datetime = NULL,
@datetime3 datetime = NULL,
@datetime4 datetime = NULL,
@datetime5 datetime = NULL,
@datetime6 datetime = NULL,
@datetime7 datetime = NULL,
@datetime8 datetime = NULL,
@datetime9 datetime = NULL,
@datetime10 datetime = NULL,
@datetime11 datetime = NULL,
@datetime12 datetime = NULL,
@datetime13 datetime = NULL,
@datetime14 datetime = NULL,
@datetime15 datetime = NULL,
@datetime16 datetime = NULL,
@bit1 bit = NULL,
@bit2 bit = NULL,
@bit3 bit = NULL,
@bit4 bit = NULL,
@bit5 bit = NULL,
@bit6 bit = NULL,
@bit7 bit = NULL,
@bit8 bit = NULL,
@bit9 bit = NULL,
@bit10 bit = NULL,
@bit11 bit = NULL,
@bit12 bit = NULL,
@bit13 bit = NULL,
@bit14 bit = NULL,
@bit15 bit = NULL,
@bit16 bit = NULL,
@uniqueidentifier1 uniqueidentifier = NULL,
@ntext1 ntext = NULL,
@ntext2 ntext = NULL,
@ntext3 ntext = NULL,
```

```
        @ntext4 ntext = NULL,
        @ntext5 ntext = NULL,
        @ntext6 ntext = NULL,
        @ntext7 ntext = NULL,
        @ntext8 ntext = NULL,
        @ntext9 ntext = NULL,
        @ntext10 ntext = NULL,
        @ntext11 ntext = NULL,
        @ntext12 ntext = NULL,
        @ntext13 ntext = NULL,
        @ntext14 ntext = NULL,
        @ntext15 ntext = NULL,
        @ntext16 ntext = NULL,
        @ntext17 ntext = NULL,
        @ntext18 ntext = NULL,
        @ntext19 ntext = NULL,
        @ntext20 ntext = NULL,
        @ntext21 ntext = NULL,
        @ntext22 ntext = NULL,
        @ntext23 ntext = NULL,
        @ntext24 ntext = NULL,
        @ntext25 ntext = NULL,
        @ntext26 ntext = NULL,
        @ntext27 ntext = NULL,
        @ntext28 ntext = NULL,
        @ntext29 ntext = NULL,
        @ntext30 ntext = NULL,
        @ntext31 ntext = NULL,
        @ntext32 ntext = NULL,
        @sql_variant1 sql_variant = NULL,
        @error_sql_variant1 int = 0,
        @sql_variant2 sql_variant = NULL,
        @error_sql_variant2 int = 0,
        @sql_variant3 sql_variant = NULL,
        @error_sql_variant3 int = 0,
        @sql_variant4 sql_variant = NULL,
        @error_sql_variant4 int = 0,
        @sql_variant5 sql_variant = NULL,
        @error_sql_variant5 int = 0,
        @sql_variant6 sql_variant = NULL,
        @error_sql_variant6 int = 0,
        @sql_variant7 sql_variant = NULL,
        @error_sql_variant7 int = 0,
        @sql_variant8 sql_variant = NULL,
        @error_sql_variant8 int = 0
    );
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection containing the list (1) that the list item is being added to.

@WebId: The **Site Identifier** (section 2.2.1.19) for the **site** containing the list (1) that the list item is being added to.

@ListID: The **List Identifier** (section 2.2.1.10) of the list (1) that the list item is being added to.

@ItemId: An output parameter that returns the identifier of the list item that has been added. If *@tp_ID* is not NULL, **proc_AddListItem** MUST use the value specified by *@tp_ID*. If *@ItemId* is NULL, **proc_AddListItem** MUST generate a **List Item Identifier** (section 2.2.1.11) for the specified *@WebId* and *@ListID*.

@UserId: The **User Identifier** (section 2.2.1.24) for the current user. **proc_AddListItem** uses this for purposes of permission-checking. This value MUST refer to an existing **User Identifier** for the specified site collection.

@Size: The size, in bytes, of the list item row to be added. This parameter MUST NOT be NULL.

@TimeNow: The current time, in UTC format, on the back-end database servers.

@ItemCountDelta: The number to be added to the list item count of the containing list (1).

@ItemName: The display name of the list item.

@UseNvarchar1ItemName: If *@ItemName* is NULL, this bit flag specifies whether to use the content of *@nvarchar1* for the display name of the list item.

@DocFullUrl: The full store-relative form URL.

@ServerTemplate: The identifier for the **List Server Template** (section 2.2.1.12) defining the base structure of the list (1) containing this list item.

@IsNotUserDisplayed: A bit flag specifying whether the user name is not displayed with list items.

@BaseType: The **List Base Type** (section 2.2.1.9) of the list (1) containing the list item.

@CheckSchemaVersion: Specifies an OPTIONAL schema version number to compare with the list schema version number. If this parameter is not NULL, the version numbers MUST match for successful completion.

@OnRestore: A bit flag that specifies whether this list item is being inserted by an implementation-specific back-up restore operation.

@tp_Ordering: Specifies the threading structure for this list item in a previous discussion board list as a concatenation of time stamp values in yyyyMMddHHmmss format.

@tp_HasAttachment: A bit flag that specifies whether the list item has an associated attachment.

@tp_ModerationStatus: A **Moderation Status** (section 2.2.1.13) value specifying the current moderation approval status of this list item.

@tp_IsCurrent: A bit flag that specifies whether this is the current version of the list item.

@tp_ItemOrder: Specifies the relative positioning order in which to view the list item when displayed with other List Items from the same list (1).

@tp_InstanceID: If this list item is associated with a particular instance of a recurring meeting, this value specifies the integer identifier of that instance. For all other list items, this parameter MUST be NULL.

@tp_GUID: A **List Item Identifier** (section 2.2.1.11) value that uniquely identifies this list item within the **UserData table** (section 2.2.5.6).

@tp_ID: The OPTIONAL integer identifier specified for this list item. If this parameter is NOT NULL, **proc_AddListItem** MUST use this value for the identifier of the list item to be added.

@tp_Author: The **User Identifier** for the user who created the list item.

@tp_Editor: The **User Identifier** for the user who last edited the list item.

@tp_Modified: A time stamp, in UTC format, that specifies when this list item was last modified.

@tp_Created: A time stamp, in UTC format, that specifies when this list item was created.

The next nine columns are duplicated a variable number of times. Each instance of these individual column names is differentiated by a suffix with a value indicated in the column description, which replaces the placeholder '#' symbol shown as follows.

@nvarchar#: User-defined columns in the list containing values of type nvarchar. There are 64 columns numbered from 1 to 64. If the column contains no data, the value MUST be NULL.

@int#: User-defined columns in the list containing values of type int. There are 16 columns numbered from 1 to 16. If the column contains no data, the value MUST be NULL.

@float#: User-defined columns in the list containing values of type float. There are 32 columns numbered from 1 to 32. If the column contains no data, the value MUST be NULL.

@datetime#: User-defined columns in the list containing values of type datetime. There are 16 columns numbered from 1 to 16. If the column contains no data, the value MUST be NULL.

@bit#: User-defined columns in the list containing values of type bit. There are 16 columns numbered from 1 to 16. If the column contains no data, the value MUST be NULL.

@uniqueidentifier1: A user-defined column in the list containing values of type uniqueidentifier. If the column contains no data, the value MUST be NULL.

@ntext#: User-defined columns in the list containing values of type ntext. There are 32 columns numbered from 1 to 32. If the column contains no data, the value MUST be NULL.

@sql_variant#: User-defined columns in the list containing values of type sql_variant. There are eight columns numbered from 1 to 8. If the column contains no data, the value MUST be NULL.

@error_sql_variant#: An integer specifying the type to be applied to the corresponding values specified as arguments for the parameter *@sql_variant#*. There are eight columns numbered from 1 to 8. Valid values for this parameter are listed in the following table.

Value	Description
1	Convert the argument value to a varbinary(2).
2	Convert the argument value to a bit.
3	Convert the argument value to a float.
4	Convert the argument value to a datetime.

Return values: The **proc_AddListItem** stored procedure MUST return an integer return code, which MUST be in the following table, or MUST be a SQL Server *@@Error* result value.

Value	Description
0	Successful execution
1	Incorrect function.
2	Postprocessing of the list item failed because a prerequisite list item was not found.
3	The directory specified for the list item does not exist.
5	The attempt to create a directory or document failed because the user does not have sufficient permissions.
13	The list item to be added is not valid.
16	Adding the list item caused updating of an existing list item to fail.
31	The process cannot access the file because it is being used by another process.
33	Cannot move directories that contain checked-out files.
50	A list item could not be deleted.
51	Windows cannot find the network path.

Value	Description
80	The document being added to the list (1) already exists.
87	Unable to add the list item because the input parameters do not match existing list items, or an error occurred during a table update operation.
138	The list (1) could not be moved to the specified location.
144	The directory is not empty.
161	A directory that spans sites cannot be moved.
183	The list item being added already exists in the list (2).
206	The file or directory name is too long.
212	The database for the site collection is locked.
214	Too many dynamic-link modules are attached to this program.
1150	Failed to update the list (1).
1638	The current schema version of the list (1) does not match the value in *@CheckSchemaVersion*.
1816	The site collection has exceeded its allocated size quota.
8398	A directory could not be deleted.

Result Sets: The **proc_AddListItem** stored procedure MUST NOT return any result set.

3.1.5.3 proc_CheckoutDocument

The **proc_CheckoutDocument** stored procedure is called in order to place a short-term lock on a document, refresh or release an existing short-term lock, or to **check out** a document. **proc_CheckoutDocument** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_CheckoutDocument (
@SiteId               uniqueidentifier
,@WebId               uniqueidentifier
,@DirName             nvarchar(256)
,@LeafName            nvarchar(128)
,@UserId              int
,@CheckoutTimeout     int
,@RefreshCheckout     bit
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection containing the document to be checked out, locked, or unlocked.

@WebId: The **Site Identifier** (section 2.2.1.19) for the **site** containing the document.

@DirName: The directory name of the document.

@LeafName: The leaf name of the document.

@UserId: The user identifier for the current user who is requesting a short-term lock or checking out the document. This value MUST refer to an existing user identifier for the specified site collection.

@CheckoutTimeout: Specifies the remaining time, in minutes, that short-term locking will be in effect for the document. A value of 0 means that the existing short-term lock MUST be released. The

@CheckoutTimeout parameter MUST be NULL if the document is being checked out instead of having a short-term lock applied.

@RefreshCheckout: A bit flag specifying whether the short-term lock on the document can be refreshed. If this parameter is set to 1, the existing short-term lock on the document MUST be refreshed for the number of minutes specified by the *@CheckoutTimeout* parameter. This parameter MUST be set to 0 to check out the document or to request a new short-term lock.

Return Values: The **proc_CheckoutDocument** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
3	File not found. A document corresponding to the specified *@SiteId*, *@WebId*, *@DirName*, and *@LeafName* parameters was not found.
33	Short-term lock error. The document cannot have a short-term lock applied because another user has the document checked out, or another user holds a short-term lock on the document.
212	Site collection locked. The site collection is in disk write lock.
1630	Unsupported document type. The document specified is not valid for check-out; folders and sites cannot be checked out.
1816	User Quota is exceeded and no additional users can be added to the site.

Result Sets: The **proc_CheckoutDocument** stored procedure MUST return a **Link Info Single Doc Result Set** (section 2.2.4.29) and a **Document Metadata Result Set** (section 2.2.4.13).

3.1.5.4 proc_CreateDir

The **proc_CreateDir** stored procedure is called to create a directory or folder with a specified name at a specified location. **proc_CreateDir** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_CreateDir (
@DirSiteId              uniqueidentifier
,@DirWebId              uniqueidentifier
,@DirDirName            nvarchar(256) OUTPUT
,@DirLeafName           nvarchar(128) OUTPUT
,@Flags                 int
,@UserId                int = null
,@DirId                 uniqueidentifier = null
,@DoclibRowIdRequired   int = null
);
```

@DirSiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection containing the directory to be created.

@DirWebId: The **Site Identifier** (section 2.2.1.19) for the **site** containing the directory to be created.

@DirDirName: Both an input and an output parameter. The input parameter MUST specify the URL of the parent location in which the specified directory is to be created. The directory name of the created directory MUST be returned as the output parameter value.

@DirLeafName: Both an input and an output parameter. The input parameter MUST specify the name of the directory to be created in the parent location specified by the *@DirDirName* parameter. The leaf name of the created directory MUST be returned as the output parameter value.

@Flags: The **Attachments Flag** (section 2.2.2.1) specifying whether the *@DirDirName* appears to be an attachment.

@UserId: The **User Identifier** (section 2.2.1.24) for the current user who is directly or indirectly requesting the directory creation. This value can be NULL. If set, this MUST be used by **proc_CreateDir** for permission checking and setting ownership of the created directory. Otherwise, the ownership of the created directory MUST be set from the container.

@DirId: The identifier for the created directory. If this parameter is passed in to **proc_CreateDir** with a non-NULL value, this value MUST be used as the **Document Identifier** (section 2.2.1.4) for the directory to be created. If this parameter is passed in with a NULL value, a new uniqueidentifier value MUST be generated for the directory.

@DoclibRowIdRequired: The **List Item Identifier** (section 2.2.1.11) to be used for this directory if it is created within a document library. If this parameter is not NULL and the directory to be created is within a document library, **proc_CreateDir** MUST set this value as the **List Item Identifier** for the created directory. This parameter MUST be NULL for directories that are not created within a document library.

Return Values: The **proc_CreateDir** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
3	The parent directory was not found.
5	The current user does not have sufficient permissions to create the directory at the specified location.
80	The specified directory already exists.
212	The site collection is locked.
1816	There is not enough database quota for the current user to complete the operation.

Result Sets: The **proc_CreateDir** stored procedure MUST NOT return any result set.

3.1.5.5 proc_DeleteAllDocumentVersions

The **proc_DeleteAllDocumentVersions** stored procedure is called to delete the older (major) versions of a document. **proc_DeleteAllDocumentVersions** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc DeleteAllDocumentVersions (
@DocSiteId     uniqueidentifier
,@DocDirName   nvarchar(256)
,@DocLeafName  nvarchar(128)
);
```

@DocSiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the document whose specified versions are being deleted.

@DocDirName: The directory name of the document for which to delete the specified version. If there is no slash in the directory name, the value MUST be an **empty string**.

@DocLeafName: The leaf name of the document.

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

Result Sets: The **proc_DeleteAllDocumentVersions** stored procedure MUST NOT return any result set.

3.1.5.6 proc_DeleteDocumentVersion

The **proc_DeleteDocumentVersion** stored procedure is called to delete a **document version**. A current version cannot be deleted. **proc_DeleteDocumentVersion** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_DeleteDocumentVersion (
@DocSiteId     uniqueidentifier
,@DocDirName   nvarchar(256)
,@DocLeafName  nvarchar(128)
,@DocVersion   int
);
```

@DocSiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the document whose specified version is being deleted.

@DocDirName: The directory name of the document to delete. If there is no slash in the directory name, the value MUST be an **empty string**.

@DocLeafName: The leaf name of the document to delete.

@DocVersion: The version of the document to delete, as seen in the front-end Web server. This value MUST NOT be a current version.

Return Values: The **proc_DeleteDocumentVersion** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
2	The version specified by *@DocSiteId*, *@DocDirName*, *@DocLeafName*, and *@DocVersion* cannot be found.

Result Sets: The **proc_DeleteDocumentVersion** stored procedure MUST NOT return any result set.

3.1.5.7 proc_DeleteUrl

The **proc_DeleteUrl** stored procedure is called to delete a document. **proc_DeleteUrl** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_DeleteUrl (
@WebSiteId                  uniqueidentifier
,@WebId                     uniqueidentifier
,@Url                       nvarchar(260)
,@UserId                    int
,@ListDeletedUrls           bit = 1
,@AttachmentsFlag           tinyint = 0
,@AttachmentOp              int = 3
,@IgnoreCheckedOutFiles     bit = 0
,@FailedUrl                 nvarchar(260) = null OUTPUT
,@DeleteFlags               int = 0
);
```

@WebSiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the document.

@WebId: The **Site Identifier** (section 2.2.1.19) of the site containing the document.

@Url: The store-relative form URL of the document.

@UserId: The **User Identifier** (section 2.2.1.24) for the current user that is requesting the operation. This value MUST refer to an existing **User Identifier** for the specified site collection.

@ListDeletedUrls: If this bit is set to 1, it specifies that result set information about the deleted documents is requested.

@AttachmentsFlag: An **Attachments Flag** (section 2.2.2.1) describing the document specified by @Url.

@AttachmentOp: If this bit is set to 1, it specifies that on attachment deletion, the attachment folder record in the store MUST be updated to reflect whether it has any attachments remaining. Any other value specifies the stored procedure. This information and other metadata in the store folder MUST be refreshed. This includes last modified date and time, the user identifier of the current user who made the modification, and incrementing of the internal folder version.

@IgnoreCheckedOutFiles: If this bit is set to 1, it specifies that the stored procedure MUST proceed with the delete operation on a folder even when it contains locked files.

@FailedUrl: An output parameter indicating the URL at which the delete operation failed. This parameter MUST be set to NULL if the deletion was successful.

@DeleteFlags: Specifies a batched delete option. This is a reserved value and MUST be set to zero.

Return Values: The **proc_DeleteUrl** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
3	The specified URL was not found.
33	Cannot delete a folder containing checked-out or locked files.
50	Cannot delete a **site** or an attachments folder.
51	Cannot delete a forms folder.
138	Cannot delete a URL containing lists.
161	Cannot delete a URL that contains sites.
8398	There was an error deleting a **list (1)**. At least one list could not be deleted.

Result Sets: The **proc_DeleteUrl** stored procedure MUST return a **Deleted Documents Result Set** (section 2.2.4.8) on successful completion if *@ListDeletedUrls* is set to 1.

3.1.5.8 proc_DirtyDependents

The **proc_DirtyDependents** stored procedure is called to mark all items that depend on a given document, system setting, as "dirty" so that action can be taken to update them as necessary. A "dependent" item is an item that requires an update to its metadata when another item is modified. **proc_DirtyDependents** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_DirtyDependents (
@SiteId          uniqueidentifier
,@DepType        tinyint
,@DepDesc        nvarchar(270)
,@DepDescLike    nvarchar(260) = null
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) for a site collection that contains the items with dependencies.

@DepType: The dependency type. Valid values for this parameter are listed in the following table.

Value	Description
1	Doc Dependency: Updates an item depending on the specified document. The *@DepDesc* parameter is the store-relative form URL of the document that has changed.
3	Configuration Dependency: Updates an item depending on site configuration. The *@DepDesc* parameter is the metakey for the metadata that has changed.

@DepDesc: Specifies the dependency description parameter, which varies according to the value of *@DepType*, as described in the preceding table.

@DepDescLike: Unused and MUST be ignored.

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

Result Sets: The **proc_DirtyDependents** stored procedure MUST NOT return any result set.

3.1.5.9 proc_EnumLists

The **proc_EnumLists** stored procedure returns a list of the lists in a **site**, along with their associated metadata. **proc_EnumLists** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_EnumLists (
@WebId          uniqueidentifier
,@Collation     nvarchar(32)
,@BaseType      int = null
,@BaseType2     int = null
,@BaseType3     int = null
,@BaseType4     int = null
,@FRootFolder   bit = null
,@ListFlags     int = null
,@FAclInfo      int = null
);
```

@WebId: The **Site Identifier** (section 2.2.1.19) for the site containing the requested lists.

@Collation: The Windows collation name string identifier that follows the format for the T-SQL COLLATE clause. This MUST be the Collation Name of one of the valid **Collation Order Enumeration** (section 2.2.1.3) values, with the case-insensitive and accent-sensitive flags set.

@BaseType: A parameter that restricts the returned lists by **List Base** type (section 2.2.1.9). If this parameter is set to a valid **List Base Type** value, the lists returned MUST be restricted to lists with this **List Base Type** and with the **List Base Type** specified by *@BaseType2*, *@BaseType3*, and *@BaseType4*, if any. If this parameter is set to -1 or NULL, the lists retrieved MUST NOT be restricted by **List Base Type** and parameters *@BaseType2*, *@BaseType3*, and *@BaseType4* MUST be ignored by **proc_EnumLists**.

@BaseType2: An additional parameter used to specify an additional **List Base Type** for lists returned, depending on the value of the *@BaseType* parameter. If *@BaseType* is set to -1 or NULL, this parameter MUST be ignored. If *@BaseType* is set to a valid **List Base Type** value, returned lists MUST be restricted to those whose **List Base Type** matches any specified by the four *@BaseType* parameters. Nonvalid and duplicated **List Base Type** values MUST have no effect on returned lists.

@BaseType3: An additional parameter used to specify an additional **List Base Type** for lists returned, depending on the value of the *@BaseType* parameter. If *@BaseType* is set to -1 or NULL, this parameter MUST be ignored. If *@BaseType* is set to a valid **List Base Type** value, returned lists MUST be restricted to those whose **List Base Type** matches any specified by the four *@BaseType* parameters. Nonvalid and duplicated **List Base Type** values MUST have no effect on returned lists.

@BaseType4: An additional parameter used to specify an additional **List Base Type** for lists returned, depending on the value of the *@BaseType* parameter. If *@BaseType* is set to -1 or NULL, this parameter MUST be ignored. If *@BaseType* is set to a valid **List Base Type** value, returned lists MUST be restricted to those whose **List Base Type** matches any specified by the four *@BaseType* parameters. Nonvalid and duplicated **List Base Type** values MUST have no effect on returned lists.

@FRootFolder: A bit specifying whether information about the root folder URL is requested. If this parameter is set to 1, the information MUST be returned in the tp_RootFolder column in **List Information Result Set** (section 2.2.4.31). Otherwise, the tp_RootFolder column MUST be returned with NULL values.

@ListFlags: A **List Flags** (section 2.2.2.4) value restricting the data returned by the **List Information Result Set**. If this parameter is not NULL, the stored procedure MUST only return data in the **List Information Result Set** for lists with a tp_Flags value matching the *@ListFlags* value. Otherwise, this parameter MUST be ignored.

@FAclInfo: A flag specifying whether ACL information is requested. If this parameter is set to 1, the ACL information MUST be returned in the tp_ACL column in the **List Information Result Set**. Otherwise, the tp_ACL column MUST be returned with NULL values.

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

Result Sets: The **proc_EnumLists** stored procedure MUST return **List Information Result Set** (section 2.2.4.31). The **List Information Result Set** MUST be returned and MUST contain zero or more rows, one for each list that matches the specified input parameters.

3.1.5.10 proc_FetchDocForHttpGet

The **proc_FetchDocForHttpGet** stored procedure is called to fetch a document for the **HTTP GET** and HEAD operations and to provide information necessary to render the document on the front-end Web server. Different sets of information are provided depending on the type of request (HEAD or GET) or the type of document requested (such as a file, a web page, or a list item view webpage). **proc_FetchDocForHttpGet** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_FetchDocForHttpGet (
@DocSiteId                  uniqueidentifier
,@DocDirName                nvarchar(256)
,@DocLeafName               nvarchar(128)
,@LooksLikeAttachmentFile   bit
,@IfModifiedSince           datetime
,@FetchType                 int
,@ValidationType            int
,@RedirectUrl               nvarchar(260) OUTPUT
,@RedirectLangTemplatePick  int OUTPUT
,@ClientVersion             int
,@ClientId                  uniqueidentifier
,@PageView                  tinyint
,@SystemID                  tSystemID
,@CurrentVirusVendorID      int
```

```
,@ChunkSize                    int
,@GetSandbox                   bit OUTPUT
);
```

@DocSiteId: The **Site Collection Identifier** (section 2.2.1.17) for a site collection containing the document.

@DocDirName: The directory name of the requested document.

@DocLeafName: The leaf name of the requested document.

@LooksLikeAttachmentFile: Set to 1 if the URL string indicates that the document is an attachment. Otherwise, the value is set to 0.

@IfModifiedSince: If the caller has a cached copy of the document, *@IfModifiedSince* indicates the last modified time value for the cached copy. Otherwise, *@IfModifiedSince* MUST be NULL.

@FetchType: Indicates the type of request. This parameter MUST be set to 0 to specify that information is requested to satisfy a normal HTTP GET and it MUST be set to 1 if the information is required only to satisfy an **HTTP HEAD** request.

@ValidationType: Specifies the criteria used to determine whether the document contents return. This parameter is used for cache optimization. Valid values for this parameter are listed in the following table.

Value	Description
0	None
1	E-Tag: Return document content if the front-end Web server version specified by input parameter *@ClientVersion* does not match the document version in the back-end database server, or if the **Document Identifier** (section 2.2.1.4) specified by input parameter *@ClientId* does not match the **Document Identifier** (section 2.2.1.4) in the back-end database server.
2	Last modified time: Return document content if the Last Modified time specified by the input parameter *@IfModifiedSince* is earlier than the Last Modified time of the document in the back-end database server.
3	Both E-Tag and Last modified time: Return document content if the front-end Web server version specified by input parameter *@ClientVersion* does not match the document version in the back-end database server, or if the **Document Identifier** (section 2.2.1.4) specified by input parameter *@ClientId* does not match the **Document Identifier** (section 2.2.1.4) in the back-end database server, or if the Last Modified time specified by the input parameter *@IfModifiedSince* is earlier than the Last Modified time of the document in the back-end database server.

@RedirectUrl: The store-relative form URL to which the request is redirected. This is an output parameter.

@RedirectLangTemplatePick: The language for the web, which is used to redirect to the appropriate language-specific site template provisioning page. This is an output parameter.

@ClientVersion: The version of the document as seen in the front-end Web server. It is used for client cache validation, as indicated by the value of *@ValidationType*.

@ClientId: The **Document Identifier** (section 2.2.1.4) as seen in the front-end Web server. It is used for validation as indicated by the value of *@ValidationType*.

@PageView: Values other than NULL indicate that the document is a view webpage, which renders an item or items in a **list (1)** or document library. A value of 1 indicates that the personalized metadata for specified user. The value 0 or other non-NULL values indicate that information is requested as seen

by all users. A NULL value indicates that the document consists of non-ASPX pages or ASPX pages that are fetched outside of a page rendering context.

@SystemID: The **SystemID** of the user originating the request. This parameter MUST refer to a valid, non-deleted user in the specified site collection, or be set to NULL to indicate an anonymous user.

@CurrentVirusVendorID: The identifier of the anti-virus vendor registered for the document.

@ChunkSize: Specifies the size, in bytes, that the document MUST be in order for the document's content to be returned as part of this query. If the document is larger than this value, a single zero byte is returned, and the front-end Web server can request the remainder of the document in a subsequent operation.

@GetSandbox: A bit output parameter. If the document is checked out by a user who is not a **principal (1)**, then the bit set to 0. Otherwise, the value is set to 1.

Return Values: The **proc_FetchDocForHttpGet** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
2	The document specified by *@DocSiteId*, *@DocDirName*, and *@DocLeafName* was not found.
18	Successful execution, file not modified: Based on input parameters, the document's copy in the client cache is valid and can be used.
144	The requested URL is a folder in a document library other than the root folder.
146	The document does not exist, but a welcome page was found, as specified by the output parameter.
149	The **site** that contains the requested URL has not yet been provisioned and the request SHOULD be redirected to the **site template** picker page. The site template selection page is an implementation-specific capability.
1168	The site collection specified by *@DocSiteId* was not found.
1244	Operation rejected: an anonymous user has requested a sandboxed copy of the file and MUST be authenticated to check if it is the same user who has checked out the file.
1271	The site is read/write locked.

Result Sets:

The **proc_FetchDocForHttpGet** stored procedure MUST return an **HTTP Document Metadata Result Set** (section 2.2.4.24).

The **proc_FetchDocForHttpGet** stored procedure MUST return a **Document Content Stream Result Set** (section 2.2.4.10) if *@FetchType* is NOT set to 1 (not indicating an HTTP HEAD-only request). **Document Content Stream Result Set** MUST return either zero or one rows. If the document is modified, a single row MUST be returned. A document is considered modified subject to the semantics indicated by the input parameter *@ValidationType*, or if its Virus Vendor ID has been updated since the client last retrieved it. In such a case, the return code 18 MUST be returned upon successful completion of **proc_FetchDocForHttpGet**. Otherwise, zero rows MUST be returned.

The **proc_FetchDocForHttpGet** stored procedure MUST return a **User Information Result Set** (section 2.2.4.59) if *@PageView* is NULL.

The **proc_FetchDocForHttpGet** stored procedure MUST return a **Site Metadata Result Set** (section 2.2.4.51). The **Site Metadata Result Set** contains metadata for the site containing the specified

document. **Site Metadata Result Set** MUST be returned only if the input parameter *@PageView* is not NULL.

The **proc_FetchDocForHttpGet** stored procedure MUST return a **Web Parts Metadata (Personalized) Result Set** (section 2.2.4.64) if *@PageView* is 1. The **Web Parts Metadata (Personalized) Result Set** contains the core metadata about the Web Parts appearing on the specified document, personalized for a specified user and MUST contain one row per Web Part.

The **proc_FetchDocForHttpGet** stored procedure MUST return a **Web Parts Metadata (Nonpersonalized) Result Set** (section 2.2.4.63) if *@PageView* is not NULL and not 1. The **Web Parts Metadata (Nonpersonalized) Result Set** contains the core metadata about the Web Parts appearing on the specified document and MUST contain one row per Web Part.

The **proc_FetchDocForHttpGet** stored procedure MUST return a **List Metadata Result Set** (section 2.2.4.32) ONLY if *@PageView* is not NULL. **List Metadata Result Set** contains the metadata for the lists associated with the Web Parts that are included on the specified document. **List Metadata Result Set** MUST return one row for each associated list (1).

The **proc_FetchDocForHttpGet** stored procedure MUST return a **List Web Parts Result Set** (section 2.2.4.33) if lists associated with the specified document exist. The **List Web Parts Result Set** contains information about the Web Parts related to the lists and it MUST contain one row per Web Part registered for each list. **List Web Parts Result Set** can be empty.

3.1.5.11 proc_FetchDocForRead

The **proc_FetchDocForRead** stored procedure is called to request the metadata information and content of a document. **proc_FetchDocForRead** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_FetchDocForRead (
@DocSiteId                    uniqueidentifier
,@DocWebId                    uniqueidentifier
,@DocDirName                  nvarchar(256)
,@DocLeafName                 nvarchar(128)
,@DocFullUrl                  nvarchar(260)
,@LooksLikeAttachmentFile     bit
,@GetMetaInfo                 bit
,@GetContent                  bit
,@GetWebListForNormalization  bit
,@bGetContainingList          bit
,@GetDirty                    bit
,@UserId                      int
,@Version                     int
,@ChunkSize                   int
,@GetSandbox                  bit OUTPUT
);
```

@DocSiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the requested document.

@DocWebId: The **Site Identifier** (section 2.2.1.19) of the **site** containing the requested document.

@DocDirName: The directory name of the requested document.

@DocLeafName: The leaf name of the requested document.

@DocFullUrl: This parameter is not used.

@LooksLikeAttachmentFile: A bit flag specifying whether the requested document appears to the front-end Web server to be an attachment to a list item. If this flag is set to 1, **proc_FetchDocForRead** MUST determine whether the current user has sufficient permissions to retrieve the requested document as an attachment.

@GetMetaInfo: A bit flag specifying whether to return the metadata of the document or not. If this flag is set to 1, **proc_FetchDocForRead** MUST return the **Link Info Single Doc Result Set** (section 2.2.4.29).

@GetContent: A bit flag specifying whether to return the content of the document or not. If this flag is set to 1, **proc_FetchDocForRead** MUST return the content of the requested document in the **Document Information and Content (Read) Result Set** (section 2.2.4.11) or the **Document Version Information and Content (Read) Result Set** (section 2.2.4.14).

@GetWebListForNormalization: A bit flag specifying whether to return the list of all sites that have the current site specified by *@DocWebId* as the immediate parent site. If this flag is set to 1, **proc_FetchDocForRead** MUST return a list of child sites in the **Subsite List Result Set** (section 2.2.4.54).

@bGetContainingList: A bit flag specifying whether to return the **list (1)** containing the requested document. If this flag is set to 1, **proc_FetchDocForRead** MAY return the **List MetaData Result Set** (section 2.2.4.32) or **Empty List Result Set** (section 2.2.4.19).

@GetDirty: A bit flag specifying whether to return the content of the document or not. If this flag is set to 1, **proc_FetchDocForRead** MUST return the **Dirty Result Set** (section 2.2.4.9).

@UserId: The **User Identifier** (section 2.2.1.24) for the current user who is requesting the information. This is used by **proc_FetchDocForRead** for permission checking.

@Version: The UI version number of the document that is being requested. A value of -1 specifies the most recent version of the document. If the *@GetContent* flag is set to 1 and *@Version* is negative, the server MUST return the **Document Information and Content (Read) Result Set**. If *@GetContent* flag is set to 1 and *@Version* is not negative, the server MUST return the **Document Version Information and Content (Read) Result Set** (section 2.2.4.14).

@ChunkSize: The maximum size, in bytes, of the document content image to be returned in the **Document Information and Content (Read) Result Set**. If the document content image size is larger than *@ChunkSize*, a single zero byte is returned as the document content image, and the front-end Web server can request the content of the document in a subsequent operation.

@GetSandbox: A bit output parameter. This parameter MUST be set to 1 if the value of *@Version* is -2 and the value of *@GetContent* is set to 0, else it MUST be set to 0.

Return Values: The **proc_FetchDocForRead** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
3	The document does not exist.

Result Sets:

The **proc_FetchDocForRead** stored procedure MUST return a **Subsite List Result Set** (section 2.2.4.54) if *@GetWebListForNormalization* is set to 1. The **Subsite List Result Set** MUST contain one row for each subsite within the specified **parent site**, and MUST contain no rows if there are no such subsites.

The **proc_FetchDocForRead** stored procedure MUST return a **Document Information and Content (Read) Result Set** if *@GetContent* is set to 1 and *@Version* is negative.

The **proc_FetchDocForRead** stored procedure MUST return a **Document Version Information and Content (Read) Result Set** (section 2.2.4.14) if *@GetContent* is set to 1 and *@Version* is not negative, with the UI version specified by *@Version*.

The **proc_FetchDocForRead** stored procedure MUST return a **Dirty Result Set** (section 2.2.4.9) if *@GetDirty* is set to 1.

The **proc_FetchDocForRead** stored procedure MUST return a **Link Info Single Doc Result Set** (section 2.2.4.29) if Link information was requested by setting *@GetMetaInfo* to 1.

The **proc_FetchDocForRead** stored procedure MUST return a **Document Metadata Result Set** (section 2.2.4.13) if the *@GetMetaInfo* parameter is set to 1 and *@Version* is negative. The **Document Metadata Result Set** MUST be returned and MUST contain a single row corresponding to the queried document.

The **proc_FetchDocForRead** stored procedure MUST return a **Document Version Metadata Result Set** (section 2.2.4.16) if *@GetMetaInfo* is set to 1 and *@Version* is not negative. If the document exists, the **Document Version Metadata Result Set** MUST contain one row. Otherwise, it MUST contain no rows.

The **proc_FetchDocForRead** stored procedure MUST return a **List Metadata Result Set** (section 2.2.4.31) if the *@bGetContainingList* parameter is set to 1 and the document is contained within an existing list (1). The **List Metadata Result Set** MUST contain one row.

The **proc_FetchDocForRead** stored procedure MUST return an **Empty List Result Set** (section 2.2.4.19) if the *@bGetContainingList* parameter is set to 1 and the document is not contained within an existing list (1). The **Empty List Result Set** contains a single, unnamed column with no rows to indicate that the document is not contained within a list (1).

The **proc_FetchDocForRead** stored procedure MUST return an **Attachment State Result Set** (section 2.2.4.5). The **Attachment State Result Set** MUST be returned and MUST contain one row.

3.1.5.12 proc_FetchDocForUpdate

The **proc_FetchDocForUpdate** stored procedure is called to request document content and metadata information. It also updates the CacheParseId flag for the requested document if the *@CacheParse* parameter is set to 1. **proc_FetchDocForUpdate** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_FetchDocForUpdate (
@DocSiteId                    uniqueidentifier
,@DocWebId                    uniqueidentifier
,@DocDirName                  nvarchar(256)
,@DocLeafName                 nvarchar(128)
,@UserId                      int
,@Version                     int
,@GetContent                  bit
,@GetWebListForNormalization  bit
,@CacheParse                  bit
,@GetWebPartInfo              bit
,@bGetContainingList          bit
,@LooksLikeAttachmentFile     bit
,@ChunkSize                   int
,@GetSandbox                  bit OUTPUT
);
```

@DocSiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the requested document.

@DocWebId: The **Site Identifier** (section 2.2.1.19) of the **site** containing the requested document.

@DocDirName: The directory name of the requested document.

@DocLeafName: The leaf name of the requested document.

@UserId: The **User Identifier** (section 2.2.1.24) for the current user who is requesting the information. This is used by **proc_FetchDocForUpdate** for permission checking.

@Version: Specifies the UI version number of the document being requested. A value of -1 specifies the most recent version of the document.

@GetContent: A bit flag specifying whether or not to return the content of the document. If this flag is set to 1 and *@Version* is negative, the server returns the **Document Information and Content (Update) Result Set** (section 2.2.4.12). If this flag is set to 1 and *@Version* is not negative, the server returns the **Document Version Information and Content Result Set** (section 2.2.4.13).

@GetWebListForNormalization: A bit flag specifying whether to return the subsite of the site specified by *@DocWebId*. If this flag is set to 1, **proc_FetchDocForUpdate** MUST return a list of child sites in the **Subsite List Result Set** (section 2.2.4.54).

@CacheParse: A bit flag specifying whether to update the CacheParseId flag of the requested document. If this flag is set to 1, **proc_FetchDocForUpdate** MUST update the CacheParseId flag for the requested document.

@GetWebPartInfo: A bit flag specifying whether to return the Web Parts information for the requested document. If this flag is set to 1, **proc_FetchDocForUpdate** MUST return the Web Parts information for the requested document in the **Web Part Info Result Set** (section 2.2.4.62) and the **Zone ID Result Set** (section 2.2.4.67).

@bGetContainingList: A bit flag specifies whether to return the **list (1)** containing the requested document. If this flag is set to 1, **proc_FetchDocForUpdate** MAY return the **List MetaData Result Set** (section 2.2.4.32) or **Empty List Result Set** (section 2.2.4.19).

@LooksLikeAttachmentFile: A bit flag specifies whether the requested document appears to the front-end Web server to be an attachment to a list item. If this flag is set to 1, **proc_FetchDocForUpdate** MUST determine whether the current user has sufficient permissions to retrieve the requested document as an attachment.

@ChunkSize: Specifies the maximum size, in bytes, of the document content image to be returned in the **Document Information and Content (Update) Result Set**. If the document content image size is larger than *@ChunkSize*, a single zero byte is returned as the document content image, and the front-end Web server can request the content of the document in a subsequent operation.

@GetSandbox: A bit output parameter. This parameter MUST be set to 1 if the value of *@Version* is -2 and the value of *@GetContent* is set to 0, else it is set to 0.

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

Result Sets:

The **proc_FetchDocForUpdate** stored procedure MUST return a **Subsite List Result Set** (section 2.2.4.54).

The **proc_FetchDocForUpdate** stored procedure MAY return a **Document Metadata Result Set** (section 2.2.4.13). The **Document Metadata Result Set** MUST be returned and MUST contain a single row corresponding to the checked-out or locked document.

The **proc_FetchDocForUpdate** stored procedure MUST return a **Document Version Metadata Result Set** (section 2.2.4.16) if *@Version* is not negative. If the document exists, the **Document Version Metadata Result Set** MUST contain one row. Otherwise, it MUST contain no rows.

The **proc_FetchDocForUpdate** stored procedure MUST return a **Link Info Single Doc Fixup Result Set** (section 2.2.4.28) if Link information was requested by setting *@CacheParse* to 1.

The **proc_FetchDocForUpdate** stored procedure MUST return a **Web Part Info Result Set** (section 2.2.4.62) if *@GetWebPartInfo* is set to 1. If Web Part data for the requested document exists, one row MUST be returned for each Web Part. Otherwise, zero rows MUST be returned.

The **proc_FetchDocForUpdate** stored procedure MUST return a **Zone ID Result Set** (section 2.2.4.67) if *@GetWebPartInfo* is set to 1. If Web Part data for the requested document exists, one row MUST be returned for each Web Part zone. Otherwise, zero rows MUST be returned.

The **proc_FetchDocForUpdate** stored procedure MUST return an **Empty List Result Set** (section 2.2.4.19) if the *@bGetContainingList* parameter is set to 1 and the document is not contained within a list (1). The **Empty List Result Set** contains a single, unnamed column with no rows to indicate that the document is not contained within a list (1).

The **proc_FetchDocForUpdate** stored procedure MUST return a **List MetaData Result Set** (section 2.2.4.32) if the *@bGetContainingList* parameter is set to 1 and the document is contained within an existing list (1). The **List Metadata Result Set** MUST contain one row.

The **proc_FetchDocForUpdate** stored procedure MUST return an **Attachment State Result Set** (section 2.2.4.5). The **Attachment State Result Set** MUST be returned and MUST contain one row.

The **proc_FetchDocForUpdate** stored procedure MUST return a **Document Information and Content (Update) Result Set** (section 2.2.4.12) if *@GetContent* is set to 1 and *@Version* is negative.

The **proc_FetchDocForUpdate** stored procedure MUST return a **Document Version Information and Content Result Set** (section 2.2.4.15) if *@GetContent* is set to 1 and *@Version* is not negative, with the UI version specified by *@Version*.

3.1.5.13 proc_FetchWelcomeNames

The **proc_FetchWelcomeNames** stored procedure is called to **list (1)** all the names of the default welcome pages used as redirection targets when folders are requested by an HTTP GET in all site collections in the back-end database server. **proc_FetchWelcomeNames** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_FetchWelcomeNames (
);
```

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

Result Sets: The **proc_FetchWelcomeNames** stored procedure MUST return a **Welcome Pages Result Set** (section 2.2.4.66).

3.1.5.14 proc_GenerateNextId

The **proc_GenerateNextId** stored procedure is called to obtain a unique **List Item Identifier** (section 2.2.1.11) for a new list item in the specified **list (1)**. It also increments the identifier by the specified value. **proc_GenerateNextId** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_GenerateNextId (
@WebId     uniqueidentifier
,@ListId   uniqueidentifier
,@NumIds   int = 1
);
```

@WebId: The **Site Identifier** (section 2.2.1.19) of the **site** containing the list (1).

@ListId: The **List Identifier** (section 2.2.1.10) of the list (1) from which to get the next identifier.

@NumIds: The value by which the identifier is incremented. This is an optional parameter. This value MUST be a positive integer. If no parameter is provided, a default value of 1 is used.

Return Values: The **proc_GenerateNextId** stored procedure MUST return an integer return code, which is the available **List Item Identifier** for the specified list (1). If the specified *@WebId* and *@ListId* values are not valid, the procedure attempts to return a status of NULL, which is not allowed. A status of 0 will be returned instead.

Result Sets: The **proc_GenerateNextId** stored procedure MUST NOT return any result set.

3.1.5.15 proc_GetAllAttachmentsInfo

The **proc_GetAllAttachmentsInfo** stored procedure is called to get attachment information. **proc_GetAllAttachmentsInfo** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_GetAllAttachmentsInfo (
@SiteID     uniqueidentifier
,@WebID     uniqueidentifier
,@ListID    uniqueidentifier
,@ItemID    int
);
```

@SiteID: The **Site Collection Identifier** (section 2.2.1.17) for the site collection.

@WebID: The **Site Identifier** (section 2.2.1.19) for the **site**.

@ListID: The **List Identifier** (section 2.2.1.10) for the **list (1)**.

@ItemID: The item identifier of the list item.

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

Result Sets:

The **proc_GetAllAttachementsInfo** stored procedure MUST return an **Attachment Item Information Result Set** (section 2.2.4.4) if *@ItemID* is set to -1.

The **proc_GetAllAttachementsInfo** stored procedure MUST return an **Attachment Document Information Result Set** (section 2.2.4.3) if *@ItemID* is not equal to -1.

3.1.5.16 proc_GetContainingList

The **proc_GetContainingList** stored procedure is called to get metadata and **event receiver** information about the list containing a specified URL. **proc_GetContainingList** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_GetContainingList (
@SiteId     uniqueidentifier
,@WebId     uniqueidentifier
,@Url       nvarchar(260)
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the **list (1)**.

@WebId: The **Site Identifier** (section 2.2.1.19) of the **site** containing the list (1).

@Url: The store-relative form URL of a list item or document within the list (1). The URL is used to derive the location of the containing list (1). The leaf name part of *@Url* is ignored because it could point to a nonexistent document or list item.

Return Values: The **proc_GetContainingList** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
1	List not found in the specified site.
3	List not found.

Result Sets:

The **proc_GetContainingList** stored procedure MUST return a **List MetaData Result Set** (section 2.2.4.32) if the document is contained within an existing list (1). The **List Metadata Result Set** MUST contain one row.

The **proc_GetContainingList** stored procedure MUST return an **Empty List Result Set** (section 2.2.4.19) if the document is not contained within a list (1).

3.1.5.17 proc_GetDocsMetaInfo

The **proc_GetDocsMetaInfo** stored procedure is called to request document metadata information for up to ten documents within a specified **site**. **proc_GetDocsMetaInfo** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_GetDocsMetaInfo (
@DocSiteId              uniqueidentifier
,@WebFullUrl            nvarchar(260)
,@GetDocsFlags          int
,@UserId                int
,@DirName1              nvarchar(256) = null
,@LeafName1             nvarchar(128) = null
,@AttachmentsFlag1      tinyint = null
,@DirName2              nvarchar(256) = null
,@LeafName2             nvarchar(128) = null
,@AttachmentsFlag2      tinyint = null
,@DirName3              nvarchar(256) = null
,@LeafName3             nvarchar(128) = null
,@AttachmentsFlag3      tinyint = null
,@DirName4              nvarchar(256) = null
,@LeafName4             nvarchar(128) = null
,@AttachmentsFlag4      tinyint = null
,@DirName5              nvarchar(256) = null
,@LeafName5             nvarchar(128) = null
,@AttachmentsFlag5      tinyint = null
,@DirName6              nvarchar(256) = null
,@LeafName6             nvarchar(128) = null
,@AttachmentsFlag6      tinyint = null
,@DirName7              nvarchar(256) = null
,@LeafName7             nvarchar(128) = null
,@AttachmentsFlag7      tinyint = null
,@DirName8              nvarchar(256) = null
,@LeafName8             nvarchar(128) = null
```

```
,@AttachmentsFlag8       tinyint = null
,@DirName9               nvarchar(256) = null
,@LeafName9              nvarchar(128) = null
,@AttachmentsFlag9       tinyint = null
,@DirName10              nvarchar(256) = null
,@LeafName10             nvarchar(128) = null
,@AttachmentsFlag10      tinyint = null
);
```

@DocSiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection containing the specified documents.

@WebFullUrl: The store-relative form URL of the site containing the specified documents.

@GetDocsFlags: A bit mask with a flag that specifies whether to return link information. If this parameter has the bit 0x00000020 set, then link information MUST be returned in the **Single Doc Link Information Result Set** (section 2.2.4.44). If this parameter has the bit 0x00004000 set, then the result sets are sorted according to the identifier of the document.

@UserId: The **User Identifier** (section 2.2.1.24) for the current user who is requesting the information. This MUST be used by **proc_GetDocsMetaInfo** for permission checking.

@DirName1: The directory name of the specified document. A NULL value signifies that no document is being fetched and the next two parameters MUST be ignored.

@LeafName1: The leaf name of the specified document. If *@DirName1* is not NULL, this MUST NOT be NULL.

@AttachmentsFlag1: An **Attachments Flag** (section 2.2.2.1) value which specifies the type of security checks to be performed by **proc_GetDocsMetaInfo**, based on whether it appears to be an attachment.

@DirName2: The directory name of the specified document. A NULL value signifies that no document is being fetched and the next two parameters MUST be ignored.

@LeafName2: The leaf name of the specified document. If *@DirName2* is not NULL, this MUST NOT be NULL.

@AttachmentsFlag2: An **Attachments Flag** value that specifies the type of security checks to be performed by **proc_GetDocsMetaInfo**, based on whether it appears to be an attachment.

@DirName3: The directory name of the specified document. A NULL value signifies that no document is being fetched and the next two parameters MUST be ignored.

@LeafName3: The leaf name of the specified document. If *@DirName3* is not NULL, this MUST NOT be NULL.

@AttachmentsFlag3: An **Attachments Flag** value that specifies the type of security checks to be performed by **proc_GetDocsMetaInfo**, based on whether it appears to be an attachment.

@DirName4: The directory name of the specified document. A NULL value signifies that no document is being fetched, and the next two parameters MUST be ignored.

@LeafName4: The leaf name of the specified document. If *@DirName4* is not NULL, this MUST NOT be NULL.

@AttachmentsFlag4: An **Attachments Flag** value that specifies the type of security checks to be performed by **proc_GetDocsMetaInfo**, based on whether it appears to be an attachment.

@DirName5: The directory name of the specified document. A NULL value signifies that no document is being fetched and the next two parameters MUST be ignored.

@LeafName5: The leaf name of the specified document. If *@DirName5* is not NULL, this MUST NOT be NULL.

@AttachmentsFlag5: An **Attachments Flag** value that specifies the type of security checks to be performed by **proc_GetDocsMetaInfo**, based on whether it appears to be an attachment.

@DirName6: The directory name of the specified document. A NULL value signifies that no document is being fetched and the next two parameters MUST be ignored.

@LeafName6: The leaf name of the specified document. If *@DirName6* is not NULL, this MUST NOT be NULL.

@AttachmentsFlag6: An **Attachments Flag** value that specifies the type of security checks to be performed by **proc_GetDocsMetaInfo**, based on whether it appears to be an attachment.

@DirName7: The directory name of the specified document. A NULL value signifies that no document is being fetched and the next two parameters MUST be ignored.

@LeafName7: The leaf name of the specified document. If *@DirName7* is not NULL, this MUST NOT be NULL.

@AttachmentsFlag7: An **Attachments Flag** value which specifies the type of security checks to be performed by **proc_GetDocsMetaInfo**, based on whether it appears to be an attachment.

@DirName8: The directory name of the specified document. A NULL value signifies that no document is being fetched and the next two parameters MUST be ignored.

@LeafName8: The leaf name of the specified document. If *@DirName8* is not NULL, this MUST NOT be NULL.

@AttachmentsFlag8: An **Attachments Flag** value that specifies the type of security checks to be performed by **proc_GetDocsMetaInfo**, based on whether it appears to be an attachment.

@DirName9: The directory name of the specified document. A NULL value signifies that no document is being fetched and the next two parameters MUST be ignored.

@LeafName9: The leaf name of the specified document. If *@DirName9* is not NULL, this MUST NOT be NULL.

@AttachmentsFlag9: An **Attachments Flag** value that specifies the type of security checks to be performed by **proc_GetDocsMetaInfo**, based on whether it appears to be an attachment.

@DirName10: The directory name of the specified document. A NULL value signifies that no document is being fetched and the next two parameters MUST be ignored.

@LeafName10: The leaf name of the specified document. If *@DirName10* is not NULL, this MUST NOT be NULL.

@AttachmentsFlag10: An **Attachments Flag** value which specifies the type of security checks to be performed by **proc_GetDocsMetaInfo**, based on whether it appears to be an attachment.

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

Result Sets:

The **proc_GetDocsMetaInfo** stored procedure MUST return a **Subsite List Result Set** (section 2.2.4.54) for all subsites whose **parent site** is specified in the *@WebFullUrl* parameter, and MUST contain no rows if there are no such subsites.

The **proc_GetDocsMetaInfo** stored procedure MUST return a **Single Doc Link Information Result Set** (section 2.2.4.44) if Link information was requested by the *@GetDocsFlags* parameter.

The **proc_GetDocsMetaInfo** stored procedure MUST return a **Multiple Document Metadata Result Set** (section 2.2.4.36). The **Multiple Document Metadata Result Set** MUST contain one row for each document where the corresponding *@DirName* parameter was not set to NULL.

The **proc_GetDocsMetaInfo** stored procedure MUST return a **Null Individual URL Security Result Set** (section 2.2.4.37) for each document whose *@Dirname* is not NULL and if the current user does not have permission to see that document's metadata.

The **proc_GetDocsMetaInfo** stored procedure MUST return an **Individual URL Security Result Set** (section 2.2.4.25) for each document whose *@Dirname* is not NULL and if the current user has permission to see this document's metadata.

3.1.5.18 proc_getGlobals

The **proc_getGlobals** stored procedure is called to get a record of global settings that apply across WSS deployment. **proc_getGlobals** is defined using T-SQL syntax, as follows.

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

@Id: The **Global Identifier** (section 2.2.1.5) of the global settings record to return.

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

Result Sets: The **proc_getGlobals** stored procedure MUST return a **Globals Result Set** (section 2.2.4.21).

3.1.5.19 proc_GetLinkInfoSingleDoc

The **proc_GetLinkInfoSingleDoc** stored procedure is called to obtain link information and status for all forward links within a specified document and for all backward links within the specified site collection to the document. **proc_GetLinkInfoSingleDoc** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_GetLinkInfoSingleDoc (
@DocSiteId     uniqueidentifier
,@DocDirName   nvarchar(256)
,@DocLeafName  nvarchar(128)
);
```

@DocSiteId: The **Site Collection Identifier** (section 2.2.1.17) for a site collection containing the document.

@DocDirName: The directory name of the document.

@DocLeafName: The leaf name of the document.

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

Result Sets: The **proc_GetLinkInfoSingleDoc** stored procedure MUST return a **Link Info Single Doc Result Set** (section 2.2.4.29).

3.1.5.20 proc_GetListFields

The **proc_GetListFields** stored procedure is called to get the mapping of fields in a list. The mapping is represented via an XML string specifying each of the fields in the **list (1)**. **proc_GetListFields** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc GetListFields (
@WebId     uniqueidentifier
,@ListId   uniqueidentifier
);
```

@WebId: The **Site Identifier** (section 2.2.1.19) of the **site** containing the list (1).

@ListId: The **List Identifier** (section 2.2.1.10) of the list (1).

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

Result Sets: The **proc_GetListFields** stored procedure MUST return a **Fields Information Result Set** (section 2.2.4.20). The **Fields Information Result Set** will be returned once, and MUST contain 0 or 1 row as follows: if no list was found matching the provided *@WebId* and *@ListId* parameters, the result set MUST contain 0 rows. Otherwise, the result set MUST contain one row.

3.1.5.21 proc_GetListRequestAccess

The **proc_GetListRequestAccess** stored procedure is called to get request access information for a list. **proc_GetListRequestAccess** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_GetListRequestAccess (
@WebID     uniqueidentifier
,@ListID   uniqueidentifier
);
```

@WebID: The **Site Identifier** (section 2.2.1.19) of the **site** containing the **list (1)**.

@ListID: The **List identifier** (section 2.2.1.10) of the list (1).

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

Result Sets: The **proc_GetListRequestAccess** stored procedure MUST return a **List Access Result Set** (section 2.2.4.30).

3.1.5.22 proc_getServerById

The **proc_getServerById** stored procedure is called to return an individual server record by identifier for the physical servers. **proc_getServerById** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_getServerById (
@ServerId uniqueidentifier
);
```

@ServerId: The **Server Identifier** (section 2.2.1.16) simple type of the server record to return.

Return Values: The **proc_getServerById** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
1	Execution failed.

Result Sets: The **proc_getServerById** stored procedure MUST return a **Server Information Result Set** (section 2.2.4.42).

3.1.5.23 proc_GetSiteFlags

The **proc_GetSiteFlags** stored procedure is called to return the **Site Collection Flags** (section 2.2.2.7) set for a specified site collection. **proc_GetSiteFlags** is defined using T-SQL syntax, as follows.

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

@WebSiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection.

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

Result Sets: The **proc_GetSiteFlags** stored procedure MUST return a **Site Collection Flags Result Set** (section 2.2.4.47). The **Site Collection Flags Result Set** will be NULL if the input *@WebSiteId* is invalid.

3.1.5.24 proc_GetTpWebMetaDataAndListMetaData

The **proc_GetTpWebMetaDataAndListMetaData** stored procedure is called to retrieve metadata for a particular **site** or list. **proc_GetTpWebMetaDataAndListMetaData** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_GetTpWebMetaDataAndListMetaData (
@WebSiteId          uniqueidentifier
,@Url               nvarchar(260)
,@ListId            uniqueidentifier
,@FetchWebParts     bit
,@RunUrlToWebUrl    bit
,@SystemId          tSystemID = null
);
```

@WebSiteId: The **Site Collection Identifier** (section 2.2.1.17) for a site collection containing the site.

@Url: The store-relative form URL of the site or **list (1)** to retrieve metadata for.

@ListId: The **List Identifier** (section 2.2.1.10) of the list (1) for which metadata is requested. A NULL value signifies to retrieve metadata for a site.

@FetchWebParts: A bit flag specifying whether to return the Web Parts. If it is set to 1, **proc_GetTpWebMetaDataAndListMetaData** MUST return the **List Webpart Result Set** (section 2.2.4.34).

@RunUrlToWebUrl: A bit flag specifying whether the *@Url* parameter is for a site or for a list (1) within a site. If this value is set to 1, the value in *@Url* is for a list and MUST be converted to a store-relative form URL for the containing site. If this value is set to 0, the value in *@Url* MUST be for a site.

@SystemId: The **SystemID** (section 2.2.1.20) of the current user requesting the metadata.

Return Values: The **proc_GetTpWebMetaDataAndListMetaData** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
3	The specified URL was not found.
1168	The site collection specified by *@WebSiteId* was not found.
1271	The site is locked.

Result Sets:

The **proc_GetTpWebMetaDataAndListMetaData** stored procedure MUST return a **List Webpart Result Set** if *@FetchWebParts* is set to 1, to retrieve metadata for particular list (1).

The **proc_GetTpWebMetaDataAndListMetaData** stored procedure MUST return a **List MetaData Result Set** (section 2.2.4.32) if *@ListId* is NOT NULL.

The **proc_GetTpWebMetaDataAndListMetaData** stored procedure MUST return a **Site Metadata Result Set** (section 2.2.4.51) on successful execution.

The **proc_GetTpWebMetaDataAndListMetaData** stored procedure MUST return a **Web Url Result Set** (section 2.2.4.65) if *@RunUrlToWebUrl* = 1.

3.1.5.25 proc_GetWebMetainfo

The **proc_GetWebMetainfo** stored procedure is called to request metadata information for a **site**. **proc_GetWebMetainfo** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_GetWebMetainfo (
@WebSiteId     uniqueidentifier
,@WebDirName   nvarchar(256)
,@WebLeafName  nvarchar(128)
,@SystemId     tSystemID = null
);
```

@WebSiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection containing the site whose metadata is requested.

@WebDirName: The directory name part of the site location.

@WebLeafName: The leaf name part of the site location.

@SystemId: The **SystemID** (section 2.2.1.20) of the current user requesting the site metadata information.

Return Values: The **proc_GetWebMetainfo** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
3	A site was not found at the specified location.
1271	The site is locked.

Result Sets:

The **proc_GetWebMetainfo** stored procedure MUST return a **Site Category Result Set** (section 2.2.4.46). The **Site Category Result Set** MUST be returned and MUST contain one row for each category specified for the site. If the specified site is not found, zero rows MUST be returned.

The **proc_GetWebMetainfo** stored procedure MUST return a **Site Metainfo Result Set** (section 2.2.4.52).

The **proc_GetWebMetainfo** stored procedure MUST return a **Site Metadata Result Set** if the site specified by *@WebSiteId* exists in the specified location.

3.1.5.26 proc_GetWebMetainfoByUrl

The **proc_GetWebMetainfoByUrl** stored procedure is called to request metadata information for a **site**. **proc_GetWebMetainfoByUrl** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_GetWebMetainfoByUrl (
@WebSiteId       uniqueidentifier
,@Url            nvarchar(260)
,@SystemId       tSystemID = null
);
```

@WebSiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection containing the site whose metadata is requested.

@Url: A complete store-relative form URL for the site or for a document within the site.

@SystemId: The **SystemID** (section 2.2.1.20) of the current user requesting the site metadata information.

Return Values: The **proc_GetWebMetainfoByUrl** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
3	The specified site was not found.
1168	The site collection does not exist.

Result Sets:

The **proc_GetWebMetainfoByUrl** stored procedure MUST return a **Site Metadata Result Set** (section 2.2.4.51). The **Site Metadata Result Set** MUST be returned for the site that is contained in an existing site collection.

The **proc_GetWebMetainfoByUrl** stored procedure MUST return a **Site Category Result Set** (section 2.2.4.46). The **Site Category Result Set** MUST be returned for the site that is contained in an existing site collection.

The **proc_GetWebMetainfoByUrl** stored procedure MUST return a **Site Metainfo Result Set** (section 2.2.4.52). The **Site Metainfo Result Set** MUST be returned for the site that is contained in an existing site collection.

The **proc_GetWebMetainfoByUrl** stored procedure MUST return a **Site URL Result Set** (section 2.2.4.53). The **Site URL Result Set** MUST be returned for the site that is contained in an existing site collection and MUST contain a single row.

3.1.5.27 proc_ListDocumentVersions

The **proc_ListDocumentVersions** stored procedure is called to **list (1)** all available version history information for a specified document. **proc_ListDocumentVersions** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_ListDocumentVersions (
@DocSiteId       uniqueidentifier
,@DocWebId       uniqueidentifier
,@DocDirName     nvarchar(256)
,@DocLeafName    nvarchar(128)
,@UserId         int
);
```

@DocSiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the document.

@DocWebId: The **Site Identifier** (section 2.2.1.19) of the **site** containing the document.

@DocDirName: The directory name of the directory containing the document.

@DocLeafName: The leaf name of the document.

@UserId: The **User Identifier** (section 2.2.1.24) of the current user.

Return Values: The **proc_ListDocumentVersions** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
2	The document specified is not a file.

Result Sets:

The **proc_ListDocumentVersions** stored procedure MUST return a **Document Versions Result Set** (section 2.2.4.17) if the specified document has a **Document Store** type (section 2.2.2.3) of 0, indicating that it is a file.

The **proc_ListDocumentVersions** stored procedure MUST return an **Individual URL Security Result Set** (section 2.2.4.25) if the document is contained within a list (1) or document library and MUST contain a single row.

The **proc_ListDocumentVersions** stored procedure MUST return a NULL **Individual URL Security Result Set** (section 2.2.4.37) if the document is not contained within a list (1) or document library and MUST return a single row.

3.1.5.28 proc_ListUrls

The **proc_ListUrls** stored procedure is called to get metadata for a document specified by a URL and the documents contained within it, if any. **proc_ListUrls** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc ListUrls (
@DirSiteId              uniqueidentifier
,@DirWebId              uniqueidentifier
,@DirFullUrl            nvarchar(260)
,@AttachmentsFlag       tinyint
,@ClientTimeStamp       datetime
,@FetchLinkInfo         bit
,@IncludeThicketDirs    bit
,@UserId                int
);
```

@DirSiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the document specified by a URL.

@DirWebId: The **Site Identifier** (section 2.2.1.19) of the site containing the document.

@DirFullUrl: The store-relative form URL specifying the document.

@AttachmentsFlag: An **Attachments Flag** (section 2.2.2.1) value specifying whether the document is a file or an attachments folder.

@ClientTimeStamp: A datetime that specifies a limiting time on the data in the result sets returned.

@FetchLinkInfo: A bit flag specifying whether to return the **Link Info Result Set** (section 2.2.4.27). If this parameter is set to 1 and the specified document is a folder, the **Link Info Result Set** MUST be returned.

@IncludeThicketDirs: A bit flag specifying whether to return **thicket folders** in the **Contained Document Metadata Result Set** (section 2.2.4.7). If this parameter is set to 1 and the specified document is a folder, any thicket folders MUST be included in the **Contained Document Metadata Result Set**.

@UserId: The **User Identifier** (section 2.2.1.24) for the current user.

Return Values: The **proc_ListUrls** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
3	The document URL is not valid, or the document is not a folder or **site**.

Result Sets:

The **proc_ListUrls** stored procedure MUST return a **Link Info Result Set** that contains information about all forward links from and backward links to the documents contained within the specified document. If the *@FetchLinkInfo* parameter is not set to 1, the **Link Info Result Set** MUST NOT be returned. Otherwise, if the *@FetchLinkInfo* parameter is set to 1, the **Link Info Result Set** MUST be returned. The **Link Info Result Set** MUST contain one row for each link that has been modified after the value in the *@ClientTimeStamp* parameter for each contained document in the site collection that has its directory name equal to the value of the *@DirFullUrl* parameter.

The **proc_ListUrls** stored procedure MUST return an **Individual URL Security Result Set** (section 2.2.4.25). The **Individual URL Security Result Set** MUST only be returned if the document is

contained within a **list (1)** or document library. Otherwise, the **NULL Individual URL Security Result Set** (section 2.2.4.37) MUST be returned instead. If returned, the **Individual URL Security Result Set** MUST contain a single row.

The **proc_ListUrls** stored procedure MUST return a **NULL Individual URL Security Result Set** that indicates that a specified document is not found. The **NULL Individual URL Security Result Set** MUST return a single row.

The **proc_ListUrls** stored procedure MUST return a **Subsite List Result Set** (section 2.2.4.54) that contains an ordered list of store-relative form URLs for all subsites whose **parent site** is the site specified by the *@DirWebId* parameter. If the specified document is not a folder, the **Subsite List Result Set** MUST NOT be returned. Otherwise, the **Subsite List Result Set** MUST be returned and MUST contain one row for each subsite within the specified site, and it MUST contain no rows if there are no such subsites.

The **proc_ListUrls** stored procedure MUST return a **Document Metadata Result Set** (section 2.2.4.13) that contains the metadata for the specified document. If the *@DirFullUrl* parameter contains an **empty string**, the **Document Metadata Result Set** MUST NOT be returned. Otherwise, the **Document Metadata Result Set** MUST contain one row with the metadata information for the document.

The **proc_ListUrls** stored procedure MUST return a **Contained Document Metadata Result Set** (section 2.2.4.7) that contains the metadata information for the documents contained within the specified document. If the specified document is not a folder, the **Contained Document Metadata Result Set** MUST NOT be returned. Otherwise, the **Contained Document Metadata Result Set** MUST return one row for each contained document in the site collection that has its directory name equal to the value of the *@DirFullUrl* parameter. The contained document MUST be a file or its **ThicketFlag** value MUST NOT be NULL.

The **proc_ListUrls** stored procedure MUST return a **Server Time Result Set** (section 2.2.4.43) that contains the current time from the back-end database server, in UTC. The **Server Time Result Set** MUST be returned and MUST contain a single row of data.

3.1.5.29 proc_putGlobals

The **proc_putGlobals** stored procedure is called to add or update a global settings record that applies across WSS deployment. **proc_putGlobals** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_putGlobals (
@Id                    uniqueidentifier
,@User                 nvarchar(255)
,@SchemaVersion        nvarchar(64)
,@UseHostHeader        bit
,@UseNtAuthForDatabase bit
,@SmtpServiceId        uniqueidentifier = null
,@MailCodePage         int = null
,@FromAddress          nvarchar(255) = null
,@ReplyToAddress       nvarchar(255) = null
,@Status               int
,@Version              timestamp = null
,@Properties           ntext
,@NewVersion           timestamp OUTPUT
,@NewId                uniqueidentifier OUTPUT
);
```

@Id: The **Global Identifier** (section 2.2.1.5) used to insert or update the global settings record.

@User: The user name of the person making modification in the global settings record.

@SchemaVersion: The schema version of the configuration database.

@UseHostHeader: A bit set to 1, if the **host header** mode is being used for WSS.

@UseNtAuthForDatabase: A bit set to 1 if Integrated Windows authentication is being used for database access. Otherwise, the bit is set to 0 if SQL authentication is being used.

@SmtpServiceId: The GUID for the Simple Mail Transfer Protocol (SMTP) Service specified in the global settings and is an implementation-specific capability.

@MailCodePage: An option selected for **character set** in the SMTP Settings specified in the global settings and is an implementation-specific capability.

@FromAddress: An email address identified in the From field of the SMTP settings specified in the global settings and is an implementation-specific capability.

@ReplyToAddress: An email address that was placed in the Reply To field of the SMTP settings specified in the global settings and is an implementation-specific capability.

@Status: The value of the configuration status.

@Version: The version value that is incremented when the UseHostHeader or SMTP Server Settings change.

@Properties: Miscellaneous properties for the deployment.

@NewVersion: An output parameter that contains the version of the newly inserted or updated row.

@NewId: An output parameter; it is not used.

Return Values: The **proc_putGlobals** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
1	Insert failed for the specified global setting record.
4	Update failed for the specified global setting record.
5	Unable to retrieve new version for the specified global setting record.
50002	Insert failed because the version is not NULL for the specified global setting record.
50003	Update failed because the current version conflicts with the specified global setting record.

Result Sets: The **proc_putGlobals** stored procedure MUST NOT return any result set.

3.1.5.30 proc_RenameUrl

The **proc_RenameUrl** stored procedure is called to rename a document or a folder within a specified **site**. **proc_RenameUrl** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_RenameUrl (
@SiteId                  uniqueidentifier
,@SubWebId               uniqueidentifier
,@OldUrl                 nvarchar(260)
,@NewUrl                 nvarchar(260)
,@UserId                 int
,@RenameFlags            int = 0
,@PutFlags               int = 0
,@ReturnFlags            int = 0
,@AttachmentOpOldUrl     int = 3
```

```
,@AttachmentOpNewUrl      int = 3
,@ParseDocsNow            tinyint = null OUTPUT
,@FailedUrl               nvarchar(260) = null OUTPUT
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the document.

@SubWebId: The **Site Identifier** (section 2.2.1.19) of the subsite containing the document.

@OldUrl: The store-relative URL of the document to be moved.

@NewUrl: The new store-relative URL of the document.

@UserId: The **User Identifier** (section 2.2.1.24) of the current user requesting the move.

@RenameFlags: A bit field that determines document rename options. This can have one or more flags set. The only valid values of the *@RenameFlags* bits are specified in **Rename Flags** (section 2.2.2.6).

@PutFlags: A bit field that determines document change options. This can have one or more flags set. The only valid values of the *@PutFlags* bits are specified in **Put Flags** type (section 2.2.2.5).

@ReturnFlags: A bit field that determines the type of information requested. This can have one or more flags set. Valid values for this parameter are listed in the following table.

Value	Description
0x01	Return information about the moved documents.
0x02	Return information about backward links referencing the moved documents.

@AttachmentOpOldUrl: An **Attachments Flag** (section 2.2.2.1) value for the document URL.

@AttachmentOpNewUrl: An **Attachments Flag** value for the destination URL.

@ParseDocsNow: An output parameter. If it is set to 1, it indicates that the moved document MUST be parsed again.

@FailedUrl: An output parameter indicating the URL at which the delete operation failed. This parameter MUST be set to NULL if the deletion was successful.

Return Values: The **proc_RenameUrl** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
2	File not found: Parent of target URL not found.
3	Item not found at specified URL, site collection, and site.
15	Cannot move to, or from, a forms folder in a **list (1)** or document library.
33	Cannot move a folder that contains checked-out documents.
50	The document is a site, but *@RenameFlags* does not indicate that a site move is requested.
51	Cannot move to, or from, a forms folder in a list (1) or document library.

Value	Description
80	Target URL is a document with a **Document Store Type** (section 2.2.2.3) of 0 (File), but the *@PutFlags* value does not indicate a request to overwrite it.
87	Bad parameter: No row was updated.
130	Cannot move to, or from, an image or a thumbnail file in an Image Library.
138	Cannot move folder across lists.
144	Not an overwrite request, and URL is a directory or web, or other invalid combination.
161	Cannot move folder across sites.
190	Cannot move folder containing thicket.
206	Target URL is too long.
214	Cannot move thicket.
1150	Concurrency violation.
8398	There was an error moving a list (1). At least one list (1) could not be deleted.

Result Sets:

The **proc_RenameUrl** stored procedure MUST return a **Rename Result Set** (section 2.2.4.40) if successful and if bit 0x01 of *@ReturnFlags* is set.

The **proc_RenameUrl** stored procedure MUST return the **Backward Link Result Set** (section 2.2.4.6) if successful and if bit 0x2 is set in *@ReturnFlags* and bit 0x4 is set in *@RenameFlags*.

3.1.5.31 proc_SecAddPrincipalToWebGroup

The **proc_SecAddPrincipalToWebGroup** stored procedure is called to add a **principal (1)** to site group. **proc_SecAddPrincipalToWebGroup** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecAddPrincipalToWebGroup (
@SiteId          uniqueidentifier
,@WebId          uniqueidentifier
,@GroupId        int
,@UserId         int
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection.

@WebId: The **Site Identifier** (section 2.2.1.19) for the **site**.

@GroupId: The identifier of the site group.

@UserId: The **User Identifier** (section 2.2.1.24) of the principal (1) to be added.

Return Values: The **proc_SecAddPrincipalToWebGroup** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.

Value	Description
3	Web group cannot be found.

Result Sets: The **proc_SecAddPrincipalToWebGroup** stored procedure MUST NOT return any result set.

3.1.5.32 proc_SecAddUser

The **proc_SecAddUser** stored procedure is called to add an entry for a **principal (1)** (user or domain group) to the list of user information stored in the back-end database server. **proc_SecAddUser** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecAddUser (
@SiteId                     uniqueidentifier
,@SystemId                  tSystemID
,@IsDomainGroup             bit
,@Login                     nvarchar(255)
,@Title                     nvarchar(255)
,@Email                     nvarchar(255)
,@Notes                     nvarchar(1023)
,@WebIdForGuestGroup        uniqueidentifier = null
,@IncrementUserCount        bit = 0
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection to associate with the principal (1).

@SystemId: The **SystemId** (section 2.2.1.20) of the principal (1) to be added or updated. If a user exists with the specified SystemID, its record will be updated. Otherwise, a new user will be added with the specified SystemID. SystemId is of type **tSystemID**, where **tSystemID** is varbinary(128).

@IsDomainGroup: A bit flag specifying whether the principal (1) to be added is a domain group. If this is set to 1, the principal (1) being added is a domain group. A value of 0 indicates that this is a user. This parameter MUST NOT be NULL.

@Login: The login name of the principal (1) to be added. This parameter MUST NOT be NULL.

@Title: The display name of the principal (1) to be added. This parameter MUST NOT be NULL.

@Email: The email address of the principal (1) to be added. This parameter MUST NOT be NULL.

@Notes: A text string containing notes about the principal (1) to be added. This parameter MUST NOT be NULL.

@WebIdForGuestGroup: The guest group to add this user to. If this is NULL, the user will not be added to any guest group.

@IncrementUserCount: A bit flag specifying whether to increment the user count of the site collection. When this parameter is set to 1, **proc_SecAddUser** MUST increment the site collection user count.

Return Values: The **proc_SecAddUser** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.

Value	Description
80	A principal (1) with the same login name but a different SystemID already has an entry in the user information.
1816	User Quota is exceeded and no additional users can be added to the **site**.

Result Sets: The **proc_SecAddUser** stored procedure MUST return a **User Identifier Result Set** (section 2.2.4.58).

3.1.5.33 proc_SecAddUserToSiteGroup

The **proc_SecAddUserToSiteGroup** stored procedure is called to add a user to a site group in the site collection. The user is added to the site group, the security version of the **site** is updated, and the change is logged. **proc_SecAddUserToSiteGroup** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecAddUserToSiteGroup (
@SiteId               uniqueidentifier
,@GroupId             int
,@UserIDToBeAdded     int
,@UserID              int
,@SiteAdmin           bit
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) for a site collection that will be queried for the requested group.

@GroupId: An identifier for the site group that the user (specified by *@UserIDToBeAdded*) refers to. *@GroupId* MUST refer to an existing group.

@UserIDToBeAdded: A **User Identifier** (section 2.2.1.24) for the user who is being added to the specified **group**. This value MUST refer to an existing **User Identifier** for the specified site collection.

@UserID: The **User Identifier** for the current user who is adding the user to the specified group. This value MUST refer to an existing **User Identifier** for the specified site collection.

@SiteAdmin: If 1, the user is a site administrator. Otherwise, the user is not a site administrator.

Return Values: The **proc_SecAddUserToSiteGroup** stored procedure MUST return one of the following integer return codes.

Value	Description
0	User was successfully added or user already belongs to site group.
5	The user specified by *@UserID* does not have rights to modify Group membership of the Group specified by @GroupID.

Result Sets: The **proc_SecAddUserToSiteGroup** stored procedure MUST NOT return any result set.

3.1.5.34 proc_SecChangeToInheritedList

The **proc_SecChangeToInheritedList** stored procedure is called to change the scope of the specified **list (1)** to the specified **site** and remove any **role assignments** and permission settings specific to the list. **proc_SecChangeToInheritedList** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecChangeToInheritedList (
@WebId     uniqueidentifier
```

```
,@ListId   uniqueidentifier
);
```

@WebId: The **Site Identifier** (section 2.2.1.19) of the site containing the list (1).

@ListId: The **List Identifier** (section 2.2.1.10) of the list (1).

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

Result Sets: The **proc_SecChangeToInheritedList** stored procedure MUST NOT return any result set.

3.1.5.35 proc_SecChangeToInheritedWeb

The **proc_SecChangeToInheritedWeb** stored procedure changes a **site** from having its own unique permissions to instead use the permissions inherited from its nearest ancestor with unique permissions. When a site is changed to have inherited permissions, all role definitions and fine-grained permissions on its lists and document libraries are reset. **proc_SecChangeToInheritedWeb** is defined using T-SQL syntax, as follows.

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

@WebId: The **Site Identifier** (section 2.2.1.19) for the site that is to start using inherited permissions.

Return Values: The **proc_SecChangeToInheritedWeb** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
3	The site was not found, the site does not have unique permissions, or an ancestor site with unique permissions was not found.

Result Sets: The **proc_SecChangeToInheritedWeb** stored procedure MUST return an ACL and **Permission Result Set** (section 2.2.4.2) if **proc_SecChangeToInheritedWeb** executes successfully.

3.1.5.36 proc_SecChangeToUniqueWeb

The **proc_SecChangeToUniqueWeb** stored procedure is called to update the group membership token or to add a new **member** to the specified **site**. **proc_SecChangeToUniqueWeb** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc SecChangeToUniqueWeb (
@WebId     uniqueidentifier
,@UserId   int
);
```

@WebId: The **Site Identifier** (section 2.2.1.19) of the site for which web membership is to be updated.

@UserId: The **User Identifier** (section 2.2.1.24) for the current user.

Return Values: The **proc_SecChangeToUniqueWeb** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
3	*@WebId* is invalid

Result Sets: The **proc_SecChangeToUniqueWeb** stored procedure MUST NOT return any result set.

3.1.5.37 proc_SecCheckDeletedAccounts

The **proc_SecCheckDeletedAccounts** stored procedure is called to check whether a login name exists in the site collection. **proc_SecCheckDeletedAccounts** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecCheckDeletedAccounts (
@SiteId uniqueidentifier
,@Login nvarchar(255)
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection.

@Login: The login name of the **principal (1)** as specified by the authentication in use. This parameter MUST NOT be NULL.

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

Result Sets: The **proc_SecCheckDeletedAccounts** stored procedure MUST return a **Login Result Set** (section 2.2.4.35) with login information, if a valid login name is found in the site collection. Otherwise, zero rows are returned.

3.1.5.38 proc_SecCheckSiteGroupExistence

The **proc_SecCheckSiteGroupExistence** stored procedure is called to determine the existence of a specific site group by its name. **proc_SecCheckSiteGroupExistence** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecCheckSiteGroupExistence (
@SiteId               uniqueidentifier
,@WebIdForGuestGroup  uniqueidentifier
,@Title               nvarchar(255)
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the specified **site**.

@WebIdForGuestGroup: The **Site Identifier** (section 2.2.1.19) of the site that is the member of the guest cross-site group.

@Title: The display name of the site group. This parameter MUST NOT be NULL.

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

Result Sets: The **proc_SecCheckSiteGroupExistence** stored procedure MUST return a **Site Group Existence Result Set** (section 2.2.4.48).

3.1.5.39 proc_SecCreateSiteGroup

The **proc_SecCreateSiteGroup** stored procedure is called to add a new sitegroup to a site collection. **proc_SecCreateSiteGroup** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecCreateSiteGroup (
@SiteId          uniqueidentifier
,@Title          nvarchar(255)
,@Description    nvarchar(512)
,@OwnerID        int
,@OwnerIsUser    bit
,@FirstMemberId  int
,@GroupID        int OUTPUT
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the new site group to be created.

@Title: The title of the new site group. This parameter MUST NOT be NULL.

@Description: A description of the new site group.

@OwnerID: A **User Identifier** (section 2.2.1.24) or **Site Group Identifier** (section 2.2.1.18) for the new sitegroup owner.

@OwnerIsUser: A bit flag specifying whether the site group owner specified by *@OwnerID* is a user or a site group. When *@OwnerIsUser* is set to 1, the new sitegroup owner is a user in the site collection. When *@OwnerIsUser* is set to 0, the owner is a sitegroup.

@FirstMemberId: A **User Identifier** for the first member of the new site group. This value MUST correspond to an existing user ID or be NULL. If this value is NULL, then the site group MUST be created with no **members**.

@GroupID: An output parameter holding the integer **Site Group Identifier** for the newly created or existing site group.

Return Values: The **proc_SecCreateSiteGroup** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
80	Failed to create the site group because a site group with an identical title exists in the site collection.
1816	Group count quota is exceeded and no additional group can be added.

Result Sets: The **proc_SecCreateSiteGroup** stored procedure MUST NOT return any result set.

3.1.5.40 proc_SecCreateWebGroup

The **proc_SecCreateWebGroup** stored procedure is called to add a new site group to a particular **site**. **proc_SecCreateWebGroup** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecCreateWebGroup (
@SiteId          uniqueidentifier
,@WebId          uniqueidentifier
,@Title          nvarchar(255)
,@Description    nvarchar(512)
,@Type           tinyint
,@IdToCreate     int
,@WebGroupId     int = null OUTPUT
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection.

@WebId: The **Site Identifier** (section 2.2.1.19) of the site for which a site group is to be created.

@Title: The title of the new site group. This parameter MUST NOT be NULL.

@Description: A description of the new site group.

@Type: The value for the newly created site group.

@IdToCreate: *@IdToCreate* contains an integer value.

@WebGroupId: A new site group identifier.

Return Values: The **proc_SecCreateWebGroup** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
80	Failed to create the site group because a site group with an identical title exists in the site collection.
1816	Quota has been exceeded on this site collection.

Result Sets: The **proc_SecCreateWebGroup** stored procedure MUST NOT return any result set.

3.1.5.41 proc_SecDecCurrentUsersCount

The **proc_SecDecCurrentUsersCount** stored procedure is called to reduce by one the total number of users in the specified **site** from the site collection. **proc_SecDecCurrentUsersCount** does not delete any user information. **proc_SecDecCurrentUsersCount** is defined using T-SQL syntax, as follows.

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

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection.

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

Result Sets: The **proc_SecDecCurrentUsersCount** stored procedure MUST NOT return any result set.

3.1.5.42 proc_SecGetAccountStatus

The **proc_SecGetAccountStatus** stored procedure is called to provide status information for a site collection's users that are not marked as deleted, and match the specified login name or email address. **proc_SecGetAccountStatus** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecGetAccountStatus (
@SiteId uniqueidentifier
,@Login nvarchar(255)
,@Email nvarchar(255)
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the requested user.

@Login: The login name of a user to be matched.

@Email: The email address of a user to be matched. If this parameter is an **empty string**, it is ignored.

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

Result Sets: The **proc_SecGetAccountStatus** stored procedure MUST return an **Account Status Result Set** (section 2.2.4.1) and MUST contain one row for each user which is not a domain group. The **Account Status Result Set** MUST contain no rows if no matching users are found.

3.1.5.43 proc_SecGetCompleteWebGroupMemberList

The **proc_SecGetCompleteWebGroupMemberList** stored procedure is called to list all group assignments for all the principals with permissions on a specified **site**. **proc_SecGetCompleteWebGroupMemberList** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecGetCompleteWebGroupMemberList (
@WebId                          uniqueidentifier
,@LatestSecurityVersion         bigint OUTPUT
);
```

@WebId: The **Site Identifier** (section 2.2.1.19) of the site whose permission information is requested.

@LatestSecurityVersion: The current security version value for the specified site collection. This value is incremented with every security setting modification on the site collection.

Return Values: The **proc_SecGetCompleteWebGroupMemberList** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
3	The site collection or the site could not be found.

Result Sets: The **proc_SecGetCompleteWebGroupMemberList** stored procedure MUST return a single result set **Group Member Result Set** (section 2.2.4.22). The **Group Member Result Set** returns one row for each group assignment in effect for the specified site. If the *@WebId* parameter is not valid, this result set would not be returned.

3.1.5.44 proc_SecGetCurrentUsersCount

The **proc_SecGetCurrentUsersCount** stored procedure is called to obtain the count of users in the specified site collection. **proc_SecGetCurrentUsersCount** is defined using T-SQL syntax, as follows:

```
PROCEDURE proc SecGetCurrentUsersCount (
@SiteId uniqueidentifier
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection.

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

Result Sets: The **proc_SecGetCurrentUsersCount** stored procedure MUST return a **User Count Result Set** (section 2.2.4.55). The **User Count Result Set** MUST return one row for a given valid *@SiteId*. Otherwise if *@SiteId* is invalid, zero rows are returned.

3.1.5.45 proc_SecGetGroupMembershipToken

The **proc_SecGetGroupMembershipToken** stored procedure is called to get the identifier of the site group to which user belongs. **proc_SecGetGroupMembershipToken** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecGetGroupMembershipToken (
@SiteId     uniqueidentifier
,@WebId     uniqueidentifier
,@UserId    int
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) for a site collection.

@WebId: The **Site Identifier** (section 2.2.1.19) for the **site**.

@UserId: The **User Identifier** (section 2.2.1.24) for the user.

Return Values: The **proc_SecGetGroupMembershipToken** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
3	The ancestor site of the given site specified by *@WebId* is null.

Result Sets: The **proc_SecGetGroupMembershipToken** stored procedure MUST return a **Group Membership Token Result Set** (section 2.2.4.23) if the site has the ancestor site that manages the security.

3.1.5.46 proc_SecGetIndividualUrlSecurity

The **proc_SecGetIndividualUrlSecurity** stored procedure is called to get security information about a document. **proc_SecGetIndividualUrlSecurity** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecGetIndividualUrlSecurity (
@SiteId                         uniqueidentifier
,@WebId                         uniqueidentifier
```

```
,@FullUrl                       nvarchar(260)
,@DirName                       nvarchar(256)
,@LeafName                      nvarchar(128)
,@UserId                        int
,@AttachmentsFlag               tinyint
,@bGetAttachmentWritePerm       bit
,@bGetListMetaData              bit = 0
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the document.

@WebId: The **Site Identifier** (section 2.2.1.19) of the **site** containing the document.

@FullUrl: The store-relative form URL of the document.

@DirName: The directory name of the document.

@LeafName: The leaf name of the document.

@UserId: The **User Identifier** (section 2.2.1.24) of the current user, used to check for access privileges.

@AttachmentsFlag: An **Attachments Flag** (section 2.2.2.1) value specifying whether the document is a file or an attachments folder.

@bGetAttachmentWritePerm: A bit flag specifying whether to return information about the write permissions of the current user to save an attachment at the specified document location. If this parameter is set to 1, and the *@AttachmentsFlag* parameter is not 0, **proc_SecGetIndividualUrlSecurity** MUST return the information about the current user's permissions as part of the **Individual URL Security Result Set** (section 2.2.4.25). This parameter MUST be ignored if *@AttachmentsFlag* is 0 or NULL.

@bGetListMetaData: A bit flag specifying whether list metadata is requested for the List.

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

Result Sets:

The **proc_SecGetIndividualUrlSecurity** stored procedure MUST return the **Null Individual URL Security Result Set** (section 2.2.4.37). The **Null Individual URL Security Result Set** MUST only be returned if the specified document is not contained within a **list (1)** and MUST return a single row.

The **proc_SecGetIndividualUrlSecurity** stored procedure MUST return the **Individual URL Security Result Set** (section 2.2.4.25). The **Individual URL Security Result Set** MUST only be returned if the specified document is contained within a list (1).

The **proc_SecGetIndividualUrlSecurity** stored procedure MUST return **List MetaData Result Set** (section 2.2.4.32). The **List Metadata Result Set** MUST only be returned if the *@bGetListMetadata* parameter is set to 1 and the document is contained within a list (1).

3.1.5.47 proc_SecGetPrincipalByEmail

The **proc_SecGetPrincipalByEmail** stored procedure is called to return user information by email address. **proc_SecGetPrincipalByEmail** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc SecGetPrincipalByEmail (
@SiteId uniqueidentifier
,@Email nvarchar(255)
```

```
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection containing the user.

@Email: The user's email address. If this parameter is NULL, the email address MUST be set to the **empty string**.

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

Result Sets: The **proc_SecGetPrincipalByEmail** stored procedure MUST return a **Principal User Information Result Set** (section 2.2.4.39). The **Principal User Information Result Set** returns information about the user associated with the specified email address. The **Principal User Information Result Set** MUST contain zero rows if no users have the specified email address or if the associated user has been marked as deleted.

3.1.5.48 proc_SecGetPrincipalById

The **proc_SecGetPrincipalById** stored procedure is called to return information about a principal identifier or collection of principal identifiers based on a specified user identifier. **proc_SecGetPrincipalById** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecGetPrincipalById (
@SiteId     uniqueidentifier
,@UserId    int
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection containing the user.

@UserId: The **User Identifier** (section 2.2.1.24) of the **principal (1)** to return information.

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

Result Sets: The **proc_SecGetPrincipalById** stored procedure MUST return a **Principal User Information Result Set** (section 2.2.4.39). The **Principal User Information Result Set** returns information about a specified principal (1) or the **members** of a site group. If *@UserId* matches a principal (1), the **Principal User Information Result Set** MUST be returned once with a single row of data for the principal.

3.1.5.49 proc_SecGetPrincipalByIdInWeb

The **proc_SecGetPrincipalByIdInWeb** stored procedure is called to return information about a **principal** or collection of principals based on a specified user identifier and site identifier. **proc_SecGetPrincipalByIdInWeb** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecGetPrincipalByIdInWeb (
@SiteId     uniqueidentifier
,@WebId     uniqueidentifier
,@UserId    int
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection containing the principal (1).

@WebId: The **Site Identifier** (section 2.2.1.19) of the **site** containing the principal (1).

@UserId: The **User Identifier** (section 2.2.1.24) of the principal (1) whose information is returned.

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

Result Sets: The **proc_SecGetPrincipalByIdInWeb** stored procedure MUST return a **Principal User Information Result Set** (section 2.2.4.39).

3.1.5.50 proc_SecGetPrincipalByLogin

The **proc_SecGetPrincipalByLogin** stored procedure is called to return security and attribute information for a principal (a user or domain group) that is identified by a specified login name. **proc_SecGetPrincipalByLogin** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecGetPrincipalByLogin (
@SiteId uniqueidentifier
,@Login nvarchar(255)
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection associated with the **principal (1)** whose information is requested.

@Login: The login name of the principal (1). This parameter MUST NOT be NULL.

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

Result Sets: The **proc_SecGetPrincipalByLogin** stored procedure MUST return a **Principal User Information Result Set** (section 2.2.4.39). The **Principal User Information Result Set** returns security and attribute information about a specified principal (1). The **Principal User Information Result Set** MUST be returned, and MUST contain one row if the *@Login* parameter matches an existing principal (1) who is not marked as deleted in the **UserInfo Table** (section 2.2.5.7). Otherwise, it MUST contain no rows.

3.1.5.51 proc_SecGetPrincipalByLogin20

The **proc_SecGetPrincipalByLogin20** stored procedure is called to return **security principal** information based on up to 20 separate login names. **proc_SecGetPrincipalByLogin20** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc SecGetPrincipalByLogin20 (
@SiteId         uniqueidentifier
,@PrincipalId01 nvarchar(255)
,@PrincipalId02 nvarchar(255)
,@PrincipalId03 nvarchar(255)
,@PrincipalId04 nvarchar(255)
,@PrincipalId05 nvarchar(255)
,@PrincipalId06 nvarchar(255)
,@PrincipalId07 nvarchar(255)
,@PrincipalId08 nvarchar(255)
,@PrincipalId09 nvarchar(255)
,@PrincipalId10 nvarchar(255)
,@PrincipalId11 nvarchar(255)
,@PrincipalId12 nvarchar(255)
,@PrincipalId13 nvarchar(255)
,@PrincipalId14 nvarchar(255)
,@PrincipalId15 nvarchar(255)
,@PrincipalId16 nvarchar(255)
,@PrincipalId17 nvarchar(255)
,@PrincipalId18 nvarchar(255)
```

```
,@PrincipalId19 nvarchar(255)
,@PrincipalId20 nvarchar(255)
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection that contains the security principals.

@PrincipalId01: The login name for the user to be returned.

@PrincipalId02: The login name for the user to be returned.

@PrincipalId03: The login name for the user to be returned.

@PrincipalId04: The login name for the user to be returned.

@PrincipalId05: The login name for the user to be returned.

@PrincipalId06: The login name for the user to be returned.

@PrincipalId07: The login name for the user to be returned.

@PrincipalId08: The login name for the user to be returned.

@PrincipalId09: The login name for the user to be returned.

@PrincipalId10: The login name for the user to be returned.

@PrincipalId11: The login name for the user to be returned.

@PrincipalId12: The login name for the user to be returned.

@PrincipalId13: The login name for the user to be returned.

@PrincipalId14: The login name for the user to be returned.

@PrincipalId15: The login name for the user to be returned.

@PrincipalId16: The login name for the user to be returned.

@PrincipalId17: The login name for the user to be returned.

@PrincipalId18: The login name for the user to be returned.

@PrincipalId19: The login name for the user to be returned.

@PrincipalId20: The login name for the user to be returned.

Return Values: The **proc_SecGetPrincipalByLogin20** stored procedure MUST return an integer return code of 0.

Result Sets: The **proc_SecGetPrincipalByLogin20** stored procedure MUST return a **Principal User Information Result Set** (section 2.2.4.39). The **Principal User Information Result Set** returns user information about a specified security principal. The **Principal User Information Result Set** MUST always return one set of security **principal(2)** information for each non-NULL *@PrincipalId##* that matches the identifier of the security principal.

3.1.5.52 proc_SecGetPrincipalByLoginInWeb

The **proc_ SecGetPrincipalByLoginInWeb** stored procedure is called to return security and attribute information for a **principal (1)** based on a specified site identifier and login. **proc_SecGetPrincipalByLoginInWeb** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecGetPrincipalByLoginInWeb (
@SiteId uniqueidentifier
,@WebId uniqueidentifier
,@Login nvarchar(255)
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection associated with the principal (1) whose information is requested.

@WebId: The **Site Identifier** (section 2.2.1.19) of the **site**.

@Login: The login name of the principal (1).

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

Result Sets: The **proc_SecGetPrincipalByLoginInWeb** stored procedure MUST return a **Principal User Information Result Set** (section 2.2.4.39). The **Principal User Information Result Set** returns user information about a specified **security principal**.

3.1.5.53 proc_SecGetPrincipalDisplayInformation20

The **proc_SecGetPrincipalDisplayInformation20** stored procedure is called to return **security principal**, site group, or web group information for up to 20 **principal (1)** identifiers. **proc_SecGetPrincipIDisplayInformation20** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc SecGetPrincipalDisplayInformation20 (
@SiteId         uniqueidentifier
,@WebId         uniqueidentifier
,@PrincipalId01 int
,@PrincipalId02 int
,@PrincipalId03 int
,@PrincipalId04 int
,@PrincipalId05 int
,@PrincipalId06 int
,@PrincipalId07 int
,@PrincipalId08 int
,@PrincipalId09 int
,@PrincipalId10 int
,@PrincipalId11 int
,@PrincipalId12 int
,@PrincipalId13 int
,@PrincipalId14 int
,@PrincipalId15 int
,@PrincipalId16 int
,@PrincipalId17 int
,@PrincipalId18 int
,@PrincipalId19 int
,@PrincipalId20 int
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection containing the security principals and the **security groups** to be listed.

@WebId: The **Site Identifier** (section 2.2.1.19) for a **site**.

@PrincipalId01: The identifier for a security principal, site group, or web group. One or more result sets are returned for a non-NULL *@PrincipalId01* parameter.

@PrincipalId02: The identifier for a security principal, site group, or web group. One or more result sets are returned for a non-NULL *@PrincipalId02* parameter.

@PrincipalId03: The identifier for a security principal, site group, or web group. One or more result sets are returned for a non-NULL *@PrincipalId03* parameter.

@PrincipalId04: The identifier for a security principal, site group, or web group. One or more result sets are returned for a non-NULL *@PrincipalId04* parameter.

@PrincipalId05: The identifier for a security principal, site group, or web group. One or more result sets are returned for a non-NULL *@PrincipalId05* parameter.

@PrincipalId06: The identifier for a security principal, site group, or web group. One or more result sets are returned for a non-NULL *@PrincipalId06* parameter.

@PrincipalId07: The identifier for a security principal, site group, or web group. One or more result sets are returned for a non-NULL *@PrincipalId07* parameter.

@PrincipalId08: The identifier for a security principal, site group, or web group. One or more result sets are returned for a non-NULL *@PrincipalId08* parameter.

@PrincipalId09: The identifier for a security principal, site group, or web group. One or more result sets are returned for a non-NULL *@PrincipalId09* parameter.

@PrincipalId10: The identifier for a security principal, site group, or web group. One or more result sets are returned for a non-NULL *@PrincipalId10* parameter.

@PrincipalId11: The identifier for a security principal, site group, or web group. One or more result sets are returned for a non-NULL *@PrincipalId11* parameter.

@PrincipalId12: The identifier for a security principal, site group, or web group. One or more result sets are returned for a non-NULL *@PrincipalId12* parameter.

@PrincipalId13: The identifier for a security principal, site group, or web group. One or more result sets are returned for a non-NULL *@PrincipalId13* parameter.

@PrincipalId14: The identifier for a security principal, site group, or web group. One or more result sets are returned for a non-NULL *@Principa14* parameter.

@PrincipalId15: The identifier for a security principal, site group, or web group. One or more result sets are returned for a non-NULL *@PrincipalId15* parameter.

@PrincipalId16: The identifier for a security principal, site group, or web group. One or more result sets are returned for a non-NULL *@PrincipalId16* parameter.

@PrincipalId17: The identifier for a security principal, site group, or web group. One or more result sets are returned for a non-NULL *@PrincipalId17* parameter.

@PrincipalId18: The identifier for a security principal, site group, or web group. One or more result sets are returned for a non-NULL *@PrincipalId18* parameter.

@PrincipalId19: The identifier for a security principal, site group, or web group. One or more result sets are returned for a non-NULL *@PrincipalId19* parameter.

@PrincipalId20: The identifier for a security principal, site group, or web group. One or more result sets are returned for a non-NULL *@PrincipalId20* parameter.

Return Values: The **proc_SecGetPrincipalDisplayInformation20** stored procedure MUST return an integer return code of 0.

Result Sets:

The **proc_SecGetPrincipalDisplayInformation20** stored procedure MUST return a **Principal Display Information Result Set** (section 2.2.4.38) for each not NULL *@PrincipalId*. If *@PrincipalId* is greater than 1073741824, the **Principal Display Information Result Set** will return information about web groups, else it will return information about site groups. If site groups information is NULL, **proc_SecGetPrincipalDisplayInformation20** will also return **User Display Information Result Set**.

The **proc_SecGetPrincipalDisplayInformation20** stored procedure MUST return a **User Display Information Result Set** (section 2.2.4.56) for each non-NULL *@PrincipalId* and if the *@PrincipalId* is less than or equal to 1073741824 and if **Principal Display Information Result Set** contain no rows.

3.1.5.54 proc_SecGetSiteGroupById

The **proc_SecGetSiteGroupById** stored procedure is called to get information about a site group given its **site collection identifier**. **proc_SecGetSiteGroupById** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecGetSiteGroupById (
 @SiteId  uniqueidentifier
,@GroupId int
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site containing the sitegroup.

@GroupId: The identifier of the site group.

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

Result Sets: The **proc_SecGetSiteGroupById** stored procedure MUST return a **Site Group Result Set** (section 2.2.4.50). The **Site Group Result Set** contains a single row where Sec_SiteGroupsView.SiteWebID MUST equal *@SiteId* and Sec_SiteGroupsView.ID MUST equal *@GroupId*. If there is no match, the **Site Group Result Set** will be empty.

3.1.5.55 proc_SecGetSiteGroupByTitle

The **proc_SecGetSiteGroupByTitle** stored procedure is called to get site group information for the site group with the specified user-friendly name. **proc_SecGetSiteGroupByTitle** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecGetSiteGroupByTitle (
@SiteId uniqueidentifier
,@Title nvarchar(255)
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection that contains the sitegroup.

@Title: The user-friendly name of the site group.

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

Result Sets: The **proc_SecGetSiteGroupByTitle** stored procedure MUST return a **Site Group Information Result Set** (section 2.2.4.49). The **Site Group Information Result Set** MUST contain one row of site group information if the *@SiteId* parameter value and the *@Title* parameter value match an existing site group. Otherwise, zero rows MUST be returned.

3.1.5.56 proc_SecGetSiteGroupByTitle20

The **proc_SecGetSiteGroupByTitle20** stored procedure is called to provide information about sitegroups in bulk, as specified by a set of up to 20 site group titles.
proc_SecGetSiteGroupByTitle20 is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecGetSiteGroupByTitle20 (
@SiteId         uniqueidentifier
,@PrincipalId01 nvarchar(255)
,@PrincipalId02 nvarchar(255)
,@PrincipalId03 nvarchar(255)
,@PrincipalId04 nvarchar(255)
,@PrincipalId05 nvarchar(255)
,@PrincipalId06 nvarchar(255)
,@PrincipalId07 nvarchar(255)
,@PrincipalId08 nvarchar(255)
,@PrincipalId09 nvarchar(255)
,@PrincipalId10 nvarchar(255)
,@PrincipalId11 nvarchar(255)
,@PrincipalId12 nvarchar(255)
,@PrincipalId13 nvarchar(255)
,@PrincipalId14 nvarchar(255)
,@PrincipalId15 nvarchar(255)
,@PrincipalId16 nvarchar(255)
,@PrincipalId17 nvarchar(255)
,@PrincipalId18 nvarchar(255)
,@PrincipalId19 nvarchar(255)
,@PrincipalId20 nvarchar(255)
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection that contains the specified site groups.

@PrincipalId01: The name of the site group.

@PrincipalId02: The name of the site group.

@PrincipalId03: The name of the site group.

@PrincipalId04: The name of the site group.

@PrincipalId05: The name of the site group.

@PrincipalId06: The name of the site group.

@PrincipalId07: The name of the site group.

@PrincipalId08: The name of the site group.

@PrincipalId09: The name of the site group.

@PrincipalId10: The name of the site group.

@PrincipalId11: The name of the site group.

@PrincipalId12: The name of the site group.

@PrincipalId13: The name of the site group.

@PrincipalId14: The name of the site group.

@PrincipalId15: The name of the site group.

@PrincipalId16: The name of the site group.

@PrincipalId17: The name of the site group.

@PrincipalId18: The name of the site group.

@PrincipalId19: The name of the site group.

@PrincipalId20: The name of the site group.

Return Values: The **proc_ SecGetSiteGroupByTitle20** stored procedure MUST return an integer return code of 0.

Result Sets: The **proc_ SecGetSiteGroupByTitle20** stored procedure MUST return one **Site Group Information Result Set** (section 2.2.4.49) for each non-NULL *@PrincipalId##*. The **Site Group Information Result Set** returns once per each non-NULL *@PrincipalId##*, and the **Site Group Information Result Set** MUST contain one row of site group information if the *@SiteId* parameter value and the *@PrincipalId##* parameter value match an existing sitegroup. Otherwise, zero rows MUST be returned.

3.1.5.57 proc_SecGetWebGroupById

The **proc_SecGetWebGroupById** stored procedure is called to get information about a specific site group of a **site** given its GroupId. **proc_SecGetWebGroupById** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecGetWebGroupById (
@SiteId   uniqueidentifier
,@WebId   uniqueidentifier
,@GroupId int
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection that contains the specified site.

@WebId: The **Site Identifier** (section 2.2.1.19) of the site.

@GroupId: An integer value referring to the **group**.

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

Result Sets: The **proc_SecGetWebGroupById** stored procedure MUST return the **Web Group Information Result Set** (section 2.2.4.61) on successful execution. The **Web Group Information Result Set** contains the information for the specified site group. **Web Group Information Result Set** MUST be empty if the group is not found in the site.

3.1.5.58 proc_SecGetWebGroupByTitle

The **proc_SecGetWebGroupByTitle** stored procedure is called to get site group information for the site group with the specified name. **proc_SecGetWebGroupByTitle** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecGetWebGroupByTitle (
@SiteId uniqueidentifier
,@WebId uniqueidentifier
,@Title nvarchar(255)
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the specified **site**.

@WebId: The **Site Identifier** (section 2.2.1.19) of the site.

@Title: The user-friendly name of the site group.

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

Result Sets: The **proc_SecGetWebGroupByTitle** stored procedure MUST return a **Web Group Information Result Set** (section 2.2.4.61). The **Web Group Information Result Set** MUST contain one row of site group information if the *@SiteId* and *@Title* parameter value match an existing site group. Otherwise, zero rows MUST be returned.

3.1.5.59 proc_SecGetWebGroupByTitle20

The **proc_SecGetWebGroupByTitle20** stored procedure is called to provide information about site groups in bulk, as specified by a set of up to 20 site group titles. **proc_SecGetWebGroupByTitle20** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecGetWebGroupByTitle20 (
@SiteId         uniqueidentifier
,@WebId         uniqueidentifier
,@PrincipalId01 nvarchar(255)
,@PrincipalId02 nvarchar(255)
,@PrincipalId03 nvarchar(255)
,@PrincipalId04 nvarchar(255)
,@PrincipalId05 nvarchar(255)
,@PrincipalId06 nvarchar(255)
,@PrincipalId07 nvarchar(255)
,@PrincipalId08 nvarchar(255)
,@PrincipalId09 nvarchar(255)
,@PrincipalId10 nvarchar(255)
,@PrincipalId11 nvarchar(255)
,@PrincipalId12 nvarchar(255)
,@PrincipalId13 nvarchar(255)
,@PrincipalId14 nvarchar(255)
,@PrincipalId15 nvarchar(255)
,@PrincipalId16 nvarchar(255)
,@PrincipalId17 nvarchar(255)
,@PrincipalId18 nvarchar(255)
,@PrincipalId19 nvarchar(255)
,@PrincipalId20 nvarchar(255)
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection that contains the specified site groups.

@WebId: The **Site Identifier** (section 2.2.1.19) for the **site** that contains the specified site groups.

@PrincipalId01: The user-friendly display name for the site group.

@PrincipalId02: The user-friendly display name for the site group.

@PrincipalId03: The user-friendly display name for the site group.

@PrincipalId04: The user-friendly display name for the site group.

@PrincipalId05: The user-friendly display name for the site group.

@PrincipalId06: The user-friendly display name for the site group.

@PrincipalId07: The user-friendly display name for the site group.

@PrincipalId08: The user-friendly display name for the site group.

@PrincipalId09: The user-friendly display name for the site group.

@PrincipalId10: The user-friendly display name for the site group.

@PrincipalId11: The user-friendly display name for the site group.

@PrincipalId12: The user-friendly display name for the site group.

@PrincipalId13: The user-friendly display name for the site group.

@PrincipalId14: The user-friendly display name for the site group.

@PrincipalId15: The user-friendly display name for the site group.

@PrincipalId16: The user-friendly display name for the site group.

@PrincipalId17: The user-friendly display name for the site group.

@PrincipalId18: The user-friendly display name for the site group.

@PrincipalId19: The user-friendly display name for the site group.

@PrincipalId20: The user-friendly display name for the site group.

Return Values: The **proc_SecGetWebGroupByTitle20** stored procedure MUST return an integer return code of 0.

Result Sets: The **proc_SecGetWebGroupByTitle20** stored procedure MUST return a **Web Group Information Result Set** (section 2.2.4.61) for each non-NULL *@PrincipalId##*. The **Web Group Information Result Set** MUST contain one row of site group information if the *@SiteId* and *@Principal##* parameter value match an existing sitegroup. Otherwise, zero rows MUST be returned.

3.1.5.60 proc_SecGetWebRequestAccess

The **proc_SecGetWebRequestAccess** stored procedure is called to get the request access email address for a specified **site**. **proc_SecGetWebRequestAccess** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc SecGetWebRequestAccess (
@WebId uniqueidentifier
);
```

@WebId: The **Site Identifier** (section 2.2.1.19) of the site for which the web access is requested.

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

Result Sets: The **proc_SecGetWebRequestAccess** stored procedure MUST return a **Request Access Email Result Set** (section 2.2.4.41).

3.1.5.61 proc_SecListAllSiteMembers

The **proc_SecListAllSiteMembers** stored procedure is called to provide information about all non-deleted **security principals** in the specified site collection. **proc_SecListAllSiteMembers** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc SecListAllSiteMembers (
@SiteId uniqueidentifier
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection that contains the requested security principal information.

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

Result Sets: The **proc_SecListAllSiteMembers** stored procedure MUST return a **Principal User Information Result Set** (section 2.2.4.39). The **Principal User Information Result Set** returns information about the non-deleted users directly associated with the site collection specified by *@SiteId*. Zero or more rows MUST be returned. There MUST be one row per non-deleted user in the site collection specified by *@SiteId*.

3.1.5.62 proc_SecListAllUsersWebGroups

The **proc_SecListAllUsersWebGroups** stored procedure is called to list the site groups for all users in a site. **proc_SecListAllUsersWebGroups** is defined using T-SQL syntax, as follows.

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

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection.

@WebId: The **Site Identifier** (section 2.2.1.19) of the **site**.

Return Values: The **proc_SecListAllUsersWebGroups** stored procedure MUST return an integer return code, which MUST be zero.

Result Sets: The **proc_SecListAllUsersWebGroups** stored procedure MUST return the **Users Web Groups Result Set** (section 2.2.4.60).

3.1.5.63 proc_SecListAllWebMembers

The **proc_SecListAllWebMembers** stored procedure is called to list all non-deleted user and domain group accounts registered with a specified **site**. **proc_SecListAllWebMembers** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc SecListAllWebMembers (
@SiteId uniqueidentifier
,@WebId uniqueidentifier
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection that contains the specified site.

@WebId: The **Site Identifier** (section 2.2.1.19) of the site.

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

Result Sets: The **proc_SecListAllWebMembers** stored procedure MUST return a **Principal User Information Result Set** (section 2.2.4.39). The **Principal User Information Result Set** MUST return zero rows if the given site has no **members**. Otherwise, it MUST contain one row for each member if *@WebId* matches an existing site within the specified site collection.

3.1.5.64 proc_SecListAllWebMembersInWebGroups

The **proc_SecListAllWebMembersInWebGroups** stored procedure is called to list all non-deleted user and domain group accounts registered with a specified **site** in the site groups. **proc_SecListAllWebMembersInWebGroups** is defined using T-SQL syntax, as follows.

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

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the site groups to be listed.

@WebId: The **Site Identifier** (section 2.2.1.19) of the site (2).

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

Result Sets: The **proc_SecListAllWebMembersInWebGroups** stored procedure returns a **Principal User Information Result Set** (section 2.2.4.39). The **Principal User Information Result Set** returns a list of all non-deleted users or domain groups registered with the specified site (2) in the site groups. The **Principal User Information Result Set** MUST return zero rows if the site specified by *@WebId* has no **members** in site groups. Otherwise, it MUST contain one row for each member if *@WebId* matches an existing site within the site collection specified by *@SiteId*.

3.1.5.65 proc_SecListDerivedDomainGroups

The **proc_SecListDerivedDomainGroups** is called to get the list of domain groups for a specified **member**. **proc_SecListDerivedDomainGroups** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecListDerivedDomainGroups (
@SiteId   uniqueidentifier
,@WebId   uniqueidentifier
,@Id      int
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection with domain groups.

@WebId: The **Site Identifier** (section 2.2.1.19) of the **site**.

@Id: An integer value which identifies a member within a site.

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

Result Sets: The **proc_SecListDerivedDomainGroups** stored procedure MUST return a **Domain Group Result Set** (section 2.2.4.18).

3.1.5.66 proc_SecListSiteGroupMembership

The **proc_SecListSiteGroupMembership** stored procedure is called to list **members** in a specified site group. **proc_SecListSiteGroupMembership** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecListSiteGroupMembership (
@SiteId   uniqueidentifier
,@GroupId int
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the specified **site**.

@GroupId: The identifier of the site group.

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

Result Sets: The **proc_SecListSiteGroupMembership** stored procedure returns a **Principal User Information Result Set** (section 2.2.4.39).

3.1.5.67 proc_SecListSiteGroups

The **proc_SecListSiteGroups** stored procedure is called to list site group information for a specified site collection. **proc_SecListSiteGroups** is defined using T-SQL syntax, as follows.

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

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection that contains the sitegroups to list.

Return Values: The **proc_SecListSiteGroups** stored procedure MUST return an integer return code, which MUST be zero.

Result Sets: The **proc_SecListSiteGroups** stored procedure MUST return a **Site Group Information Result Set** (section 2.2.4.49). The **Site Group Information Result Set** MUST be empty if the site collection is not found. Otherwise, 1 row MUST be returned for each site group in the specified site collection.

3.1.5.68 proc_SecListSiteGroupsContainingUser

The **proc_SecListSiteGroupsContainingUser** stored procedure is called to list the site groups in which the user is a **member**. **proc_SecListSiteGroupsContainingUser** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecListSiteGroupsContainingUser (
@SiteId     uniqueidentifier
,@UserId    int
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection that contains the sitegroups to be listed. This parameter MUST correspond to a valid site collection, and MUST NOT be NULL.

@UserId: The **User Identifier** (section 2.2.1.24) used to find sitegroups that contain this user.

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

Result Sets: The **proc_SecListSiteGroupsContainingUser** stored procedure MUST return a **Site Group Information Result Set** (section 2.2.4.49), which contains the information of site groups where the user is a member and MUST contain one row for each sitegroup found for the user.

3.1.5.69 proc_SecListSiteGroupsInWebGroup

The **proc_SecListSiteGroupsInWebGroup** is called to list sitegroup information for a specified site collection, site identifier, and site group identifier. **proc_SecListSiteGroupsInWebGroup** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecListSiteGroupsInWebGroup (
@SiteId     uniqueidentifier
,@WebId     uniqueidentifier
,@RoleId    int
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection that contains the site groups to list.

@WebId: The **Site Identifier** (section 2.2.1.19) of the **site** to query for **members**.

@RoleId: The **Role Identifier** (section 2.2.1.15) for the role.

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

Result Sets: The **proc_SecListSiteGroupsInWebGroup** stored procedure returns a **Site Group Result Set** (section 2.2.4.50). The **Site Group Result Set** MUST return zero or one rows. The **Site Group Result Set** contains a single row where Sec_SiteGroupsView.SiteID MUST equal *@SiteId* for the requested WebId and RoleId for the site. If there is no match, the **Site Group Result Set** will be empty.

3.1.5.70 proc_SecListSiteGroupsInWebGroups

The **proc_SecListSiteGroupsInWebGroups** stored procedure is called to list the sitegroups information given in the site groups. **proc_SecListSiteGroupsInWebGroups** is defined using T-SQL syntax, as follows.

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

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection that contains the sitegroups to be listed.

@WebId: The **Site Identifier** (section 2.2.1.19) of the **site**.

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

Result Sets: The **proc_SecListSiteGroupsInWebGroups** stored procedure MUST return a **Site Group Result Set** (section 2.2.4.50). The **Site Group Result Set** returns zero or one rows. The **Site Group Result Set** contains a single row where Sec_SiteGroupsView.SiteID MUST equal *@SiteId* for the requested WebId for the site. If there is no match, the **Site Group Result Set** will be empty.

3.1.5.71 proc_SecListSiteGroupsWhichUserOwns

The **proc_SecListSiteGroupsWhichUserOwns** stored procedure is called to get site group information for the site groups that are owned by the specified **principal (1)**. **proc_SecListSiteGroupsWhichUserOwns** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecListSiteGroupsWhichUserOwns (
@SiteId     uniqueidentifier
,@UserId    int
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection that contains the principal (1) and the site groups to be listed.

@UserId: The **User Identifier** (section 2.2.1.24) of the principal (1), used to find the site groups it owns.

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

Result Sets: The **proc_SecListSiteGroupsWhichUserOwns** stored procedure MUST return a **Site Group Information Result Set** (section 2.2.4.49).

3.1.5.72 proc_SecListWebGroupMembership

The **proc_SecListWebGroupMembership** stored procedure is called to get information about a specific site group of a **site** given its *@GroupId*. **proc_SecListWebGroupMembership** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecListWebGroupMembership (
@SiteId   uniqueidentifier
,@WebId   uniqueidentifier
,@GroupId int
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection.

@WebId: The **Site Identifier** (section 2.2.1.19) of the site.

@GroupId: The identifier of the site group.

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

Result Sets: The **proc_SecListWebGroupMembership** stored procedure MUST return a **Principal User Information Result Set** (section 2.2.4.39). The **Principal User Information Result Set** returns user information about a specified **security principal**.

3.1.5.73 proc_SecListWebGroups

The **proc_SecListWebGroups** stored procedure is called to list site group information for a specified site collection. **proc_SecListWebGroups** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc SecListWebGroups (
@SiteId uniqueidentifier
,@WebId uniqueidentifier
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection that contains the specified **site**.

@WebId: The **Site Identifier** (section 2.2.1.19) of the site.

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

Result Sets: The **proc_SecListWebGroups** stored procedure MUST return a **Web Group Information Result Set** (section 2.2.4.61). The **Web Group Information Result Set** contains all site group information for the site collection. The **Web Group Information Result Set** MUST be empty if the site collection is not found. Otherwise, one row MUST be returned for each site group in the specified site collection.

3.1.5.74 proc_SecListWebGroupsByType

The **proc_SecListWebGroupByType** stored procedure is called to list all the site group **member** information given its group type. **proc_SecListWebGroupByType** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc SecListWebGroupsByType (
@SiteId      uniqueidentifier
,@WebId      uniqueidentifier
,@GroupType  int
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the specified **site**.

@WebId: The **Site Identifier** (section 2.2.1.19) of the site.

@GroupType: It is an integer value indicating the type of **group**.

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

Result Sets: The **proc_SecListWebGroupByType** stored procedure MUST return a **Web Group Information Result Set** (section 2.2.4.61). The **Web Group Information Result Set** contains all site group information for the site collection. The **Web Group Information Result Set** MUST be empty if the site collection is not found. Otherwise, one row MUST be returned for each site group in the specified site collection.

3.1.5.75 proc_SecListWebGroupsContainingSiteGroup

The **proc_SecListWebGroupsContainingSiteGroup** stored procedure is called to list all the roles a site group belongs to. **proc_SecListWebGroupsContainingSiteGroup** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecListWebGroupsContainingSiteGroup (
@SiteId   uniqueidentifier
,@WebId   uniqueidentifier
,@GroupId int
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection.

@WebId: The **Site Identifier** (section 2.2.1.19) of the **site**.

@GroupId: An integer value referring to a site group to be listed.

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

Result Sets: The **proc_SecListWebGroupsContainingSiteGroup** stored procedure MUST return a **Web Group Information Result Set** (section 2.2.4.61) for the specified group in the site. The **Web Group Information Result Set** contains the site group information for the site collection.

3.1.5.76 proc_SecListWebGroupsContainingUser

The **proc_SecListWebGroupsContainingUser** stored procedure is called to list the site groups in which the user is a **member**. **proc_SecListWebGroupsContainingUser** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecListWebGroupsContainingUser (
@SiteId     uniqueidentifier
,@WebId     uniqueidentifier
,@UserId    int
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection that contains the site groups to be listed.

@WebId: The **Site Identifier** (section 2.2.1.19) of the site group.

@UserId: The **User Identifier** (section 2.2.1.24) used to find site groups that contain this user.

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

Result Sets: The **proc_SecListWebGroupsContainingUser** stored procedure MUST return a **Web Group Information Result Set** (section 2.2.4.61). The **Web Group Information Result Set** contains information for site groups where the user is a member. The **Web Group Information Result Set** MUST be returned and MUST contain one row for each site group found for the user.

3.1.5.77 proc_SecMigrateUser

The **proc_SecMigrateUser** stored procedure is called to update the SystemID and login name for a specified user (that is, a **principal (1)** that is not a domain group) in the **UserInfo Table** (section 2.2.5.7). If an existing user is found with the new login name and **SystemID**, **proc_SecMigrateUser** MUST update that user with a new unique **SystemID** and mark that user as deleted in the UserInfo table before updating the specified user. **proc_SecMigrateUser** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecMigrateUser (
@OldLogin        nvarchar(255)
```

```
,@OldSystemId    tSystemID
,@NewLogin       nvarchar(255)
,@NewSystemId    tSystemID
,@NullSystemId   tSystemID
);
```

@OldLogin: The current value of the user's login name.

@OldSystemId: The current value of the user's **SystemID**. This parameter can be NULL. If the *@OldSystemId* parameter is not NULL, then the user specified by *@OldLogin* MUST also have the **SystemID** specified by *@OldSystemId*.

@NewLogin: The new value to set as the user's login name. This parameter MUST NOT be NULL.

@NewSystemId: The new **SystemID** to set for the user. This parameter MUST NOT be NULL.

@NullSystemId: The **SystemID** to set for the user. This parameter can be NULL.

Return Values: The **proc_SecMigrateUser** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
1003	An existing user was found with the new login name and **SystemID**, but **proc_SecMigrateUser** failed to update it with a new **SystemID** and mark it as deleted.

Result Sets: The **proc_SecMigrateUser** stored procedure MUST NOT return any result set.

3.1.5.78 proc_SecRemovePrincipalFromWebGroup

The **proc_SecRemovePrincipalFromWebGroup** stored procedure is called to remove a **principal (1)** from a site group. **proc_SecRemovePrincipalFromWebGroup** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecRemovePrincipalFromWebGroup (
@SiteId          uniqueidentifier
,@WebId          uniqueidentifier
,@GroupId        int
,@PrincipalId    int
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection associated with the principal (1).

@WebId: The **Site Identifier** (section 2.2.1.19) of the **site**.

@GroupId: An identifier for the site group that the principal (1) refers to.

@PrincipalId: The **User Identifier** (section 2.2.1.24) of the principal (1) to be deleted.

Return Values: The **proc_SecRemovePrincipalFromWebGroup** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.

Value	Description
4335	Cannot delete the last remaining item in the administrator site group.

Result Sets: The **proc_SecRemovePrincipalFromWebGroup** stored procedure MUST NOT return any result set.

3.1.5.79 proc_SecRemoveSiteGroup

The **proc_SecRemoveSiteGroup** stored procedure is called to remove a sitegroup. **proc_SecRemoveSiteGroup** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecRemoveSiteGroup (
@SiteId          uniqueidentifier
,@GroupId        int
,@UserID         int
,@SiteAdmin      bit
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the site group to be removed.

@GroupId: The identifier of the site group to be removed.

@UserID: An integer value identifying the owner of the **group** to be removed.

@SiteAdmin: A bit flag specifying whether the current user is a site administrator.

Return Values: The **proc_SecRemoveSiteGroup** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
5	Cannot delete site group.

Result Sets: The **proc_SecRemoveSiteGroup** stored procedure MUST return a **Site Acl Result Set** (section 2.2.4.45) only on successful execution.

3.1.5.80 proc_SecRemoveSiteGroupFromWeb

The **proc_SecRemoveSiteGroupFromWeb** stored procedure is called to remove a site group from a site (2). **proc_SecRemoveSiteGroupFromWeb** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecRemoveSiteGroupFromWeb (
@SiteId   uniqueidentifier
,@WebId   uniqueidentifier
,@GroupId int
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection that contains the specified site group.

@WebId: The **Site Identifier** (section 2.2.1.19) for the **site** that contains the specified site group.

@GroupId: The **Site Group Identifier** (section 2.2.1.18) of the site group to which the user belongs.

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

Result Sets: The **proc_SecRemoveSiteGroupFromWeb** stored procedure MUST return a **Site Acl Result Set** (section 2.2.4.45).

3.1.5.81 proc_SecRemoveUserFromSite

The **proc_SecRemoveUserFromSite** stored procedure is called to remove a user from the specified **site**. If the user is the site owner or secondary contact for the site, the user is not removed. The user's role memberships are cleared. If the user has ownership of any site groups, they are transferred to the new user if specified or to the site owner by default.
proc_SecRemoveUserFromSite is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecRemoveUserFromSite (
@SiteId                    uniqueidentifier
,@NewSiteGroupOwnerId      int
,@UserId                   int
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection that contains the user to be removed.

@NewSiteGroupOwnerId: The **User Identifier** (section 2.2.1.24) for the user who will take over **group** ownership from the user being removed. To change the group ownership, this value MUST be -1 or a valid user identifier in the specified site collection who is different from the user being removed. If the value is -1, the site collection owner will be the new site group owner.

@UserId: The **User Identifier** for the user who is being removed. If *@UserId* is the site collection owner or secondary contact, the change will not succeed. To succeed, this value MUST be an existing **User Identifier** for the specified site collection.

Return Values: The **proc_SecRemoveUserFromSite** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
16	The user being removed is the same as the new user. No changes were made.
4335	The user being removed is the owner or the secondary contact for the specified site collection. No changes were made.

Result Sets: The **proc_SecRemoveUserFromSite** stored procedure MUST return a **Site Acl Result Set** (section 2.2.4.45).

3.1.5.82 proc_SecRemoveUserFromSiteByLogin

The **proc_SecRemoveUserFromSiteByLogin** stored procedure is called to remove a user identified by login name from a specified site collection. **proc_SecRemoveUserFromSiteByLogin** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecRemoveUserFromSiteByLogin (
@SiteId                    uniqueidentifier
,@NewSiteGroupOwnerId      int
,@LoginName                nvarchar(255)
```

```
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection that contains the user to be removed.

@NewSiteGroupOwnerId: The **User Identifier** (section 2.2.1.24) for the user who will take over **group** ownership from the user being removed. To change the group ownership, this value MUST be -1. If the value is -1, the site collection owner will be the new site group owner.

@LoginName: The login name of the user to be removed from the site collection.

Return Values: The **proc_SecRemoveUserFromSiteByLogin** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
16	The user being removed is the same as the new user.
4335	The user being removed is the owner or the secondary contact for the specified site collection. No changes were made.

Result Sets:

The **proc_SecRemoveUserFromSiteByLogin** stored procedure MUST return a **User ID Result Set** (section 2.2.4.57).

The **proc_SecRemoveUserFromSiteByLogin** stored procedure MUST return a **Site Acl Result Set** (section 2.2.4.45) if **proc_SecRemoveUserFromSiteByLogin** executes successfully.

3.1.5.83 proc_SecRemoveUserFromSiteGroup

The **proc_SecRemoveUserFromSiteGroup** stored procedure is called to remove a user from a sitegroup in a site collection. **proc_SecRemoveUserFromSiteGroup** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecRemoveUserFromSiteGroup (
@SiteId                 uniqueidentifier
,@GroupId               int
,@UserIDToBeDeleted     int
,@UserID                int
,@SiteAdmin             bit
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection that will be queried for the requested site group.

@GroupId: The identifier for the site group that the user specified by *@UserIDToBeDeleted* is to be removed from.

@UserIDToBeDeleted: The **User Identifier** (section 2.2.1.24) for the user to remove from the specified sitegroup. If *@UserId* and *@UserIdToBeDeleted* refer to the same user, and the site group permits users to remove themselves from site group membership, then the user specified by *@UserIdToBeDeleted* MUST be removed from the site group.

@UserID: The **User Identifier** for the current user who is removing the user to be deleted from the specified sitegroup. If *@UserId* and *@UserIdToBeDeleted* refer to the same user and the site group

permits site group **members** to edit membership, then the user specified by *@UserIdToBeDeleted* is removed from the sitegroup.

@SiteAdmin: A bit flag specifying whether the user specified by *@UserID* is a site collection administrator of the site collection specified by *@SiteId*. If *@SiteAdmin* is set to 1, then **proc_SecRemoveUserFromSiteGroup** MUST remove the user specified by *@UserIDToBeDeleted* from the sitegroup.

Return Values: The **proc_SecRemoveUserFromSiteGroup** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
5	The current user has insufficient authority to remove the user from the site group.

Result Sets: The **proc_SecRemoveUserFromSiteGroup** stored procedure MUST NOT return any result set.

3.1.5.84 proc_SecRemoveUserFromSiteGroupByLogin

The **proc_SecRemoveUserFromSiteGroupByLogin** stored procedure is called to remove a user identified by login name from a specified sitegroup. **proc_SecRemoveUserFromSiteGroupByLogin** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecRemoveUserFromSiteGroupByLogin (
@SiteId          uniqueidentifier
,@GroupId        int
,@LoginName      nvarchar(255)
,@UserID         int
,@SiteAdmin      bit
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection that contains the sitegroup from which the user is to be removed.

@GroupId: The identifier of the sitegroup that contains the user to be removed.

@LoginName: The login name of the user to be removed from the sitegroup.

@UserID: The **User Identifier** (section 2.2.1.24) of the current user.

@SiteAdmin: A bit flag specifying whether the user specified by *@UserID* is a site administrator of the site specified by *@SiteId*. If *@SiteAdmin* is set to 1, and *@LoginName* specifies an existing user, **proc_SecRemoveUserFromSiteGroupByLogin** MUST remove the user specified by *@LoginName* from the site group.

Return Values: The **proc_SecRemoveUserFromSiteGroupByLogin** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
5	The current user doesn't have sufficient permission to remove the specified user.
1317	UserId not found for site collection or login name.

Result Sets: The **proc_SecRemoveUserFromSiteGroupByLogin** stored procedure MUST NOT return any result set.

3.1.5.85 proc_SecRemoveUserFromWeb

The **proc_SecRemoveUserFromWeb** stored procedure is called to remove a user belonging to a particular **site**, given its UserId. **proc_SecRemoveUserFromWeb** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc SecRemoveUserFromWeb (
@SiteId     uniqueidentifier
,@WebId     uniqueidentifier
,@UserId    int
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the user to be removed.

@WebId: The **Site Identifier** (section 2.2.1.19) of the site.

@UserId: The **User Identifier** (section 2.2.1.24) of the user to be removed.

Return Values: The **proc_SecRemoveUserFromWeb** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
4335	Cannot successfully remove the last user from the administrators **group**.

Result Sets: The **proc_SecRemoveUserFromWeb** stored procedure MUST return a **Site Acl Result Set** (section 2.2.4.45).

3.1.5.86 proc_SecRemoveUserFromWebByLogin

The **proc_SecRemoveUserFromWebByLogin** stored procedure is called to remove a user identified by login name from a specified **site**. **proc_SecRemoveUserFromWebByLogin** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecRemoveUserFromWebByLogin (
@WebId          uniqueidentifier
,@SiteId        uniqueidentifier
,@LoginName     nvarchar(255)
);
```

@WebId: The **Site Identifier** (section 2.2.1.19) of the site.

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection that contains the site from which the user is to be removed.

@LoginName: The login name of the user to be removed from the site.

Return Values: The **proc_SecRemoveUserFromWebByLogin** stored procedure MUST return one of the following integer return codes.

Value	Description
0	Successful execution.
4335	The last remaining user in this **group** or resource cannot be deleted.

Result Sets:

The **proc_SecRemoveUserFromWebByLogin** stored procedure MUST return a **User Identifier Result Set** (section 2.2.4.58). It MUST return a single row holding a single column which MUST contain the **User Identifier** (section 2.2.1.24) of the removed **principal (1)**.

The **proc_SecRemoveUserFromWebByLogin** stored procedure MUST return a **Site Acl Result Set** (section 2.2.4.45) for the specified site.

3.1.5.87 proc_SecRemoveUserFromWebGroupByLogin

The **proc_SecRemoveUserFromWebGroupByLogin** stored procedure is called to remove a user identified by *@LoginName* from a specified site group.
proc_SecRemoveUserFromWebGroupByLogin is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecRemoveUserFromWebGroupByLogin (
@SiteId          uniqueidentifier
,@WebId          uniqueidentifier
,@GroupId        int
,@LoginName      nvarchar(255)
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the user to be removed.

@WebId: The **Site Identifier** (section 2.2.1.19) of the **site**.

@GroupId: An identifier for the site group that the user refers to. *@GroupId* MUST refer to an existing **group**.

@LoginName: The login name of the user to be removed from the site group.

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

Value	Description
0	Successful execution.
1317	The specified user does not exist.
4335	Cannot delete the last remaining item in the Administrator site group.

Result Sets: The **proc_SecRemoveUserFromWebGroupByLogin** stored procedure MUST NOT return any result set.

3.1.5.88 proc_SecRemoveWebGroup

The **proc_SecRemoveWebGroup** stored procedure is called to remove a site group.
proc_SecRemoveWebGroup is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecRemoveWebGroup (
```

```
@SiteId          uniqueidentifier
,@WebId          uniqueidentifier
,@WebGroupId     int
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection that contains the site group to be removed.

@WebId: The **Site Identifier** (section 2.2.1.19) of the **site** to be removed.

@WebGroupId: The identifier of the site group to be removed.

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

Result Sets: The **proc_SecRemoveWebGroup** stored procedure MUST return a **Site Acl Result Set** (section 2.2.4.45).

3.1.5.89 proc_SecResetToUniqueWeb

The **proc_SecResetToUniqueWeb** stored procedure is called to reset the existing roles and permissions for a specified **site** with a default set of role definitions, as specified by input parameters. **proc_SecResetToUniqueWeb** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecResetToUniqueWeb (
@WebId                          uniqueidentifier
,@AuthorID                      int
,@AdminsName                    nvarchar(255)
,@AdminsDescription             nvarchar(512)
,@AuthorsName                   nvarchar(255)
,@AuthorsDescription            nvarchar(512)
,@ContributorsName              nvarchar(255)
,@ContributorsDescription       nvarchar(512)
,@BrowsersName                  nvarchar(255)
,@BrowsersDescription           nvarchar(512)
,@GuestsName                    nvarchar(255)
,@GuestsDescription             nvarchar(512)
,@Acl                           image
,@AnonymousPermMask             int
);
```

@WebId: The **Site Identifier** (section 2.2.1.19) of the site.

@AuthorID: The **User Identifier** (section 2.2.1.24) of the administrator for the site.

@AdminsName: The display name of the administrator for the site.

@AdminsDescription: The description of the administrator for the site.

@AuthorsName: The display name of the web designer for the site.

@AuthorsDescription: The description of the web designer for the site.

@ContributorsName: The display name of the contributor for the site.

@ContributorsDescription: The description of the contributor for the site.

@BrowsersName: The display name of the reader for the site.

@BrowsersDescription: The description of the reader for the site.

@GuestsName: The display name of the guest for the site.

@GuestsDescription: The description of the guest for the site.

@Acl: The binary serialization of the **WSS ACL Format** (section 2.2.3.2) ACL for the site.

@AnonymousPermMask: A **WSS Rights Mask** (section 2.2.2.10) that indicates the permissions granted to a user that is anonymous, or has no specific permissions, to the site.

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

Value	Description
0	Successful execution.
3	Site does not exist.

Result Sets: The **proc_SecResetToUniqueWeb** stored procedure MUST NOT return any result set.

3.1.5.90 proc_SecSetGroupMembershipTokenAndEnsureWebMembership

The **proc_SecSetGroupMembershipTokenAndEnsureWebMembership** stored procedure is called to update the **group** membership token or to add a new **member** to the specified **site**. **proc_SecSetGroupMembershipTokenAndEnsureWebMembership** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecSetGroupMembershipTokenAndEnsureWebMembership (
@SiteId          uniqueidentifier
,@WebId          uniqueidentifier
,@UserId         int
,@STSToken       varbinary(4000)
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection.

@WebId: The **Site Identifier** (section 2.2.1.19) of the site for which web membership is to be updated.

@UserId: The **User Identifier** (section 2.2.1.24) for the current user. If this parameter is NULL, the update MUST fail.

@STSToken: An identifier that identifies the authentication process applied to a user. It contains a unique string for the user (SID) and a list of attributes for the user, specifying the site group membership of the specified user.

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

Value	Description
0	Successful execution.
3	The specified website was not found.

Result Sets: The **proc_SecSetGroupMembershipTokenAndEnsureWebMembership** stored procedure MUST NOT return any result set.

3.1.5.91 proc_SecSetSiteGroupProperties

The **proc_SecSetSiteGroupProperties** is called to update properties of a sitegroup. **proc_SecSetSiteGroupProperties** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc SecSetSiteGroupProperties (
@SiteId          uniqueidentifier
,@GroupId        int
,@Title          nvarchar(255)
,@Description    nvarchar(512)
,@OwnerID        int
,@OwnerIsUser    bit
,@UserID         int
,@SiteAdmin      bit
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection that contains the sitegroup to update.

@GroupId: The identifier for the site group to update.

@Title: The display name for the site group. If this parameter is NULL, the update MUST fail and the return code MUST be 80.

@Description: The description for the site group. This parameter can be NULL.

@OwnerID: The identifier of a user, domain group, or site group to be set as the owner of the site group. This parameter MUST contain the identifier of an existing user, domain group, or site group.

@OwnerIsUser: A bit that specifies whether the value in *@OwnerID* is a user or a site group. When *@OwnerIsUser* is set to 1, the site group owner is a user in the site collection. When *@OwnerIsUser* is set to 0, the owner is a site group.

@UserID: The **User Identifier** (section 2.2.1.24) of the current user.

@SiteAdmin: A bit flag specifying whether the current user has site collection administration permissions. This parameter can be NULL.

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

Value	Description
0	Successful execution.
5	The current user does not have sufficient permissions to manage the site group.
80	An update error occurred or a parameter was invalid.

Result Sets: The **proc_SecSetSiteGroupProperties** stored procedure MUST NOT return any result set.

3.1.5.92 proc_SecSetWebGroupProperties

The **proc_SecSetWebGroupProperties** stored procedure is called to update properties of a site group. **proc_SecSetWebGroupProperties** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecSetWebGroupProperties (
@SiteId          uniqueidentifier
,@WebId          uniqueidentifier
```

```
,@GroupId        int
,@Title          nvarchar(255)
,@Description    nvarchar(512)
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection that contains the site group to update.

@WebId: The **Site Identifier** (section 2.2.1.19) of the **site** that contains the site group to update.

@GroupId: The identifier for the site group to update.

@Title: The display name for the site group. If this parameter is NULL, the update MUST fail and the return code MUST be 80.

@Description: The description for the site group. This parameter can be NULL.

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

Value	Description
0	Successful execution.
80	An update error occurred or a parameter was invalid.

Result Sets: The **proc_SecSetWebGroupProperties** stored procedure MUST NOT return any result set.

3.1.5.93 proc_SecSetWebRequestAccess

The **proc_SecSetWebRequestAccess** stored procedure is called to set the request access email address for a specified **site**. Site access requests to the particular site are routed to the specified request access email address. **proc_SecSetWebRequestAccess** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc SecSetWebRequestAccess (
@WebId                  uniqueidentifier
,@RequestAccessEmail    nvarchar(255)
);
```

@WebId: The **Site Identifier** (section 2.2.1.19) of the site to set the request access email address.

@RequestAccessEmail: The new request access email address.

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

Result Sets: The **proc_SecSetWebRequestAccess** stored procedure MUST NOT return any result set.

3.1.5.94 proc_SecUpdateListAcl

The **proc_SecUpdateListAcl** stored procedure is called to update the ACL for a list. **proc_SecUpdateListAcl** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecUpdateListAcl (
```

```
@WebId           uniqueidentifier
,@ListId         uniqueidentifier
,@Acl            image
,@AnonymousMask  tPermMask
);
```

@WebId: The **Site Identifier** (section 2.2.1.19) of the **site** that contains the **list (1)**.

@ListId: The **List Identifier** (section 2.2.1.10) of the list (1).

@Acl: The binary serialization of the **WSS ACL Format** (section 2.2.3.2) for the scope in effect.

@AnonymousMask: Contains the **WSS Rights Mask** (section 2.2.2.10) that applies to an anonymous user in the scope of the list (1).

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

Result Sets: The **proc_SecUpdateListAcl** stored procedure MUST NOT return any result set.

3.1.5.95 proc_SecUpdateUser

The **proc_SecUpdateUser** stored procedure is called to update the display name, email address, notes, and administrative privileges listed in the user information that is associated with a **principal (1)**. **proc_SecUpdateUser** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecUpdateUser (
@SiteId                uniqueidentifier
,@UserIdToUpdate       int
,@Title                nvarchar(255)
,@Email                nvarchar(255)
,@Notes                nvarchar(1023)
,@SiteAdmin            bit
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection that contains the principal (1) to be updated.

@UserIdToUpdate: The **User Identifier** (section 2.2.1.24) of the principal (1) whose user information is to be updated.

@Title: The display name to be set in the principal's user information. If this parameter is NULL, the user information update MUST fail.

@Email: The email address to be updated in the principal's user information. If this parameter is NULL, the user information update MUST fail.

@Notes: The notes text to be updated in the principal's user information. If this parameter is NULL, the user information update MUST fail.

@SiteAdmin: A bit specifying whether or not to set the principal (1) as a site collection administrator. When this parameter is set to "1", if the current user is a site administrator and the principal (1) is not a domain group, the principal MUST be set as a site collection administrator. When the *@SiteAdmin* parameter is set to "0", **proc_SecUpdateUser** MUST clear the site collection administrator privilege in the user information for the principal (1). If the parameter is NULL, **proc_SecUpdateUser** MUST NOT make changes to the site collection administrator flag.

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

Value	Description
0	Successful execution.
1399	The principal (1) cannot be a site collection administrator.
4335	Cannot successfully remove the site collection administrator privilege from the principal.

Result Sets: The **proc_SecUpdateUser** stored procedure MUST NOT return any result set.

3.1.5.96 proc_SecUpdateWebAcl

The **proc_SecUpdateWebAcl** stored procedure is called to update any **access control list (ACL)** for a group of permission-sharing sites. **proc_SecUpdateWebAcl** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_SecUpdateWebAcl (
@WebId                          uniqueidentifier
,@Acl                           image
,@AnonymousMask                 tPermMask
,@EffectiveAnonymousMask        tPermMask = null OUTPUT
);
```

@WebId: The **Site Identifier** (section 2.2.1.19) of the **site** to be updated.

@Acl: The access control list (ACL) for the site in the **WSS ACL Format** (section 2.2.3.2). When ACL is NULL, it gives out error as Cannot insert NULL values and update fails.

@AnonymousMask: A **WSS Rights Mask** (section 2.2.2.10) specifying the rights granted to an anonymous user or a user who has no specific rights on this document. If this value is NULL, then **proc_SecUpdateWebAcl** MUST NOT change the existing anonymous permissions mask. If this value is 0, the anonymous permissions mask for all lists belonging to this site is changed to 0.

@EffectiveAnonymousMask: An output parameter that specifies the effective anonymous permissions mask (**WSS Rights Mask**) for the site.

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

Result Sets: The **proc_SecUpdateWebAcl** stored procedure MUST NOT return any result set.

3.1.5.97 proc_UncheckoutDocument

The **proc_UncheckoutDocument** stored procedure is called to release the short-term lock on a document, or to release a checked out document. The current document is set to the most recent previous published or **draft** version, or to the version currently checked out, depending on the document state and **proc_UncheckoutDocument** parameters. proc_UncheckoutDocument is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_UncheckoutDocument (
@SiteId          uniqueidentifier
,@WebId          uniqueidentifier
,@DirName        nvarchar(256)
,@LeafName       nvarchar(128)
,@UserId         int
,@ShortTerm      bit
,@Force          bit
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the document to have its short-term lock released or its checkout reverted.

@WebId: The **Site Identifier** (section 2.2.1.19) of the **site** containing the document.

@DirName: The document's directory name.

@LeafName: The document's leaf name.

@UserId: The **User Identifier** (section 2.2.1.24) of the current user.

@ShortTerm: Specifies whether to release a short-term lock on the document or to revert a check-out of the document. This parameter MUST NOT be NULL.

- If *@ShortTerm* is set to 1 and either the current user holds a short-term lock on the document or *@Force* is set to 1, the lock MUST be release.
- If *@ShortTerm* is set to 0 and either the current user has checked out the document or *@Force* is set to 1, the checkout MUST be reverted, **proc_UncheckoutDocument** MUST discard any draft document created for the checkout and MUST set the most recent previous draft or **published version** as the current version.

@Force: Specifies whether to force the release of a short-term lock or the reversion of a checkout of the document held by another user. If this parameter is set to 1, the short-term lock or checkout of the document (as specified by the *@ShortTerm* parameter) MUST be released, and the most recent previous draft or published version MUST be reverted to the current version, even if the current user is not the holder of the short-term lock or the user the document is checked out to. If *@Force* is not set to 1, the short-term lock or check-out of the document will not be released, and the most recent previous draft or published version will not revert to the current version if the current user is not the holder of the short-term lock or the user the document is checked out to.

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

Value	Description
0	Successful execution.
3	The document was not found at the specified location.
158	The document is already unlocked.
173	Another user has checked out the document or has a short-term lock on the document, and *@Force* is not set to 1.

Result Sets:

The **proc_UncheckoutDocument** stored procedure MUST return a **Link Info Single Doc Result Set** (section 2.2.4.29). The **Link Info Single Doc Result Set** returns information about all forward links and backward links associated with the document. The **Link Info Single Doc Result Set** MUST be returned and MUST contain one row for each forward link and backward link associated with the specified document.

The **proc_UncheckoutDocument** stored procedure MUST return a **Document Metadata Result Set** (section 2.2.4.13). The **Document Metadata Result Set** returns the metadata for the specified document. The **Document Metadata Result Set** MUST be returned and MUST contain one row for the specified document.

3.1.5.98 proc_UpdateDocument

The **proc_UpdateDocument** stored procedure is called to update the metadata and contents of a document. **proc_UpdateDocument** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc UpdateDocument (
@DocSiteId                      uniqueidentifier
,@DocWebId                      uniqueidentifier
,@DocDirName                    nvarchar(256)
,@DocLeafName                   nvarchar(128)
,@DocContent                    image
,@DocMetaInfo                   image
,@DocSize                       int
,@DocMetainfoSize               int
,@DocDirty                      bit
,@DocVersion                    int
,@DocFlags                      tinyint
,@DocIncomingDTM                datetime
,@ThicketMainFile               bit
,@GetWebListForNormalization    bit
,@VersioningEnabled             bit
,@UserId                        int
,@AttachmentOp                  int
,@PutFlags                      int
,@UpdateFlags                   int
,@CharSet                       int
,@WebParts                      bit
,@VirusVendorID                 int
,@VirusStatus                   int
,@VirusInfo                     nvarchar(255)
,@@DocId                        uniqueidentifier OUTPUT
,@@DoclibRowId                  int OUTPUT
,@@DocDTM                       datetime OUTPUT
,@@DocUIVersion                 int OUTPUT
,@@DocFlagsOut                  tinyint OUTPUT
,@ChunkSize                     int
,@@DocTextptr                   varbinary(16) OUTPUT
);
```

@DocSiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection containing the document to be updated.

@DocWebId: The **Site Identifier** (section 2.2.1.19) for the site containing the document.

@DocDirName: The directory name of the document location.

@DocLeafName: The leaf name of the document location.

@DocContent: The document stream of the content of the document.

@DocMetaInfo: The document metadata **stream** for the document.

@DocSize: The size, in bytes, of the document.

@DocMetainfoSize: The size, in bytes, of the document's metadata.

@DocDirty: A bit flag specifying whether the document has dependencies, such as links to other items, that have to be updated. If this parameter is set to 1, the document has dependencies to be updated. This flag MUST be stored by the back-end database server for subsequent updating of the dependent information.

@DocVersion: The original internal version counter value of the document to be updated. This value is incremented whenever a change is made to this document, and is used for internal conflict detection. If this value does not match the current internal version counter value of the document,

proc_UpdateDocument MUST NOT update the document version. If the current version number is NULL, **proc_UpdateDocument** MUST NOT update the document version.

@DocFlags: A **Doc Flags** (section 2.2.2.2) value describing the document to be updated.

@DocIncomingDTM: A timestamp, in UTC format, of the last modification date/time of the document. This value can be NULL, specifying the current time on the back-end database server as the last modification time.

@ThicketMainFile: A bit flag specifying whether the document is a thicket main file. If this parameter is set to 1, the document is a thicket main file and **proc_UpdateDocument** MUST store this information.

@GetWebListForNormalization: A bit flag specifying whether to retrieve the list of sites whose immediate parent site is the **site** containing the document to be updated. If this parameter is set to 1, the procedure MUST return a list of subsite of the document's containing site in a **Subsite List Result Set** (section 2.2.4.54) upon successful completion.

@VersioningEnabled: A bit flag specifying whether UI version numbering is enabled on the document. If this parameter is set to 1, major UI version numbering MUST be enabled on the document.

@UserId: The User Identifier (section 2.2.1.24) of the current user that is making the request to the front-end Web server.

@AttachmentOp: An integer containing **Attachments Flags** (section 2.2.2.1) describing the document.

@PutFlags: A **Put Flags** type (section 2.2.2.5) value that specifies document update options.

@UpdateFlags: A bit field that tracks additional document change options. If this parameter is 1, then links that are no longer applicable after the document is updated MUST be deleted. If this parameter is not 1, it MUST be ignored.

@CharSet: An identifier for the Windows code page to associate with the document. Any Windows code page identifier is valid for *@CharSet*. This value can be NULL, specifying no preference.

@WebParts: A bit flag specifying whether the update to the document includes updates to Web Parts. If this parameter is set to 1, **proc_UpdateDocument** MUST update the document's links to Web Parts.

@VirusVendorID: If provided, specifies the vendor identifier of the virus scanner that processed this document.

@VirusStatus: A **Virus Status** (section 2.2.1.26) value specifying the current virus check status of this document.

@VirusInfo: A string containing a provider-specific message that was returned by the virus scanner when it last processed the document.

@@DocId: Output parameter. The **Document Identifier** (section 2.2.1.4) of the document.

@@DoclibRowId: Output parameter. The identifier of the row representing this document in the document library. This value MUST be zero if the document is not stored in a List.

@@DocDTM: Output parameter. The timestamp of the last modification date of the document. Set to the value of *@DocIncomingDTM*, or to the current UTC date if *@DocIncomingDTM* is NULL.

@@DocUIVersion: Output parameter. The UI version number associated with this document. This value MUST be NULL in the case of a document that does not exist.

@@DocFlagsOut: Output parameter. A **Doc Flags** value, set to the DocFlags describing this document after successful completion of the update.

@ChunkSize: Specifies the size, in bytes, that the document MUST be in order for the content to be included in the Update query. This value can be NULL if the *@DocContent* parameter is NULL.

@@DocTextptr: Output parameter. A varbinary pointer set to the value of *@DocContent* if *@DocSize* is greater than *@ChunkSize*. Otherwise, *@@DocTextptr* is set to NULL.

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

Value	Description
0	Successful execution.
3	The document does not exist or has been deleted.
33	Locking violation.
212	The site containing the document is locked.
1150	Version mismatch with *@DocVersion* prevented document update.
1816	Not enough quota for locking; the document cannot be updated.

Result Sets:

The **proc_UpdateDocument** stored procedure MUST return a **Subsite List Result Set** (section 2.2.4.54). The **Subsite List Result Set** returns a list of URLs for the immediate child subsites of the site containing the newly updated document. When the input parameter *@GetWebListForNormalization* is set to 1 and execution has been successful up to the point of updating the document, the **Subsite List Result Set** MUST be produced. The **Subsite List Result Set** MUST contain one row for each subsite found.

3.1.5.99 proc_UpdateListItem

The **proc_UpdateListItem** stored procedure is called to update an item for a particular **list (1)**. **proc_UpdateListItem** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_UpdateListItem(
      @SiteId                        uniqueidentifier,
      @WebId                         uniqueidentifier,
      @ListId                        uniqueidentifier,
      @ItemId                        int,
      @Size                          int,
      @ItemName                      nvarchar(255) = NULL,
      @UseNvarchar1ItemName          bit = 1,
      @DocFullUrl                    nvarchar(260) = NULL,
      @UserId                        int,
      @TimeNow                       datetime,
      @NeedsAuthorRestriction        bit = 0,
      @PreserveVersion               bit = NULL,
      @IsMeetingsList                bit = NULL,
      @IsIssueList                   bit = NULL,
      @IsNotUserDisplayed            bit = NULL,
      @CheckSchemaVersion            int = NULL,
      @tp_Ordering                   varchar(512) = NULL,
      @tp HasAttachment              bit = NULL,
      @tp_ModerationStatus           int = NULL,
      @tp IsCurrent                  bit = NULL,
      @tp_ItemOrder                  float = NULL,
      @tp_Version                    int = NULL,
```

```
@tp_InstanceID                 int = NULL,
@BumpLastDelete                bit = NULL,
@NoAlert                       bit = 0,
@nvarchar1 nvarchar(255) = NULL,
@nvarchar2 nvarchar(255) = NULL,
@nvarchar3 nvarchar(255) = NULL,
@nvarchar4 nvarchar(255) = NULL,
@nvarchar5 nvarchar(255) = NULL,
@nvarchar6 nvarchar(255) = NULL,
@nvarchar7 nvarchar(255) = NULL,
@nvarchar8 nvarchar(255) = NULL,
@nvarchar9 nvarchar(255) = NULL,
@nvarchar10 nvarchar(255) = NULL,
@nvarchar11 nvarchar(255) = NULL,
@nvarchar12 nvarchar(255) = NULL,
@nvarchar13 nvarchar(255) = NULL,
@nvarchar14 nvarchar(255) = NULL,
@nvarchar15 nvarchar(255) = NULL,
@nvarchar16 nvarchar(255) = NULL,
@nvarchar17 nvarchar(255) = NULL,
@nvarchar18 nvarchar(255) = NULL,
@nvarchar19 nvarchar(255) = NULL,
@nvarchar20 nvarchar(255) = NULL,
@nvarchar21 nvarchar(255) = NULL,
@nvarchar22 nvarchar(255) = NULL,
@nvarchar23 nvarchar(255) = NULL,
@nvarchar24 nvarchar(255) = NULL,
@nvarchar25 nvarchar(255) = NULL,
@nvarchar26 nvarchar(255) = NULL,
@nvarchar27 nvarchar(255) = NULL,
@nvarchar28 nvarchar(255) = NULL,
@nvarchar29 nvarchar(255) = NULL,
@nvarchar30 nvarchar(255) = NULL,
@nvarchar31 nvarchar(255) = NULL,
@nvarchar32 nvarchar(255) = NULL,
@nvarchar33 nvarchar(255) = NULL,
@nvarchar34 nvarchar(255) = NULL,
@nvarchar35 nvarchar(255) = NULL,
@nvarchar36 nvarchar(255) = NULL,
@nvarchar37 nvarchar(255) = NULL,
@nvarchar38 nvarchar(255) = NULL,
@nvarchar39 nvarchar(255) = NULL,
@nvarchar40 nvarchar(255) = NULL,
@nvarchar41 nvarchar(255) = NULL,
@nvarchar42 nvarchar(255) = NULL,
@nvarchar43 nvarchar(255) = NULL,
@nvarchar44 nvarchar(255) = NULL,
@nvarchar45 nvarchar(255) = NULL,
@nvarchar46 nvarchar(255) = NULL,
@nvarchar47 nvarchar(255) = NULL,
@nvarchar48 nvarchar(255) = NULL,
@nvarchar49 nvarchar(255) = NULL,
@nvarchar50 nvarchar(255) = NULL,
@nvarchar51 nvarchar(255) = NULL,
@nvarchar52 nvarchar(255) = NULL,
@nvarchar53 nvarchar(255) = NULL,
@nvarchar54 nvarchar(255) = NULL,
@nvarchar55 nvarchar(255) = NULL,
@nvarchar56 nvarchar(255) = NULL,
@nvarchar57 nvarchar(255) = NULL,
@nvarchar58 nvarchar(255) = NULL,
@nvarchar59 nvarchar(255) = NULL,
@nvarchar60 nvarchar(255) = NULL,
@nvarchar61 nvarchar(255) = NULL,
@nvarchar62 nvarchar(255) = NULL,
@nvarchar63 nvarchar(255) = NULL,
@nvarchar64 nvarchar(255) = NULL,
@int1 int = NULL,
@int2 int = NULL,
```

```
@int3 int = NULL,
@int4 int = NULL,
@int5 int = NULL,
@int6 int = NULL,
@int7 int = NULL,
@int8 int = NULL,
@int9 int = NULL,
@int10 int = NULL,
@int11 int = NULL,
@int12 int = NULL,
@int13 int = NULL,
@int14 int = NULL,
@int15 int = NULL,
@int16 int = NULL,
@float1 float = NULL,
@float2 float = NULL,
@float3 float = NULL,
@float4 float = NULL,
@float5 float = NULL,
@float6 float = NULL,
@float7 float = NULL,
@float8 float = NULL,
@float9 float = NULL,
@float10 float = NULL,
@float11 float = NULL,
@float12 float = NULL,
@float13 float = NULL,
@float14 float = NULL,
@float15 float = NULL,
@float16 float = NULL,
@float17 float = NULL,
@float18 float = NULL,
@float19 float = NULL,
@float20 float = NULL,
@float21 float = NULL,
@float22 float = NULL,
@float23 float = NULL,
@float24 float = NULL,
@float25 float = NULL,
@float26 float = NULL,
@float27 float = NULL,
@float28 float = NULL,
@float29 float = NULL,
@float30 float = NULL,
@float31 float = NULL,
@float32 float = NULL,
@datetime1 datetime = NULL,
@datetime2 datetime = NULL,
@datetime3 datetime = NULL,
@datetime4 datetime = NULL,
@datetime5 datetime = NULL,
@datetime6 datetime = NULL,
@datetime7 datetime = NULL,
@datetime8 datetime = NULL,
@datetime9 datetime = NULL,
@datetime10 datetime = NULL,
@datetime11 datetime = NULL,
@datetime12 datetime = NULL,
@datetime13 datetime = NULL,
@datetime14 datetime = NULL,
@datetime15 datetime = NULL,
@datetime16 datetime = NULL,
@bit1 bit = NULL,
@bit2 bit = NULL,
@bit3 bit = NULL,
@bit4 bit = NULL,
@bit5 bit = NULL,
@bit6 bit = NULL,
@bit7 bit = NULL,
```

```
        @bit8 bit = NULL,
        @bit9 bit = NULL,
        @bit10 bit = NULL,
        @bit11 bit = NULL,
        @bit12 bit = NULL,
        @bit13 bit = NULL,
        @bit14 bit = NULL,
        @bit15 bit = NULL,
        @bit16 bit = NULL,
        @uniqueidentifier1 uniqueidentifier = NULL,
        @ntext1 ntext = NULL,
        @ntext2 ntext = NULL,
        @ntext3 ntext = NULL,
        @ntext4 ntext = NULL,
        @ntext5 ntext = NULL,
        @ntext6 ntext = NULL,
        @ntext7 ntext = NULL,
        @ntext8 ntext = NULL,
        @ntext9 ntext = NULL,
        @ntext10 ntext = NULL,
        @ntext11 ntext = NULL,
        @ntext12 ntext = NULL,
        @ntext13 ntext = NULL,
        @ntext14 ntext = NULL,
        @ntext15 ntext = NULL,
        @ntext16 ntext = NULL,
        @ntext17 ntext = NULL,
        @ntext18 ntext = NULL,
        @ntext19 ntext = NULL,
        @ntext20 ntext = NULL,
        @ntext21 ntext = NULL,
        @ntext22 ntext = NULL,
        @ntext23 ntext = NULL,
        @ntext24 ntext = NULL,
        @ntext25 ntext = NULL,
        @ntext26 ntext = NULL,
        @ntext27 ntext = NULL,
        @ntext28 ntext = NULL,
        @ntext29 ntext = NULL,
        @ntext30 ntext = NULL,
        @ntext31 ntext = NULL,
        @ntext32 ntext = NULL,
        @sql_variant1 sql_variant = NULL,
        @error_sql_variant1 int = 0,
        @sql_variant2 sql_variant = NULL,
        @error sql variant2 int = 0,
        @sql_variant3 sql_variant = NULL,
        @error sql variant3 int = 0,
        @sql_variant4 sql_variant = NULL,
        @error_sql_variant4 int = 0,
        @sql_variant5 sql_variant = NULL,
        @error sql variant5 int = 0,
        @sql_variant6 sql_variant = NULL,
        @error sql variant6 int = 0,
        @sql_variant7 sql_variant = NULL,
        @error_sql_variant7 int = 0,
        @sql variant8 sql variant = NULL,
        @error sql variant8 int = 0)
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection containing the list item to be updated.

@WebId: The **Site Identifier** (section 2.2.1.19) for the **site** containing the list item.

@ListId: The **List Identifier** (section 2.2.1.10) of the list (1) containing the list item.

@ItemId: An integer identifier of the list item.

@Size: The size, in bytes, of the list item row to be updated. It MUST NOT be NULL.

@ItemName: The display name to be updated for the list item.

@UseNvarchar1ItemName: If *@ItemName* is NULL, this bit flag specifies whether to use the content of *@nvarchar1* for the new display name for the list item.

@DocFullUrl: The full store-relative form URL.

@UserId: The **User Identifier** (section 2.2.1.24) for the current user.

@TimeNow: The current time, in UTC format, on the back-end database server.

@NeedsAuthorRestriction: A bit flag specifying whether only the list item's author is permitted to update the list item.

@PreserveVersion: A bit flag specifying whether to preserve the **internal version number** of the list item to be updated. If this parameter is set to 1, the internal version number of the list item MUST NOT be incremented.

@IsMeetingsList: A bit flag specifying whether the list item is contained in a Meetings List (a list with a **List Server Template** (section 2.2.1.12) value of 200).

@IsIssueList: A bit flag specifying whether the list item is contained in an Issue List (a list with a **List Server Template** value of 1100).

@IsNotUserDisplayed: A bit flag specifying whether the display name of the current user is not to be displayed. If this parameter is set to 1, the display name of the current user MUST NOT be used and the string "***" MUST be used instead.

@CheckSchemaVersion: Specifies a schema version number to compare with the list schema version number. If this parameter is not NULL, the version numbers MUST match for successful completion.

@tp_Ordering: Specifies the threading structure for this list item in a previous discussion board list as a concatenation of time stamp values in yyyyMMddHHmmss format.

@tp_HasAttachment: A bit flag specifying whether the list item has an associated attachment.

@tp_ModerationStatus: A **Moderation Status** (section 2.2.1.13) value specifying the current moderation approval status of this list item.

@tp_IsCurrent: A bit flag specifying whether this is the current version of the list item.

@tp_ItemOrder: Specifies the relative positioning order to view the list item when displayed with other list items from the same list.

@tp_Version: A value to compare with the internal counter of the list item.

@tp_InstanceID: If this list item is associated with a particular instance of a recurring meeting, this parameter specifies the integer identifier of that instance. For all other list items, this parameter MUST be NULL.

@BumpLastDelete: A bit flag specifying whether to update the *tp_LastDeleted* (a column in the lists table) property on the list specified by *@ListId*. If set to 0, the *tp_LastDeleted* property MUST NOT be updated.

@NoAlert: A bit flag specifying whether to add an alert event for the update. If *@PreserveVersion* and *@NoAlert* are either NULL or 0, an **event (1)** is added.

The next nine columns are duplicated a variable number of times. Each instance of these individual column names is differentiated by a suffix with a value indicated in the column description, which replaces the placeholder '#' symbol shown as follows.

@nvarchar#: User-defined columns in the list containing values of type nvarchar. There are 64 columns numbered from 1 to 64. If the column contains no data, the value MUST be NULL.

@int#: User-defined columns in the list containing values of type int. There are 16 columns numbered from 1 to 16. If the column contains no data, the value MUST be NULL.

@float#: User-defined columns in the list containing values of type float. There are 32 columns numbered from 1 to 32. If the column contains no data, the value MUST be NULL.

@datetime#: User-defined columns in the list containing values of type datetime. There are 16 columns numbered from 1 to 16. If the column contains no data, the value MUST be NULL.

@bit#: User-defined columns in the list containing values of type bit. There are 16 columns numbered from 1 to 16. If the column contains no data, the value MUST be NULL.

@uniqueidentifier1: A user-defined column in the list containing values of type uniqueidentifier. If the column contains no data, the value MUST be NULL.

@ntext#: User-defined columns in the list containing values of type ntext. There are 32 columns numbered from 1 to 32. If the column contains no data, the value MUST be NULL.

@sql_variant#: User-defined columns in the list containing values of type *sql_variant*. There are 8 columns numbered from 1 to 8. If the column contains no data, the value MUST be NULL.

@error_sql_variant#: An integer specifying the type to be applied to the corresponding values specified as arguments for the parameter *@sql_variant#*. There are eight columns numbered from 1 to 8. Valid values for this parameter are listed in the following table.

Value	Description
1	Convert the argument value to a varbinary(2).
2	Convert the argument value to a bit.
3	Convert the argument value to a float.
4	Convert the argument value to datetime.

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

Value	Description
0	Successful execution.
3	WebUrl not found.
5	The list to be updated has an attachment.
13	The list item is not valid.
87	Error occurred during a table update operation.
212	The database for the site is locked.
1150	The value of *@tp_Version* mismatches with the internal version of the list item.

Value	Description
1638	The current schema version of the list does not match the value specified in *@CheckSchemaVersion*.
1639	It is not the current version of the list item.
1816	The site collection has exceeded its allocated size quota.
4005	The specified list item was not found.

Result Sets: The **proc_UpdateListItem** stored procedure MUST return the **Item Update Result Set** (section 2.2.4.26).

3.1.5.100 proc_UpdateListSettings

The **proc_UpdateListSettings** stored procedure is called to update list metadata. **proc_UpdateListSettings** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_UpdateListSettings (
@SiteId                         uniqueidentifier
,@WebId                         uniqueidentifier
,@ListId                        uniqueidentifier
,@BaseType                      int
,@ServerTemplate                int
,@Title                         nvarchar(255)
,@ReadSecurity                  int
,@WriteSecurity                 int
,@NewDefaultView                uniqueidentifier
,@Description                   ntext
,@TemplateDirName               nvarchar(256)
,@TemplateLeafName              nvarchar(128)
,@Fields                        ntext
,@Direction                     int
,@EventSinkAssembly             nvarchar(255)
,@EventSinkClass                nvarchar(255)
,@EventSinkData                 nvarchar(255)
,@Flags                         int
,@ImageUrl                      nvarchar(255)
,@ThumbnailSize                 int
,@WebImageWidth                 int
,@WebImageHeight                int
,@Version                       int
,@ConvertToGlobalMtgDataList    bit
,@EmailInsertsFolder            nvarchar(255)
,@EmailInsertsLastSyncTime      nvarchar(50)
,@NavBarEid                     int
,@AddToNavBar                   bit
);
```

@SiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the **list (1)** whose metadata is being updated.

@WebId: The **Site Identifier** (section 2.2.1.19) of the **site** containing the list (1).

@ListId: The **List Identifier** (section 2.2.1.10) of the list (1) to be updated.

@BaseType: The **List Base** type (section 2.2.1.9) value of the list. If *@ConvertToGlobalMtgDataList* is equal to 1, then this value MUST be 1. Otherwise, the list data will not be converted to global meeting list data.

@ServerTemplate: The identifier for the **List Server Template** (section 2.2.1.12) defining the base structure of this list (1). If *@ConvertToGlobalMtgDataList* is equal to 1, the **List Server Template** MUST NOT be 200 (Meeting), 202 (Meeting user) or 212 (Homepage).

@Title: The new title for the list (1). If this parameter is NULL, then the current value MUST NOT be changed.

@ReadSecurity: The security policy that MUST be set for read access on list items in the list (1). Valid values for this parameter are listed in the following table.

Value	Description
1	Users with read permissions can read all list items.
2	Users with read permissions can only read their own items.
NULL	**proc_UpdateListSettings** MUST make no changes to read security for the list.

@WriteSecurity: The security policy that MUST be set for write access on list items in the list. Valid values for this parameter are listed in the following table.

Value	Description
1	Users with write permissions have write access to all list items.
2	Users with write permissions have write access to their own items only.
4	Users have no write access to any list items.
NULL	**proc_UpdateListSettings** MUST make no changes to write security for the list.

@NewDefaultView: A **View Identifier** (section 2.2.1.25) of a defined **list view** to be displayed by default when navigating to the list (1). If this parameter is NULL, then the current value MUST NOT be changed.

@Description: A user-provided readable text description for the list (1). If this parameter is NULL, then the current value MUST NOT be changed.

@TemplateDirName: The directory name for the **document template** associated with the list (1). This parameter can be NULL. The parameter MUST be ignored if **@TemplateLeafName** is NULL or is an **empty string**.

@TemplateLeafName: The leaf name for the document template associated with the list (1). If this parameter is an empty string, then the current document template value is changed to "none". If this parameter is NULL, then the current value MUST NOT be changed.

@Fields: The field definitions for the list (1), consisting of the concatenation of an implementation-specific version string for the list template, followed by an XML representation of the field definitions. If this parameter is NULL, then the current value MUST NOT be changed.

@Direction: The direction of text flow for front-end Web server elements presented by this list (1). Valid values for this parameter are listed in the following table.

Value	Description
0	No explicit direction is specified.
1	Text flow is left to right.
2	Text flow is right to left.

Value	Description
NULL	**proc_UpdateListSettings** MUST make no changes to the current value.

@EventSinkAssembly: A .NET assembly name for an event sink handler for the list (1). If this parameter is NULL, **proc_UpdateListSettings** MUST make no changes to this value.

@EventSinkClass: A .NET assembly class identifier for an event sink handler for the list (1). If this parameter is NULL, **proc_UpdateListSettings** MUST make no changes to this value.

@EventSinkData: The event sink data for an event sink handler for the list (1). If NULL, **proc_UpdateListSettings** MUST make no changes to this value.

@Flags: A **List Flags** (section 2.2.2.4) value for the list (1). If NULL, **proc_UpdateListSettings** MUST make no changes to this value.

@ImageUrl: The URL of the image used to represent this list (1). If NULL, **proc_UpdateListSettings** MUST make no changes to this value.

@ThumbnailSize: A value specifying the width, in pixels, used by lists of type Image Library to determine the rendering size of an image thumbnail. If NULL, **proc_UpdateListSettings** MUST make no changes to this value.

@WebImageWidth: A value specifying the width, in pixels, used by lists of type Image Library to determine the rendering width of an image. If NULL, **proc_UpdateListSettings** MUST make no changes to this value.

@WebImageHeight: A value specifying the height, in pixels, used by lists of type Image Library to determine the rendering height of an image. If NULL, **proc_UpdateListSettings** MUST make no changes to this value.

@Version: An integer used to match on the internal version counter of the list (1) to be updated. This is used to prevent updates with out-of-date data. On successful execution, the internal version counter value of the list will be incremented by 1.

@ConvertToGlobalMtgDataList: A value to indicate that the list data SHOULD be converted to global meeting list data. If this parameter is set to 1, the list data will be converted to global meeting list data.

@EmailInsertsFolder: A URL fragment specifying the directory on the configured email inserts server which SHOULD be inspected for new email messages to be processed for this list (1). If NULL, **proc_UpdateListSettings** MUST make no changes to this value.

@EmailInsertsLastSyncTime: The time stamp, in UTC format, when the email inserts server was last processed for new email messages. If NULL, **proc_UpdateListSettings** MUST make no changes to this value. If this parameter contains an empty string, then the current time will be used.

@NavBarEid: A **navigation node element identifier** for the parent **navigation node** of the node that will represent the list. If *@AddToNavBar* is NULL or 0, this parameter MUST be ignored. Otherwise, it MUST contain a valid navigation node entity identifier.

@AddToNavBar: A value to specify whether to add or remove this list (1) from the site's navigation bar. If this parameter is set to 0, the list MUST be removed from the site's navigation bar. If NULL, **proc_UpdateListSettings** MUST make no changes to the site's navigation bar. Otherwise, if the list is not in the site's navigation bar, it MUST be added to it. To move nodes from one location to another, the navigation node is first removed and then re-added with the new desired setting.

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

Value	Description
0	Successful execution.
13	*@ConvertToGlobalMtgDataList* is set to 1, but the list template is not valid for using global data.
3	A site with the specified *@WebId* and *@ListId* does not exist.
15	The template document URL (combination of *@TemplateDirName* and *@TemplateLeafName*) is not valid.
52	A list with an identical title exists.
1150	Concurrency violation occurred.

Result Sets: The **proc_UpdateListSettings** stored procedure can return a **Deleted Documents Result Set** (section 2.2.4.8) if *@ConvertToGlobalMtgDataList* is set to 1.

3.1.5.101 proc_UpdateSandboxDocument

The **proc_UpdateSandboxDocument** stored procedure is called to update the contents of a sandbox document. **proc_UpdateSandboxDocument** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_UpdateSandboxDocument (
@DocSiteId       uniqueidentifier
,@DocDirName     nvarchar(256)
,@DocLeafName    nvarchar(128)
,@DocContent     image
,@DocSize        int
,@DocVersion     int
,@UserId         int
,@@DocDTM        datetime OUTPUT
,@ChunkSize      int
,@@DocTextptr    varbinary(16) OUTPUT
);
```

@DocSiteId: The **Site Collection Identifier** (section 2.2.1.17) for the site collection containing the document.

@DocDirName: The directory name of the document.

@DocLeafName: The leaf name of the document.

@DocContent: The document stream of the content of the document.

@DocSize: The size, in bytes, of the document.

@DocVersion: The original internal version counter value of the document.

@UserId: The **User Identifier** (section 2.2.1.24) of the current user that is making the request to the front-end Web server.

@@DocDTM: Output parameter. The timestamp of the last modification date of the document.

@ChunkSize: Specifies the size, in bytes, that the document MUST be in order for the content to be included in the update query.

@@DocTextptr: Output parameter. A varbinary pointer set to the value of *@DocContent* if *@DocSize* is greater than *@ChunkSize*. Otherwise, *@@DocTextptr* is set to NULL.

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

Value	Description
0	Successful execution.
3	The document does not exist or has been deleted.
33	Locking violation because another user has the document checked out.
158	Checkout is enforced, but the document is not checked out.
212	The document is locked.
1150	Version mismatch with *@DocVersion* prevented document update.
1816	Not enough quota for locking; the document cannot be updated.

Result Sets: The **proc_UpdateSandboxDocument** stored procedure MUST NOT return any result set.

3.1.5.102 proc_UrlToWebUrl

The **proc_UrlToWebUrl** stored procedure is called to get the site URL of a site collection. **proc_UrlToWebUrl** is defined using T-SQL syntax, as follows.

```
PROCEDURE proc_UrlToWebUrl (
@WebSiteId     uniqueidentifier
,@Url          nvarchar(260)
);
```

@WebSiteId: The **Site Collection Identifier** (section 2.2.1.17) of the site collection containing the document.

@Url: The store-relative form URL of the document.

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

Value	Description
0	Successful execution.
1168	Site with given *@WebSiteId* not found.

Result Sets: The **proc_UrlToWebUrl** stored procedure MUST return a **Web Url Result Set** (section 2.2.4.65).

3.1.6 Timer Events

If the execution timeout **event (2)** is triggered, the execution of the stored procedure is terminated and the call fails.

3.1.7 Other Local Events

None.

3.2 Client Details

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

3.2.1 Abstract Data Model

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

The front-end Web server can maintain the following sets of data for this protocol within object structures. There is no requirement for the data within these structures to be a complete representation of all data maintained on the back-end web server, but the data sets can be populated as various requests to the back-end database servers 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 (2)**.

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

3.2.2 Timers

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

3.2.3 Initialization

The front-end Web server MUST validate the user who is making the request before calling the stored procedure(s). The **Site Collection Identifier** (section 2.2.1.17) and the **User Identifier** (section 2.2.1.24) for the user making the request are looked up by the front-end Web server before it calls additional stored procedure(s).

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 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 procedure. SQL queries MUST NOT attempt to add, remove, or update data in any table or view in the content or configuration databases, unless explicitly described in this section.

3.2.6 Timer Events

If the connection timeout **event (2)** is triggered, the connection and the stored procedure call fails.

3.2.7 Other Local Events

None.

4 Protocol Examples

This section provides specific example scenarios for end-to-end file and permissions management against WSS. These examples describe in detail the process of communication between the various server components involved in the WSS deployment. In conjunction with the detailed protocol documentation described in the reference documents, this information is intended to provide a comprehensive view of how the WSS front-end Web server communicates with both end-user client and back-end database server systems.

4.1 File: GetDocsMetaInfo RPC

This example describes the requests made and responses returned when a user retrieves the Metainfo for a requested **site**, using the Web Service RPC methods provided by the FrontPage Server Extensions: Website Management Specification [MC-FPSEWM], as might be generated by a Windows client or the Microsoft SharePoint Designer application.

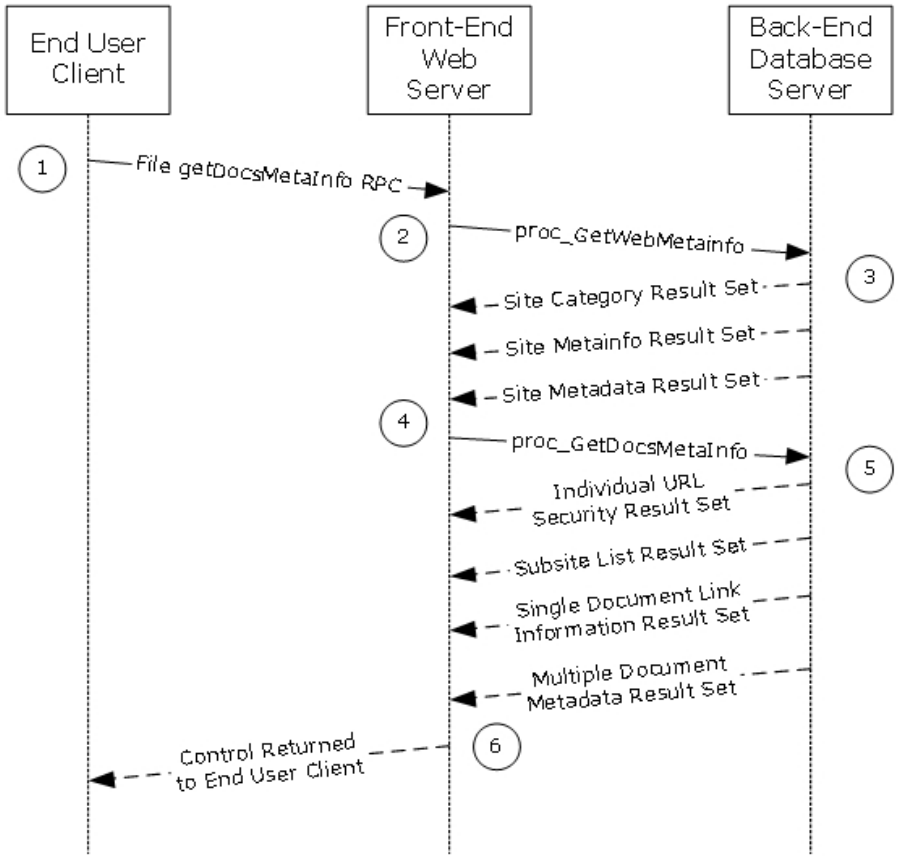


Figure 1: Request-response made to retrieve metadata for a site

This scenario is initiated by a call to the command **getDocsMetaInfo** (described in [MC-FPSEWM] section 3.1.5.3.16). For simplicity's sake, this example assumes that a requested document containing some data has already been uploaded to the Shared Documents folder in the root site (2) of the site collection on the front-end Web server. The following actions happen:

1. The remote procedure call to **getDocsMetaInfo** (see [MC-FPSEWM] section 3.1.5.3.16) is sent to the front-end Web server over HTTP.

2. The front-end Web server in turn requests site metadata information for the requested site from the back-end database server. It does this by calling the **proc_GetWebMetainfo** (section 3.1.5.25) stored procedure using TDS.

3. The back-end database server returns three result sets:

- **Site Category Result Set** (section 2.2.4.46). This returns the unordered set of categories defined for content in the requested site, one category per row.
- **Site Metainfo Result Set** (section 2.2.4.52). This returns metainfo for the requested site.
- **Site Metadata Result Set** (section 2.2.4.51). This returns metadata for the requested site.

4. The front-end Web server then requests metadata information for the documents contained in the requested site. It does this by calling the **proc_GetDocsMetaInfo** stored procedure using TDS.

5. The back-end database server returns four result sets:

- **Individual URL Security Result Set** (section 2.2.4.25). This returns information about the security permissions for the documents contained in the requested site.
- **Subsite List Result Set** (section 2.2.4.54). This returns a list of store-relative form URLs for all subsites of the requested site.
- **Single Doc Link Information Result Set** (section 2.2.4.44). This returns a list of all forward links and backward links for the documents contained in the requested site.
- **Multiple Document Metadata Result Set** (section 2.2.4.36). This returns the metadata for the documents contained in the requested site.

6. A **Server HTTP Response** (see [MC-FPSEWM] section 4.2.1.2) is returned to the user, listing information about the requested site.

4.2 File: Open File OM

This example describes the requests and responses made when the front-end Web server opens an existing file stored as a document in a document library on the back-end database server with a SharePoint Object Model call.

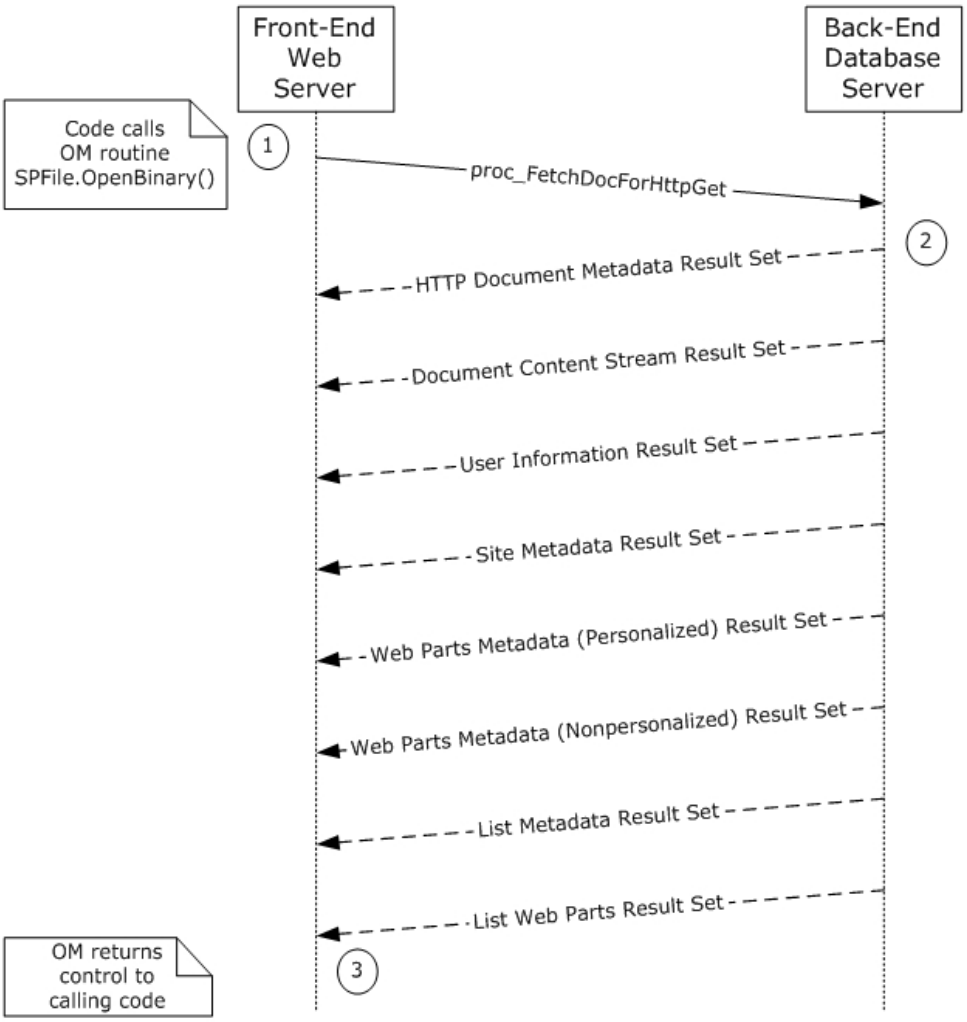


Figure 2: Request-response made to open a document in a document library

This scenario is initiated by a call to the Microsoft.SharePoint.SPFile.OpenBinary() object model command. For simplicity's sake, this example assumes that the file is stored as a document in a document library, and that the requested version is a **draft** created by the same user who is opening the file. This example assumes that:

- The code has already instantiated the site collection (SPSite), Site (SPWeb), and document library (SPList) objects containing the document to be opened.
- Auditing is disabled for the site collection.
- The current user has File Open permissions for the document.
- Site groups in the site collection do not include any domain groups as **members**.

The following actions happen:

1. The front-end Web server builds a dynamic query that calls the **proc_FetchDocForHttpGet** (section 3.1.5.10) stored procedure.

2. The back-end database server returns a return code of 0, and returns the following result sets:

- **HTTP Document Metadata Result Set** (section 2.2.4.24). This returns the document metadata required to further process the document.
- **Document Content Stream Result Set** (section 2.2.4.10). This contains the document's binary **stream** and associated metadata.
- **User Information Result Set** (section 2.2.4.59). Used to establish that the current user has permissions to open the file.
- **Site Metadata Result Set** (section 2.2.4.51). This contains metadata for a **site** or sites.
- **Web Parts Metadata (Personalized) Result Set** (section 2.2.4.64). This contains the core metadata about the Web Parts appearing on the specified document, personalized for a specified user.
- **Web Parts Metadata (Nonpersonalized) Result Set** (section 2.2.4.63). This contains the core metadata about the Web Parts appearing on a document.
- **List Metadata Result Set** (section 2.2.4.32). This contains the metadata associated with the **list (1)** containing the requested document.
- **List Web Parts Result Set** (section 2.2.4.34). This contains information about the Web Parts related to the lists.

3. The OM returns control to the calling application with the array of bytes for the document stream of the requested file.

4.3 Group Add User To Site Group OM

This example describes the requests made and responses returned when a user is added to a site group using SharePoint Object Model code running on the front-end Web server.

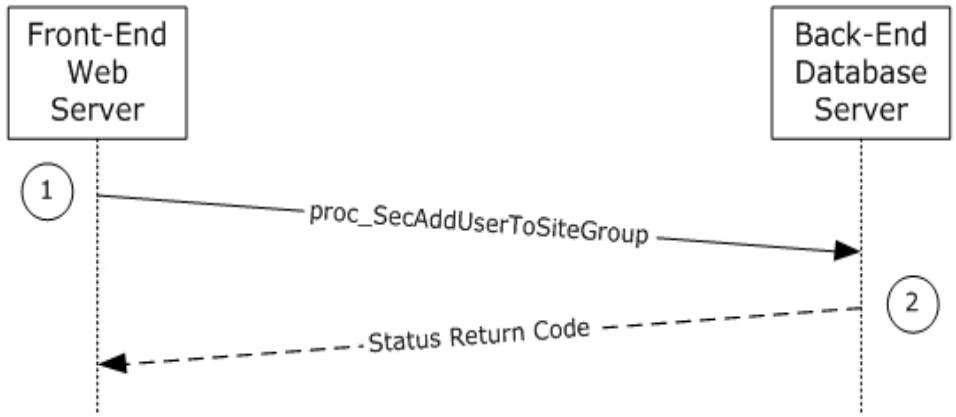


Figure 3: Request-response made to add a user to a site group

This scenario is initiated by a call to the object model command SPGroup.AddUser().

The following actions happen:

1. The front-end Web server attempts to add the user to the site collection using the stored procedure **proc_SecAddUserToSiteGroup** (section 3.1.5.33).

2. The back-end database server returns a single return code indicating the status of the actions.

4.4 Security: Add User to Document Library via Object Model

This example describes the requests made when a user is added to the "contributor" role of a document library that has its own scope for independently managed permissions.

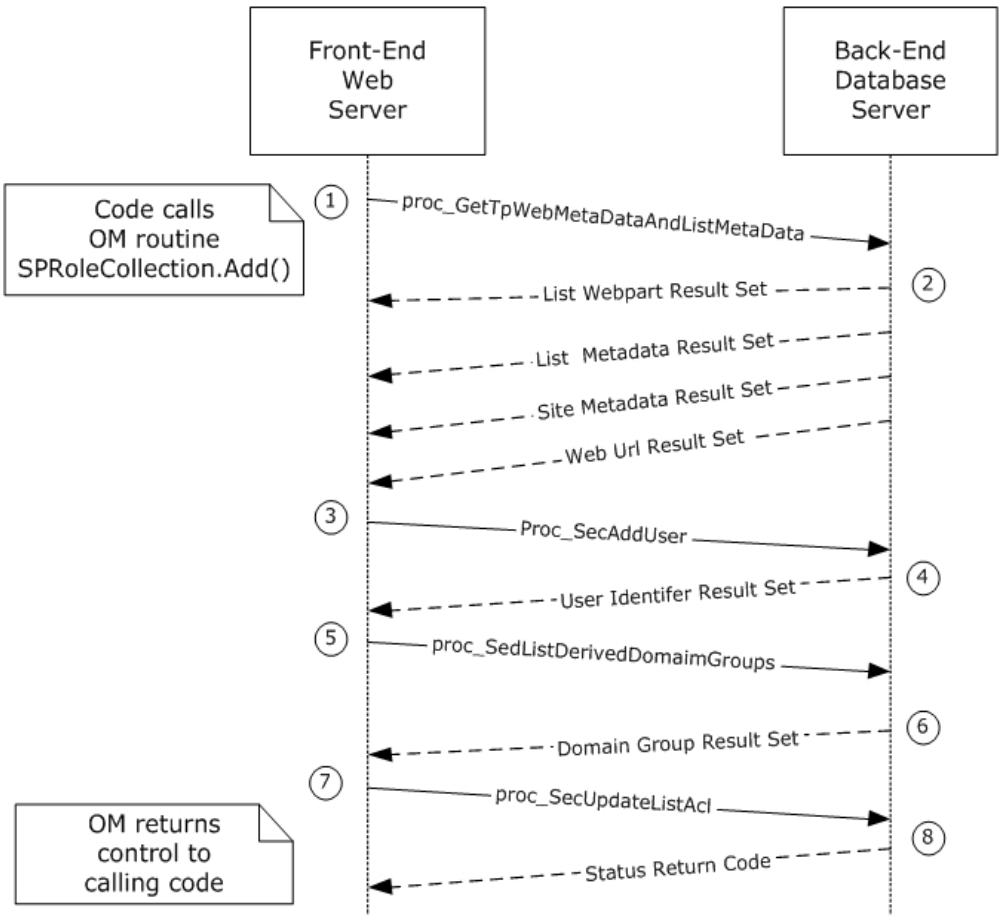


Figure 4: Adding a user to a document library

This scenario is initiated by a call to the object model command SPRoleCollection.Add().

For simplicity's sake, this example assumes that the code has already instantiated the necessary site collection (SPSite), site (SPWeb), and list (SPList) objects in order to construct a representation of the role within the document library to which the user will be added.

The following actions happen:

1. The front-end Web server first retrieves metadata for the requested child **site**. It does this by calling the **proc_GetTpWebMetaDataAndListMetaData** (section 3.1.5.24) stored procedure using TDS.

2. The back-end database server returns the following four result sets:

- **List Webpart Result Set** (section 2.2.4.34). This contains information about the Web Parts related to the lists associated with a specified document.
- **List Metadata Result Set** (section 2.2.4.32). This contains the metadata associated with the **list (1)** containing the requested document.
- **Site Metadata Result Set** (section 2.2.4.51). This contains metadata for the requested site.
- **Web Url Result Set** (section 2.2.4.65). This contains the store-relative form URL of the root of the requested child site.

3. The front-end Web server calls the **proc_SecAddUser** (section 3.1.5.32) to add an entry for a **principal (1)** (a user or domain group) to the list of user information stored in the back-end database server.
4. The back-end database server returns **User Identifier Result Set** (section 2.2.4.58). This contains the identifier of the user.
5. The front-end Web server gets the list of domain groups for a specified **member**. It does this by calling the **proc_SecListDerivedDomainGroups** (section 3.1.5.65) stored procedure.
6. The back-end database server returns **Domain Group Result Set** (section 2.2.4.18). This returns the SystemId of the specified member.
7. The front-end Web server updates the ACL for a list (1). It does this by calling the **proc_SecUpdateListAcl** (section 3.1.5.94) stored procedure.
8. The back-end database server returns a single return code indicating the status of the actions.

4.5 Update List Settings OM

This example describes the interactions made when settings are updated for a particular **list (1)**.

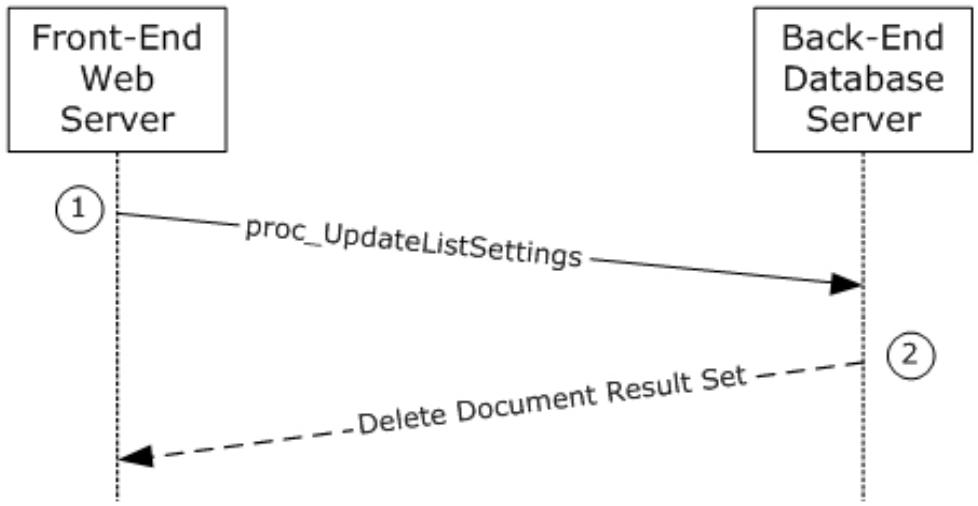


Figure 5: Request-response made to update a list's settings

This scenario is initiated by a call to the object model command SPList.Update().

The following actions happen:

1. The front-end Web server attempts to update the list metadata using the stored procedure **proc_UpdateListSettings** (section 3.1.5.100).

2. The back-end database server returns one result set if the list data is converted to global list meeting data.

- **Deleted Document Result Set** (section 2.2.4.8). This contains information about the deleted documents.

4.6 List Urls

This example describes the request made to get metadata for a document specified by a URL and the documents contained within it, if any.

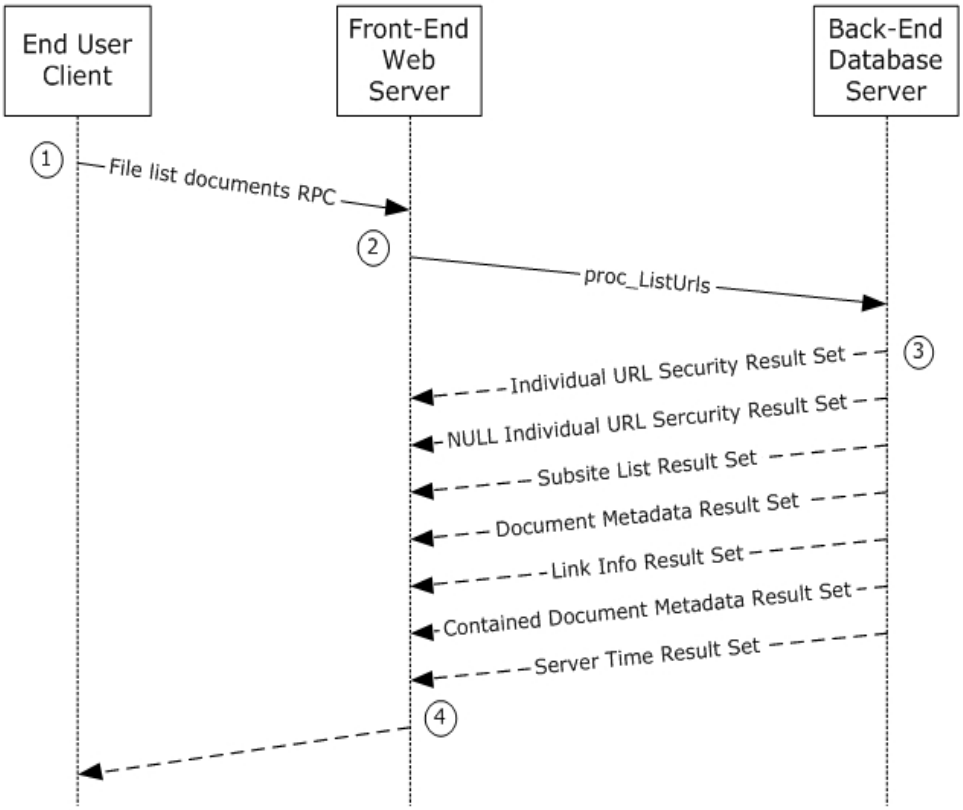


Figure 6: Request-response made to retrieve metadata for a document

1. The Remote Procedure Call to **list documents** (see [MC-FPSEWM] section 3.1.5.3.20) is sent to the front-end Web server over HTTP.

2. The front-end Web server attempts to get metadata for a document specified by a URL by calling stored procedure **proc_ListUrls** (section 3.1.5.28).

3. The back-end database server returns seven result sets.

- **Individual URL Security Result Set** (section 2.2.4.25). This contains security information about the specified document.
- **NULL Individual URL Security Result Set** (section 2.2.4.37). This indicates that a specified document is not found.
- **Subsite List Result Set** (section 2.2.4.54). This contains an ordered list of store-relative form URLs for all subsites of specified **parent site**.
- **Document Metadata Result Set** (section 2.2.4.13). This contains the metadata for the specified document.
- **Link Info Result Set** (section 2.2.4.27). This that contains information about all forward links from and backward links to the documents contained within the specified document.
- **Contained Document Metadata Result Set** (section 2.2.4.7). This contains the metadata information for the documents contained within the specified document.
- **Server Time Result Set** (section 2.2.4.43). This contains the current time from the back-end database server in UTC.

4. An HTTP Response (see [MC-FPSEWM] section 4.2.1.2) is returned to the user, getting the metadata for the document.

4.7 Security: Break Web Inheritance OM

This example describes the requests made to create unique **security role** assignments for a **site**, rather than inheriting the assignments from a parent.

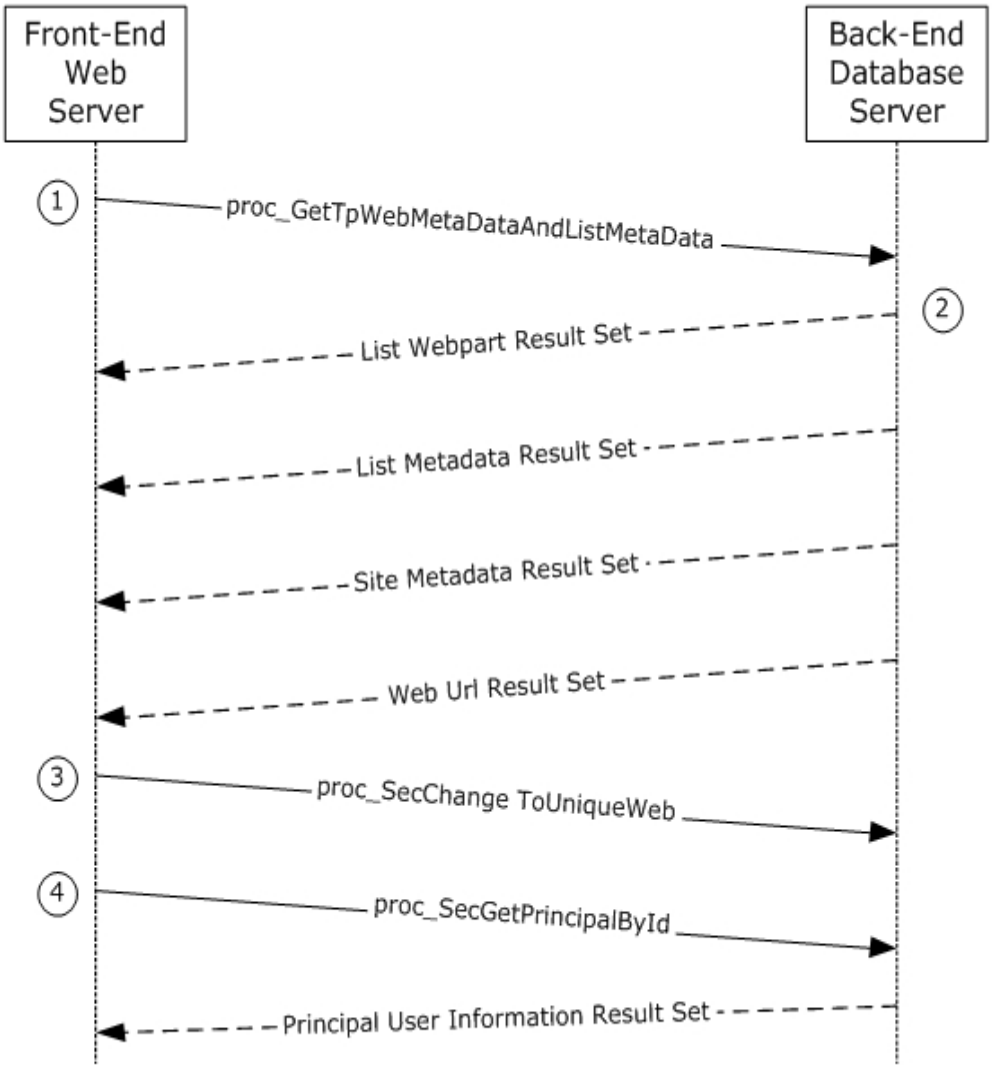


Figure 7: Request-response made to create unique security role assignments for a site

This scenario is initiated by a call to the object model command spweb.BreakRoleInheritance(true). For simplicity's sake, this example assumes that the code has already instantiated the site collection (SPSite), the parent site and child subsite (SPWeb) objects for this session, and the child subsite is initially in the same scope as its **parent site**.

The following actions happen:

1. The front-end Web server first retrieves metadata for the requested child site. It does this by calling the **proc_GetTpWebMetaDataAndListMetaData** (section 3.1.5.24) stored procedure using TDS.

2. The back-end database server returns the following four result sets:

- **List Webpart Result Set** (section 2.2.4.34). This contains information about the Web Parts related to the lists associated with a specified document.

- **List Metadata Result Set** (section 2.2.4.32). This contains the metadata associated with the list containing the requested document.
- **Site Metadata Result Set** (section 2.2.4.51). This contains metadata for the requested site.
- **Web Url Result Set** (section 2.2.4.65). This contains the store-relative form URL of the root of the requested child site.

3. The **proc_SecChangeToUniqueWeb** (section 3.1.5.36) stored procedure is called to update the **group** membership token or to add a new **member** to the specified site.

4.8 Remove Web Group

This example describes the requests made when a Web Group is to be removed using SharePoint Object Model code running on the front-end Web server.

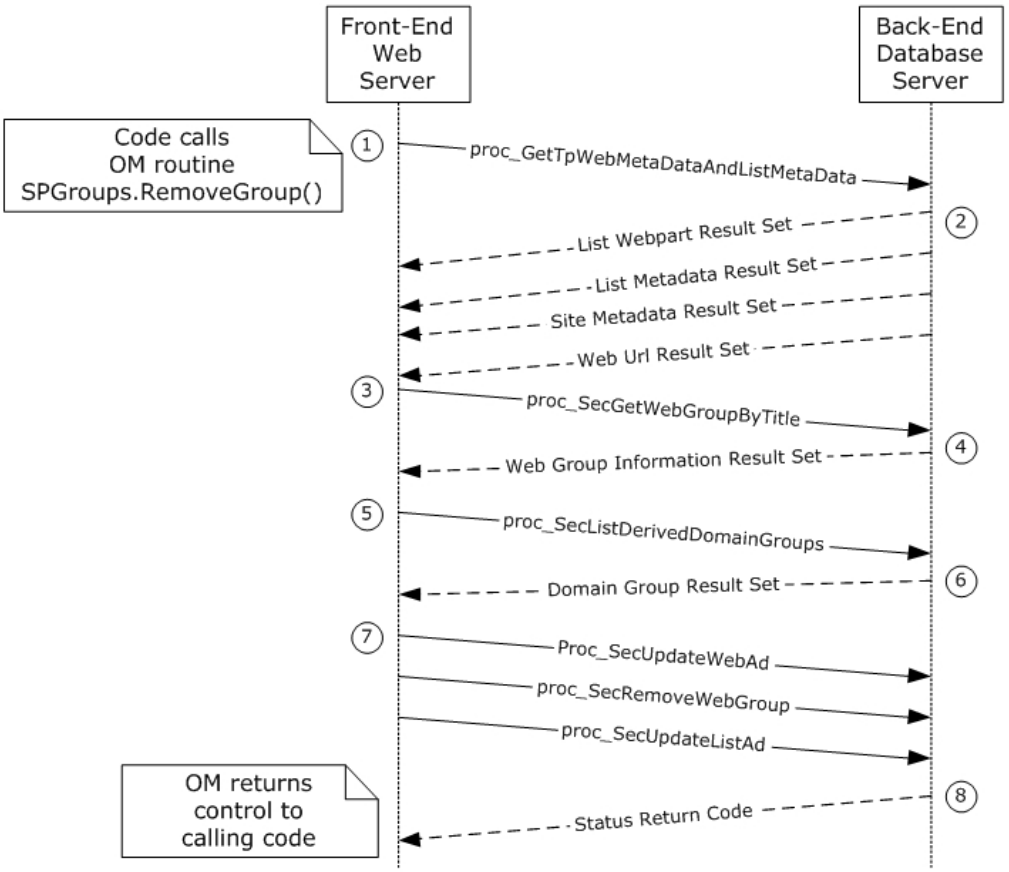


Figure 8: Request-response for removing a Web Group

This scenario is initiated by a call to the object model command SPGroups.RemoveGroup().

The following actions happen:

1. The front-end Web server first retrieves metadata for the requested child **site**. It does this by calling the **proc_GetTpWebMetaDataAndListMetaData** (section 3.1.5.24) stored procedure using TDS.

2. The back-end database server returns the following four result sets:

- **List Webpart Result Set** (section 2.2.4.34). This contains information about the Web Parts related to the lists associated with a specified document.
- **List Metadata Result Set** (section 2.2.4.32). This contains the metadata associated with the list containing the requested document.
- **Site Metadata Result Set** (section 2.2.4.51). This contains metadata for the requested site.
- **Web Url Result Set** (section 2.2.4.65). This contains the store-relative form URL of the root of the requested child site.

3. The front-end Web server calls the **proc_SecGetWebGroupByTitle** (section 3.1.5.58) called to get site group information for the sitegroup.
4. The back-end database server returns a **Web Group Information Result Set** (section 2.2.4.61).
5. The front-end Web server calls the **proc_SecListDerivedDomainGroups** (section 3.1.5.65) to get the list of domain groups for a specified **member**.
6. The back-end database server returns a Domain Group Result Set (section 2.2.4.18).
7. The front-end Web server calls the following procedures:

- **proc_SecUpdateWebAcl** (section 3.1.5.96) to update any **access control list (ACL)** for a group of permission-sharing web.
- **proc_SecRemoveWebGroup** (section 3.1.5.88) to remove a web group.

8. The back-end database server returns a Site Acl Result Set (section 2.2.4.45) contains information about any access control list (ACL) for the specified site returns.
9. The front-end Web server calls proc_SecUpdateListAcl (section 3.1.5.94) to update the ACL for a list.
10. The back-end database server returns a single output parameter indicating the status of the actions.

5 Security

5.1 Security Considerations for Implementers

In a trusted subsystem model, the process running on the front-end Web server uses its own **security principal** identity to access the content database on the back-end database server on behalf of the user, rather than using the account of the user accessing the front-end Web server as a database access account to access the content database. The database access account used by the **front-end Web server** has to have access to the appropriate content database on the back-end database server. If the account does not have the correct access rights, access will be denied when attempting to set up the [MS-TDS] connection to the content database, or when calling the stored procedures.

5.2 Index of Security Parameters

Security Parameter	Section
proc_SecAddPrincipalToWebGroup	3.1.5.31
proc_SecAddUser	3.1.5.32
proc_SecAddUserToSiteGroup	3.1.5.33
proc_SecChangeToInheritedList	3.1.5.34
proc_SecChangeToInheritedWeb	3.1.5.35
proc_SecChangeToUniqueWeb	3.1.5.36
proc_SecCheckDeletedAccounts	3.1.5.37
proc_SecCheckSiteGroupExistance	3.1.5.38
proc_SecCreateSiteGroup	3.1.5.39
proc_SecCreateWebGroup	3.1.5.40
proc_SecDecCurrentUsersCount	3.1.5.41
proc_SecGetAccountStatus	3.1.5.42
proc_SecGetCompleteWebGroupMemberList	3.1.5.43
proc_SecGetCurrentUsersCount	3.1.5.44
proc_GetGroupMembershipToken	3.1.5.45
proc_SecGetIndividualUrlSecurity	3.1.5.46
proc_SecGetPrincipalByEmail	3.1.5.47
proc_SecGetPrincipalById	3.1.5.48
proc_SecGetPrincipalByIdInWeb	3.1.5.49
proc_SecGetPrincipalByLogin	3.1.5.50
proc_SecGetPrincipalByLogin20	3.1.5.51
proc_SecGetPrincipalByLoginInWeb	3.1.5.52
proc_SecGetPrincipalDisplayInformation20	3.1.5.53

Security Parameter	Section
proc_SecGetSiteGroupById	3.1.5.54
proc_SecGetSiteGroupByTitle	3.1.5.55
proc_SecGetSiteGroupByTitle20	3.1.5.56
proc_SecGetWebGroupById	3.1.5.57
proc_SecGetWebGroupByTitle	3.1.5.58
proc_SecGetWebGroupByTitle20	3.1.5.59
proc_SecGetWebRequestAccess	3.1.5.60
proc_SecListAllSiteMembers	3.1.5.61
proc_SecListAllUsersWebGroups	3.1.5.62
proc_SecListAllWebMembers	3.1.5.63
proc_SecListAllWebMembersInWebGroups	3.1.5.64
proc_SecListDerivedDomainGroups	3.1.5.65
proc_SecListSiteGroupMembership	3.1.5.66
proc_SecListSiteGroups	3.1.5.67
proc_SecListSiteGroupsContainingUser	3.1.5.68
proc_SecListSiteGroupsInWebGroup	3.1.5.69
proc_SecListSiteGroupsInWebGroups	3.1.5.70
proc_SecListSiteGroupsWhichUserOwns	3.1.5.71
proc_SecListWebGroupMembership	3.1.5.72
proc_SecListWebGroups	3.1.5.73
proc_SecListWebGroupsByType	3.1.5.74
proc_SecListWebGroupsContainingSiteGroup	3.1.5.75
proc_SecListWebGroupsContainingUser	3.1.5.76
proc_SecMigrateUser	3.1.5.77
proc_SecRemovePrincipalFromWebGroup	3.1.5.78
proc_SecRemoveSiteGroup	3.1.5.79
proc_SecRemoveSiteGroupFromWeb	3.1.5.80
proc_SecRemoveUserFromSite	3.1.5.81
proc_SecRemoveUserFromSiteByLogin	3.1.5.82
proc_SecRemoveUserFromSiteGroup	3.1.5.83
proc_SecRemoveUserFromSiteGroupByLogin	3.1.5.84
proc_SecRemoveUserFromWeb	3.1.5.85

Security Parameter	Section
proc_SecRemoveUserFromWebByLogin	3.1.5.86
proc_SecRemoveUserFromWebGroupByLogin	3.1.5.87
proc_SecRemoveWebGroup	3.1.5.88
proc_SecResetToUniqueWeb	3.1.5.89
proc_SecSetGroupMembershipTokenAndEnsureWebMembership	3.1.5.90
proc_SecSetSiteGroupProperties	3.1.5.91
proc_SecSetWebGroupProperties	3.1.5.92
proc_SecSetWebRequestAccess	3.1.5.93
proc_SecUpdateListAcl	3.1.5.94
proc_SecUpdateUser	3.1.5.95
proc_SecUpdateWebAcl	3.1.5.96

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 updates to those products.

- Windows Server 2003 operating system
- Windows SharePoint Services 2.0

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

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

<1> Section 2.2.6.3.3.3: Windows SharePoint Services 2.0 emits this data if it is present in the underlying template definition, even if the field type is not of the specified type.

<2> Section 2.2.6.3.3.3: Windows SharePoint Services 2.0 sends this data if it is present in the underlying template definition, even if the field type is not of the specified type.

<3> Section 2.2.6.3.3.3: Windows SharePoint Services 2.0 sends this data if it is present in the underlying template definition, even if the field type is not of the specified type.

<4> Section 2.2.6.3.3.3: Windows SharePoint Services 2.0 sends this data if it is present in the underlying template definition, even if the field type is not of the specified type.

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

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

Z