[MS-WSSO]:
Windows SharePoint Services Overview

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

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

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

This document provides an overview of the Windows SharePoint Services Overview Protocol Family. It is intended for use in conjunction with the Microsoft Protocol Technical Documents, publicly

available standard specifications, network programming art, and Microsoft Windows distributed systems concepts. It assumes that the reader is either familiar with the aforementioned material or has immediate access to it.

A Protocol Family System Document does not require the use of Microsoft programming tools or programming environments in order to implement the Protocols in the System. Developers who have access to Microsoft programming tools and environments are free to take advantage of them.

Abstract

This document describes the intended functionality of the Windows SharePoint Services File, Print, and User/Group Administration system and how the protocols in this system interact. It provides examples of some of the common user scenarios. It does not restate the processing rules and other details that are specific for each protocol. These details are described in the protocol specifications for each of the protocols and data structures that make up this system.

Revision Summary

Date	Revision History	Revision Class	Comments
03/14/2008	0.1	Major	Initial Availability.
06/20/2008	0.1.1	Editorial	Revised and edited the technical content.
07/25/2008	0.1.2	Editorial	Revised and edited the technical content.
08/29/2008	1.0	Major	Updated and revised the technical content.
10/24/2008	1.0.1	Editorial	Revised and edited the technical content.
12/05/2008	1.0.2	Editorial	Initial availability
01/16/2009	1.0.3	Editorial	Revised and edited the technical content.
02/27/2009	1.0.4	Editorial	Revised and edited the technical content.
04/10/2009	2.0	Major	Updated and revised the technical content.
05/22/2009	2.0.1	Editorial	Revised and edited the technical content.
07/02/2009	3.0	Major	Updated and revised the technical content.
08/14/2009	3.0.1	Editorial	Revised and edited the technical content.
09/25/2009	3.0.2	Editorial	Revised and edited the technical content.
11/06/2009	3.0.3	Editorial	Revised and edited the technical content.
12/18/2009	4.0	Major	Updated and revised the technical content.
01/29/2010	4.0.1	Editorial	Revised and edited the technical content.
03/12/2010	5.0	Major	Updated and revised the technical content.

Date	Revision History	Revision Class	Comments
04/23/2010	6.0	Major	Updated and revised the technical content.
06/04/2010	6.0.1	Editorial	Revised and edited the technical content.
07/16/2010	6.0.1	No change	No changes to the meaning, language, or formatting of the technical content.
08/27/2010	6.0.1	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	7.0	Major	Significantly changed the technical content.
11/19/2010	7.1	Minor	Clarified the meaning of the technical content.
01/07/2011	7.1	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	7.1	No change	No changes to the meaning, language, or formatting of the technical content.
03/25/2011	7.1	No change	No changes to the meaning, language, or formatting of the technical content.
05/06/2011	7.1	No change	No changes to the meaning, language, or formatting of the technical content.
06/17/2011	7.2	Minor	Clarified the meaning of the technical content.

Contents

1 Introduction

This document provides an informative overview of the back-end protocols implemented by Windows® SharePoint® Services (WSS) File, Print and User/Group administration capabilities. WSS is a Web-based technology that provides:

- An out-of-the-box, team-oriented Web site for collaboration.
- A development platform for building Web-based experiences that take advantage of the collaboration features of WSS.
- A framework for deploying and managing the WSS team site and applications built on the WSS platform.

As part of the collaboration services, WSS provides support for document collaboration, including the ability to store, update, and view documents. This capability is delivered through document libraries within team sites. Much of the WSS infrastructure is designed to ensure that WSS sites provide these services in a highly scalable, manageable, and extensible way, as described in detail later in this document.

The purpose of this document is to provide an understanding of the concepts and architecture underlying the file management and security related features of WSS. In order to deliver these file services capabilities, WSS uses three major sets of protocols:

1. File-oriented communication between the **End-User Client (EUC)** and the WSS **Web Front-End (WFE)** using the **Web Distributed Authoring and Versioning (WebDAV)** Protocol as described in [RFC2518], [MS-WDV], and [MS-WDVSE]. The EUC can also use the FrontPage Server Extensions Remote Protocol as described in [MS-FPSE]. The use of WebDAV is recommended over the FrontPage Server Extensions Remote Protocol.
2. Web pages presented to the client using standard HTTP.
3. Communication between the WFE and the WSS **Back-End Database Server (BEDS)** using specific queries and stored procedures implemented using **Tabular Data Stream (TDS)**, a protocol for SQL communication described in [MS-TDS]. Details of the File, Print, and User/Group administration communication between the WFE and BED service is described in [MS-WSSFO] for Windows® SharePoint® Services 3.0 and [MS-WSSFO2] for Microsoft® SharePoint® Foundation 2010.

This document provides an overview for protocols for both Windows SharePoint Services 3.0 and SharePoint Foundation 2010. It generally refers to WSS when the subject applies to both versions, and explicitly calls out the version when necessary for clarity.

Note The T-SQL based protocols have changed significantly in their function between Windows SharePoint Services 3.0 and SharePoint Foundation 2010. New versions of the specification documents for these protocols that target the SharePoint Foundation 2010 release use the same titles as their predecessors with the addition of "Version 2" in the protocol title. However, although the document titles simply show the addition of a version number, the protocols described in the documents are considered to be completely different, and cross-compatibility between the protocol versions is not supported. SharePoint Foundation 2010 does not support any of the T-SQL protocols that are not identified with "Version 2" in the title.

This overview document covers both the original protocol version and the new "Version 2" protocol documents, and where appropriate identifies the appropriate protocol name and/or document short name.

1.1 Glossary

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

access control entry (ACE)
access control list (ACL)
Active Directory
forest
Lightweight Directory Access Protocol (LDAP)
security identifier (SID)
security principal

The following terms are specific to this document:

Back-End Database Server (BEDS): A computer that runs SQL Server and responds to requests from the **WFE** server.

binary large object (BLOB): The SQL Server concept of unstructured, binary data streams commonly associated with files.

End-User Client (EUC): A computer on which an individual user is requesting specific file operations.

round-robin load balancer: A resource management procedure where each process is assigned an equal portion of computer resources in a circular order.

scale-out: A method of adding computing resources by adding additional computers to the system, rather than increasing the computing resources on the computers in the system.

SharePoint Farm: WSS deployments scaled-out with multiple computers.

Tabular Data Stream (TDS): A protocol for SQL communication specified in [MS-TDS].

Web Front-End (WFE): A computer that receives requests from an EUC and provides Windows SharePoint Services capabilities by manipulating information stored in a database.

The following protocol abbreviations are used in this document:

FPSE: [MS-FPSE]: FrontPage Server Extensions Remote Protocol Specification.

TDS: [MS-TDS]: MS-Tabular Data Stream Protocol Specification.

WebDAV: [MS-WDV]: Web Distributed Authoring and Versioning (WebDAV) Protocol: Client Extensions Specification, [MS-WDVSE]: Web Distributed Authoring and Versioning (WebDAV) Protocol: Server Extensions Specification.

1.2 References

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

1.2.1 Normative References

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

[MS-ADTS] Microsoft Corporation, "Active Directory Technical Specification".

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

[MS-SYS] Microsoft Corporation, "Windows System Overview".

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

[MS-WDV] Microsoft Corporation, "Web Distributed Authoring and Versioning (WebDAV) Protocol: Client Extensions".

[MS-WDVSE] Microsoft Corporation, "Web Distributed Authoring and Versioning (WebDAV) Protocol: Server Extensions".

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

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

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

[RFC2396] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998, http://www.ietf.org/rfc/rfc2396.txt

[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

1.2.2 Informative References

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

[MSDN-SHPTSDK] Microsoft Corporation, "Windows SharePoint Services 3.0 SDK", http://msdn.microsoft.com/en-us/library/ms441339.aspx

[MSDN-SHPTSDK4] Microsoft Corporation, "SharePoint Products and Technologies SDK: 2010 API Reference (Technical Preview)", July 2009, http://msdn.microsoft.com/en-us/library/cc339475.aspx

[MSDN-SQLRBS] Microsoft Corporation, "Remote BLOB Store Provider Library Implementation Specification", Microsoft SQL Server 2008, http://msdn.microsoft.com/en-us/library/cc905212.aspx

[MSDN-WSSEBS] Microsoft Corporation, "External Storing of Binary Large Objects (BLOBs) in Windows SharePoint Services", http://msdn.microsoft.com/en-us/library/bb802976.aspx

2 Functional Architecture

Windows® SharePoint® Services (WSS) provides team-oriented collaboration Web sites, a platform for building Web-based applications that use the WSS collaboration features and a framework for deploying and managing these sites and applications.

This section describes the architecture for delivering and supporting the framework in terms of computers, databases, external services, and protocols, and the architecture for supporting the collaboration features in terms of storage concepts within the framework.

A detailed discussion of security concepts is provided in section 2.9.

2.1 Overview

WSS is designed to support a broad range of deployments. At a high level, four different types of systems are involved:

1. The End-User Client (EUC) is a computer on which an individual user is requesting specific file and user operations. These requests are communicated using HTTP-based protocols, including HTTP, DAV and Microsoft FrontPage Server Extensions.

2. The Web Front-End (WFE) is a computer that receives requests from an EUC and provides WSS capabilities by manipulating information stored in a database. These database requests are expressed in the T-SQL language and communicated using the TDS protocol.

3. The Back-End Database Server (BEDS) is a computer that runs Microsoft® SQL Server® and responds to requests from the WFE server.

4. An **Active Directory** server responds to authentication requests from the EUC, WFE, and BEDS. These components could use an alternate authentication mechanism besides Active Directory.

These systems are shown in the following diagram.

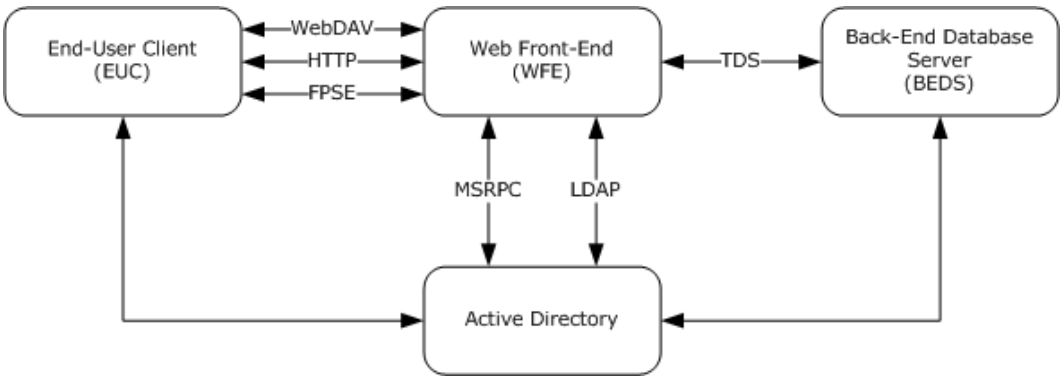

Figure 1: Intersystem protocol relationships

WFE and BEDS are considered part of the WSS deployment; the EUC is normally a user's desktop or laptop computer that connects to WSS services, and Active Directory is part of the generally available infrastructure. Both WFE and BEDS can run on a single computer, so the simplest WSS deployment could be just a single server. Alternatively, multiple instances of WFE and BEDS computers can be installed for greater throughput and redundancy.

The following sections describe the architectural concepts pertaining to how WSS uses the protocols specified by the Member Protocol specifications identified in section 2.2. These sections describe:

- How WSS deployments can be scaled out with multiple computers, called a **SharePoint Farm**. This **scale-out** is transparent to individual WFE computers as they respond to individual requests from an EUC.
- The WSS storage model, which allows for a variety of data management and organization techniques:
  - Storage for non-file system objects, such as sites, site collections, lists, and so on.
  - Storage for file system objects, such as files and folders.
  - Advanced storage concepts, such as attachments, thickets, ghosting, and so on.
- The SQL databases required for the operation of a WSS deployment.

2.1.1 Scale-out Technologies

When a WSS deployment is scaled-out across multiple servers in a farm deployment, it uses two main technologies to increase throughput and availability.

1. **Network Load Balancing of Web Front-Ends:** WSS supports network load balancing technologies that distribute client requests across multiple servers in a farm. These individual WFEs are stateless; any WFE in the farm is prepared to handle any client request in the same way as any other WFE in the farm. By eliminating state or session information about the WFE, overall operational throughput can be increased simply by adding more WFE servers. This stateless operation also makes the end-user experience more robust, because if a WFE fails, another WFE server in the farm can handle future requests from the user. WFEs do not communicate with one another in responding to client requests, but independently handle requests directed to them by the load-balancing technology.
2. **Vertical Data Partitioning across SQL Databases:** As the deployment grows and the capacity of an individual server running SQL Server is fully consumed, additional back-end SQL resources can be deployed by adding additional servers that host completely separate content databases. Different **Site Collections** can be deployed into those separate content databases, and when a client request comes to a particular WFE, that WFE will fetch the site content strictly from the appropriate back-end database. This provides the ability to load-balance across multiple back-end resources, but does require manual placement of high-load sites into separate content databases.

The network load-balancing technology creates the need for each WFE in the server farm to behave identically to all other WFEs. The load balancer can be Windows Network Load Balancer, any one of a number of third-party hardware products, or even a simple custom **round-robin load balancer**.

2.1.2 Storage Architecture

WSS provides a flexible model for storage which allows for a robust variety of data management and organization techniques. The following diagram presents a high-level view of the containers in this hierarchy. These containers provide important organizational and management tools for content stored in WSS, and also form the core securable objects.

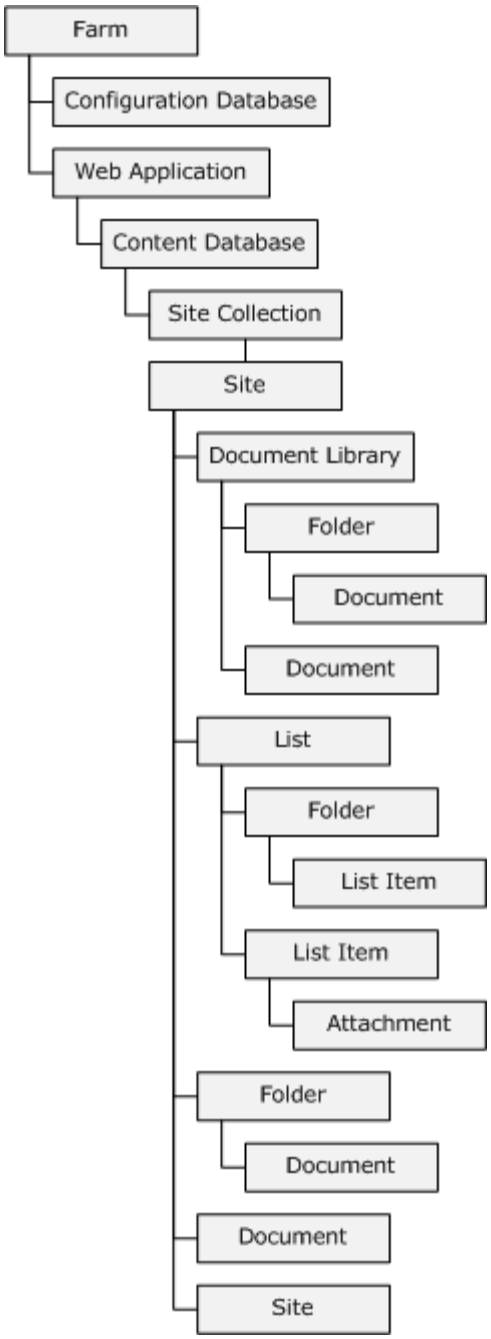

Figure 2: SharePoint storage object hierarchy

Each level of this hierarchy provides a specific set of management and deployment capabilities, and each item shown in the diagram is explained in detail in a later section.

2.1.2.1 Non-File System Objects

Multiple objects within WSS are not directly tied to the representation of a file system. These objects are described in detail in later sections.

2.1.2.1.1 Farm

The SharePoint Farm is the root container for an individual SharePoint deployment. This corresponds to a single Configuration Database (section 2.1.2.6) that describes the topology of the farm and global settings. Every object and container within a farm has a unique HTTP URL, which can be used to directly reference that object.

2.1.2.1.2 Web Application

The Web Application is the container for grouping Site Collections within the context of the Web server. A Web Application is associated with one or more Content Databases that hold the content for Site Collections.

2.1.2.1.3 Site Collection

A Site Collection is a Web site within WSS that can be created and managed directly by end-users or, alternatively, more directly controlled by IT administrators. Characteristics of Site Collections include:

- They are the basic unit of scale for WSS.
- They can be transparently organized.
- Multiple Site Collections can be distributed across multiple Content Databases.
- An individual Site Collection can only reside in one Content Database.

Site Collections also provide a boundary, defining Site groups that can be given access to specific resources inside WSS.

2.1.2.1.4 Site

A Site is a container within a Site Collection that allows for delegated administration where certain actions, such as managing permissions to content within the Site, can be delegated by the Site Collection administrator to the Site administrator, who will have more direct knowledge of the business needs and allow for more efficient administration. A Site Collection will only have one Site at the root, but can have many nested Sites under the root Site.

2.1.2.1.5 List

A List is a location within a Site that maintains a configurable data structure where end-user data can be stored. A List is composed of multiple columns that define the structure for the data stored within the List.

2.1.2.1.6 List Item

A List Item is a unique row within a List where individual end-user data elements are stored. A List item follows the structure of columns as defined by the List that contains the List Item.

2.1.2.2 File System Objects

An important part of WSS is the file system abstraction that it presents over data. WSS uses basic file system concepts, such as files and folders. The following sections describe the WSS objects that are exposed as file system concepts.

2.1.2.2.1 Document Library

A Document Library is a type of List specifically designed to be a location within a Site where end-user Documents are stored.

2.1.2.2.2 Folder

A Folder is an organizational tool used within a Site or Document Library that enables easier browsing and navigation. Within a Document Library, Folders appear similar to their equivalent type within a file system.

2.1.2.2.3 Document

A Document is an individual file that a user might create, read, or update using an authoring application. Documents can be stored within a Site, Document Library, Folder, or as an Attachment.

2.1.2.3 Advanced Storage Concepts

In addition to basic file system concepts, WSS provides a variety of specialized file system objects and operations as described in the following sections.

2.1.2.3.1 Attachment

Many List structures within WSS can be configured to allow Attachments, which allows multiple files to be included with each list item.

2.1.2.3.2 Thickets

For some complex HTML documents, WSS provides the capability to store the document separated into its component sibling documents. This group of documents is treated like a single document for most file operations, but is stored as a set of related documents within the file store location.

2.1.2.3.3 Ghosting

Sites and Lists tend to share a relatively small set of common system files across many instances within the BEDS. To avoid having to store these files repeatedly for every Site or List in the BEDS, WSS allows files to be ghosted, meaning that the content of the file is not actually stored in the BEDS. Instead, a reference to a location on the WFE where the source of the file can be found is stored in the file metadata.

2.1.2.3.4 Versioning

WSS can be configured on a per-List or per-Document Library basis to store multiple versions of Documents. If versioning is configured for a storage location, each new version of a Document is stored with an incrementing version number that can be either in the form 'Major Version Number' (#) or 'Major Version Number.Minor Version Number' (#.#).

If Major Version Number.Minor Version Number version numbering is used, individual Documents start their numbering at 0.1, and the version can be promoted to a Major Version using the

Publishing feature. Prior versions of a document can be retrieved by users with the appropriate access rights.

2.1.2.3.5 Publishing

WSS can be configured on a per-List or per-Document Library basis to allow publishing features with Documents. If publishing features are configured for a storage location, each version of a Document can be configured as a draft, and therefore, not be available to be viewed by users without the appropriate access rights for viewing unpublished versions.

2.1.2.3.6 Document Property Promotion

The columns within a Document Library can be populated with data directly extracted from the Document. A Document can contain properties (usually metadata about the document such as author or title) which can be associated with columns on a one property to one column basis. When a document is uploaded, the metadata is extracted from the document, and these associated columns are populated with the data. Conversely, WSS has the capability to demote the associated column data back into the document properties if changes have occurred within the Document Library.

2.1.2.3.7 Large File Access

When dealing with very large files, obtaining the complete file contents in a single buffer as part of one operation can be a burden on system resources. Instead, the BEDS provides functionality to return files larger than a specified size to the WFE in a series of smaller chunks that can be processed more smoothly.

2.1.2.3.8 BLOB Storage Outside the Content Database

WSS provides the ability to allow file metadata to be stored by WSS in a Content Database, while allowing the actual file contents to be stored in an external file system. The term **Binary Large Object (BLOB)** refers to the SQL Server concept of unstructured, binary data streams that are commonly associated with files.

WSS does not natively provide a BLOB store. There are two APIs that allow a BLOB storage provider to be built and registered with WSS:

1. External BLOB Storage: For more information about External BLOB Storage, see [MSDN-WSSEBS]. Windows® SharePoint® Services 3.0 and Microsoft® SharePoint® Foundation 2010 support External BLOB Storage providers.

2. Remote BLOB Storage: For more information about Remote BLOB Storage, see [MSDN-SQLRBS]. SharePoint Foundation 2010 supports Remote BLOB Storage providers.

   Note In SharePoint Foundation 2010, the use of Remote BLOB Storage is recommended over External BLOB Storage.

2.1.2.4 SQL Databases

The Content Database and Configuration Database are two core types of databases that are required for the operation of a WSS deployment. These databases are considered "internal" to WSS, which does not support users, developers, or system administrators directly accessing or manipulating content in these databases. This is true regardless of whether the WSS deployment is running on a single server or across multiple computers.

Instead, WSS exposes a full set of APIs to manage access to this data. For more information regarding use of these APIs in Windows® SharePoint® Services 3.0, see [MSDN-SHPTSDK]. For more information regarding use of these APIs in Microsoft® SharePoint® Foundation 2010, see [MSDN-SHPTSDK4].

2.1.2.5 Content Databases

A **Content Database** stores and manages end-user content. Each Web Application will have one or more Content Databases. The WSS Content Database stores the data and documents that end-users have associated with their SharePoint Site. The contents of this database are fully normalized in order to efficiently perform the kinds of operations required for the high-scale, highly-configurable system previously described. The role of a WSS WFE is to fetch the appropriate data from these multiple locations within SQL using the appropriate set of queries and stored procedures and to correctly interpret and map the results into a correct response to the EUC. The queries and procedures are specified in [MS-WSSFO] for Windows® SharePoint® Services 3.0, and [MS-WSSFO2] for Microsoft® SharePoint® Foundation 2010.

2.1.2.6 Configuration Database

The **Configuration Database** describes the topology of the farm and global settings. Each farm will have only one Configuration Database. The Configuration Database is essentially the definition of the WSS deployment, for either a single instance of WSS or for a farm. In the farm context, the Configuration Database contains information representing the global settings that are required to provide consistent operation across all servers within the farm, and to map requests to particular Content Databases. The Configuration Database allows only restricted access, as described in section 2.9.2.2. In a typical setup, the Configuration Database contents can only be modified from the central administration Web Application, while run-time Web Applications have only read access to these configuration settings.

2.1.2.6.1 Site Map

In addition to a description of the global topology, the Configuration Database also stores a Site Map, which is a mapping of all Site Collections to the individual Content Databases that contain the end-user content for the Site Collections. The following diagram shows how those URLs can be mapped to individual Content Databases.

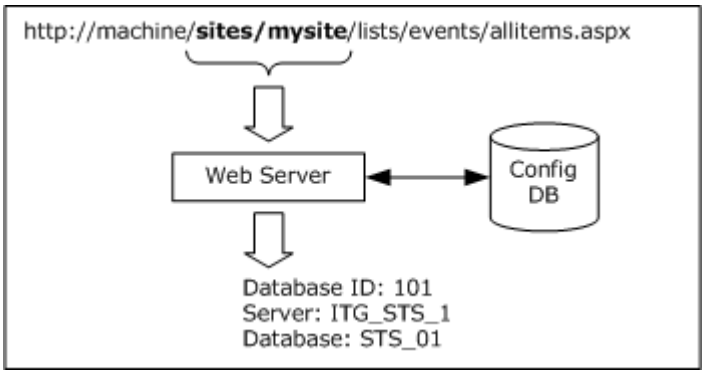

Figure 3: Determining Site Collection URL

This mapping can use a section of the URL (in this case "sites/mysite") to map to an individual BEDS and Content Database. The URL is stored in a Server-Relative format, so this mapping is effective across multiple, different names for the server. The server addresses http://machine, http://localhost, or http://157.55.234.184 are all equivalent.

The Site Collection Lookup (section 2.1.2.6.1.1) describes the process of looking up a Site Collection from the Site Map, and determining the Connection String to the Content Database that holds the Site Collection's end-user generated content.

2.1.2.6.1.1 Site Collection Lookup

When a request is made for content at a specific URL, WSS has to determine to which Site Collection the URL is referring, as well as the Connection String of the Content Database holding the content for that Site Collection. This is accomplished in the following steps:

1. **Web Application Lookup:** Site Collection lookup begins by examining the portion of the incoming URL beginning with the Scheme Component and ending with the Authority Component (for example, http://example.com:80). Scheme and Authority are defined in [RFC2396] sections 3.1 and 3.2. This URL is compared against a stored set of Web Application URLs. If one of the URLs in the list matches the incoming URL, the associated Web Application is used for the remainder of the operation.

2. **Prefix Matching:** Web Applications contain a set of Site Collection prefixes. These prefixes are URL Path Components that are used to determine which portion of the incoming URL Path Component is the Server-Relative URL of the Site Collection. This is done by matching all of the prefixes against the start of the Path Component of the incoming URL. If more than one prefix matches the beginning of the incoming URL Path Component, the longest matching prefix is used. A Web Application can contain any combination of the two types of prefix:

   - **Explicit Prefixes:** An explicit prefix indicates that the portion of the Path Component up to and including the prefix is included in the Site Collection Server-Relative URL. For example, if a user requests http://example.com/sitename/web/list/document.htm, and if the Web Application corresponding to http://example.com contains an explicit prefix named "sitename", then "/sitename" is the Server-Relative URL of the Site Collection.

Incoming URL	Web Application explicit prefixes	Resulting Site Collection Server-Relative URL
http://example.com/a/b/c.htm	"a"	"/a"
http://example.com/a/b/c.htm	"a", "a/b"	"/a/b"
http://example.com/a/b.htm	"a", "a/b"	"/a"
http://example.com/a/b.htm	"c"	<No Match>
http://example.com/a/b.htm	""	"/"

   - **Wildcard Prefixes:** A wildcard prefix indicates that the portion of the Path Component up to and including the first Path Component segment following the prefix is included in the Site Collection name. For example, if a user makes a request for http://example.com/sites/sitename/web/list/document.htm, and if the Web Application corresponding to http://example.com contains a wildcard prefix named "sites", then "/sites/sitename" is the Server-Relative URL of the Site Collection.

Incoming URL	Web Application wildcard prefixes	Resulting Site Collection Server-Relative URL
http://example.com/a/b/c/d.htm	"a", "a/b"	"/a/b/c"
http://example.com/a/b.htm	"a", "a/b"	<No Match>

Incoming URL	Web Application wildcard prefixes	Resulting Site Collection Server-Relative URL
http://example.com/a/b.htm	""	"/a"

3. **Site Collection Identifier Lookup:** Once the Site Collection URL is determined, it is passed to the Configuration Database, along with the Web Application identifier. A Site Collection identifier is returned along with the identifier of the Content Database in which the Site Collection content is stored. If the specified combination Site Collection URL and Web Application identifier cannot be found in the Configuration Database, the Site Collection does not exist.

   Some Site Collections are identified not by the Path Component of the URL, but by the URL Authority Component (For example, "example.com:80"). These are known as "Host Header Site Collections". If the Web Application cannot be identified from the Scheme and Authority Components of the incoming URL, Site Collection lookup assumes that the incoming URL refers to a Host Header Site Collection.

   In this case, the Authority Component of the incoming URL is passed to the Configuration Database which returns the corresponding Site Collection identifier. The Site Collection identifier is then passed back to the Configuration Database which returns the identifier of the Content Database in which the Site Collection content is stored. If the specified Authority Component cannot be found in the Configuration Database, the Site Collection does not exist.

4. **Content Database Connection String Lookup:** Once the Content Database identifier is known, a lookup occurs to determine connection string information about the Content Database. The following steps occur to generate this connection string:

   1. The Content Database identifier is passed to the Configuration Database, which returns the Content Database name and the identifier of the Database Service that is hosting the Content Database. If SQL Authentication is intended to be used when connecting to the Content Database, the connection user name and password are also returned at this time.

   2. The identifier of the Database Service is then passed to the Configuration Database, which returns the name of the Database Service and the identifier of the Server on which the Database Service is running.

   3. The identifier of the Server is passed to the Configuration Database which returns the address of the Server.

   4. Finally, the Server address, Database Service name, Content Database name, and optionally, the Content Database user name and password are combined to build the Content Database Connection String.

2.2 Protocol Summary

The following table provides a comprehensive list of the Member Protocols of the Windows® SharePoint® Services File, Print, and User/Group Administration system.

Protocol name	Description	Short name
Windows SharePoint Services (WSS): File Operations Database Communications	This protocol specifies the communication between the WFE and the BEDS server used to satisfy requests involving file access and administration of users and groups within Windows® SharePoint® Services 3.0.	[MS-WSSFO]

Protocol name	Description	Short name
Protocol		
Windows SharePoint Services (WSS): File Operations Database Communications Version 2 Protocol	This protocol specifies the communication between the WFE and the BEDS server used to satisfy requests involving file access and administration of users and groups within Microsoft® SharePoint® Foundation 2010.	[MS-WSSFO2]
Web Distributed Authoring and Versioning (WebDAV) Protocol: Client Extensions	The client extensions in this protocol extend the WebDAV Protocol, as specified in [RFC2518], by introducing new headers that both enable the file types that are not currently manageable and optimize protocol interactions for file system clients. These WebDAV Protocol: Client Extensions do not introduce new functionality into the WebDAV Protocol, but instead optimize processing and eliminate the need for special-case processing.	[MS-WDV]
Web Distributed Authoring and Versioning (WebDAV) Protocol: Server Extensions	The server extensions in this protocol extend WebDAV by introducing new HTTP request and response headers that both enable the file types that are not currently manageable and optimize protocol interactions for file system clients. These extensions also introduce a new WebDAV method that is used to send search queries to disparate search providers.	[MS-WDVSE]
FrontPage Server Extensions Remote Protocol	This protocol specifies a set of server extensions that can be used to augment a basic HTTP server. These extensions provide file server functionality similar to WebDAV, allowing a Web site to be presented as a shared folder. The use of WebDAV is recommended over the FrontPage Server Extensions Remote Protocol. The SharePoint Team Services dialogview is an application of the FrontPage Server Extensions Remote Protocol that is addressed in the FrontPage Server Extensions Remote Protocol Specification [MS-FPSE] because it has certain behaviors apart from the normal FrontPage Server Extensions Remote Protocol communications. The purpose of the dialogview is to allow a client to display a server-rendered HTML-based rendering of the files located on a particular Web site.	[MS-FPSE]

2.3 Environment

The following sections identify the context in which the system exists. This includes the systems that use the interfaces provided by this system of protocols, other systems that depend on this system, and, as appropriate, how components of the system communicate.

2.3.1 Dependencies on this System

None of the systems that are used to deliver file, print, user administration, or group administration services depend on this system.

2.3.2 Dependencies on Other Systems/Components

The Windows® SharePoint® Services File, Print, and User/Group Administration system depends on the following systems:

- Microsoft Windows® System: [MS-SYS]
- Tabular Data Stream Protocol: [MS-TDS]
- Active Directory: [MS-ADTS]

Windows® SharePoint® Services 3.0 depends on the following components to function:

- Windows Server® 2003 operating system with Service Pack 1 (SP1), Windows Server® 2003 operating system with Service Pack 2 (SP2), Windows Server® 2003 operating system with Service Pack 3 (SP3), and Windows Server® 2003 R2 operating system
  - Internet Information Services (IIS) 6.0
- Microsoft® .NET Framework 3.0 or Microsoft® .NET Framework 3.5
  - Microsoft® ASP.NET 2.0

Microsoft® SharePoint® Foundation 2010 depends on the following systems/components to function:

- Windows Server® 2008 operating system with Service Pack 2 (SP2) and Windows Server® 2008 R2 operating system
  - Internet Information Services (IIS) 7.0
- .NET Framework 3.5
  - ASP.NET 2.0
- Microsoft® Forefront™ Unified Access Gateway
- Microsoft® SQL Server® 2008 Express Edition with Service Pack 1

2.3.2.1 Domain Controller/Directory Service

In addition, WSS can communicate with an Active Directory domain controller to provide authentication services that enable the User/Group administration functions described in this document. This domain controller provides an **LDAP**-enabled directory service that stores user information, such as name and e-mail addresses.

2.4 Assumptions and Preconditions

This section briefly documents the assumptions and preconditions required by the system. The scope of this discussion is intended to be implementation-independent and is limited to the system level of WSS.

- The WSS server(s) is reachable by external clients via an established IP address (or IP addresses).
- The WSS server(s) functional components are started collectively and the WSS server(s) accepts client requests.
- The WSS WFE servers can reach BEDS and have appropriate permissions to access data in the Content and Configuration Databases.
- The WSS WFE and BEDS are matching versions, or within an acceptable range of versions. For more information about versioning, see section 2.6.

- In the case where Active Directory is used to provide authentication, the Directory Service is accessible to the WSS server. Any intermediate firewalls, routers, or connection points between components of the system have all required ports and gateways open for communication between them.

2.5 Use Cases

The following use case is provided only for an understanding of the Windows SharePoint Services File, Print, and User/Group Administration system. It is not intended to be a thorough and complete modeling of the system for implementation purposes.

2.5.1 Creating a SharePoint Document Library File from the Client Console

This use case describes the simplest way to create a file using the protocols covered in this system. The actor in this case is the user who is creating the text file hello.txt in a WSS Document library. The text file contains the text "hello". For details regarding a scenario of this type, see the example Create File from Client in section 3.4.

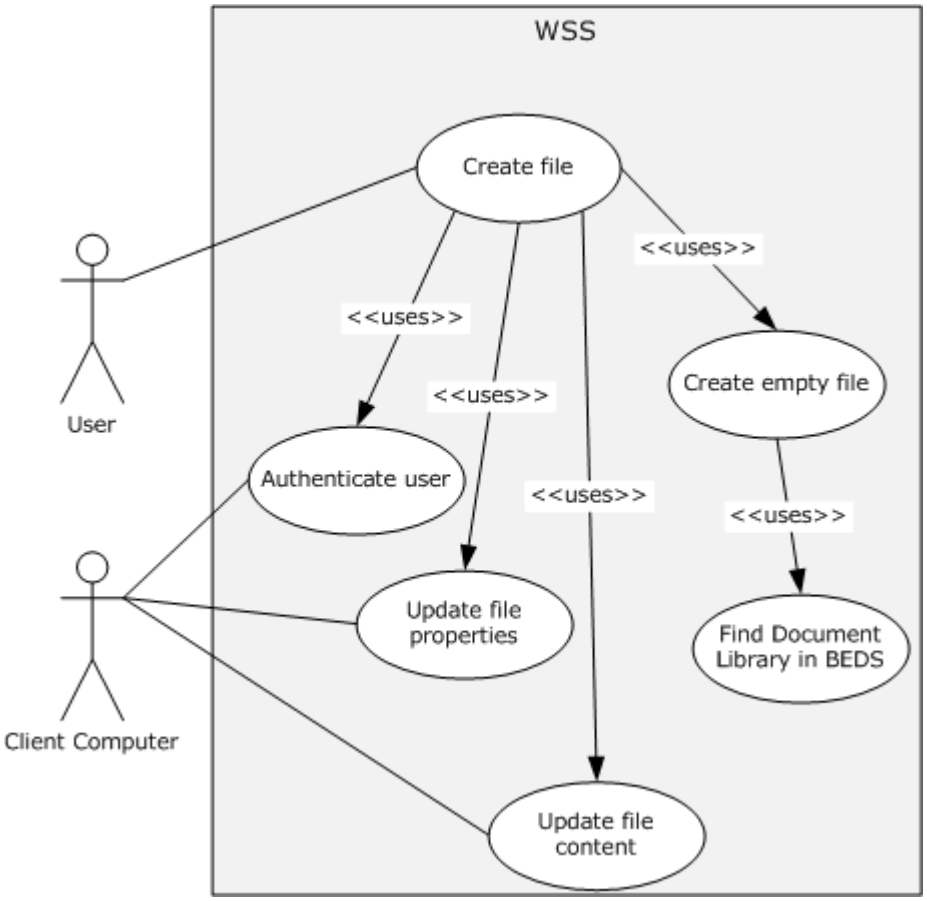

Figure 4: Create file from client computer

Preconditions

- The user has read/write access permissions to an existing SharePoint Document Library called http://server/site/doclib.

- The user is logged on to a client computer running Windows® 7 operating system<1> with an authenticated Windows session, and can access the WSS site containing the Document Library.
- From a command prompt window, the user types the following command:

```
echo hello >\\server\site\doclib\hello.txt
```

Main Flow

1. The user presses Enter after typing the "echo" command.
2. The WSS WFE authenticates the user.
3. WSS finds the location of the document library and verifies that the user has the appropriate access permissions.
4. WSS creates an empty file in the document library and confirms successful creation of the file back to the client.
5. The client updates the file properties and the file contents.

Error Scenarios

1. The user does not have the appropriate access permissions: The client notifies the user that access is denied.
2. The client cannot connect to the WSS WFE: The client notifies the user of the connection error.
3. The client cannot update the file properties or file contents: The client notifies the user of the file write properties or file contents write error.

2.6 Versioning, Capability Negotiation, and Extensibility

The WSS WFE and WSS BEDS perform explicit version verifications.

The client calls proc_GetVersion to retrieve version information from the server and to decide whether it should connect to the database. proc_GetVersion is described in [MS-WSSFO] section 3.1.5.39 for Windows® SharePoint® Services 3.0, and in [MS-WSSFO2] section 3.1.5.44 for Microsoft® SharePoint® Foundation 2010.

The version information is stored in the Versions table, which is described in [MS-WSSFO] section 2.2.7.11 for Windows SharePoint Services 3.0, and [MS-WSSFO2] section 2.2.8.11 for SharePoint Foundation 2010.

WSS WFE initializes the connection to the BEDS according to the following steps:

1. When the WFE was added to the SharePoint Farm, the administrator gave a connection string to the Configuration Database.
2. In SharePoint Foundation 2010, the WFE verifies that the Configuration Database is the correct version by calling proc_GetVersion with each of the version identifiers specified in table B of [MS-WSSFO2] section 3.1.5.44. The WFE also ensures that the version numbers are within the acceptable range defined in that same table. If one or more versions are not within the acceptable range, the WFE disconnects from the Configuration Database.

- In Windows SharePoint Services 3.0, the version of the Configuration Database is not checked; the connection initialization operation proceeds directly to the next step.

3. Once the WFE verifies that it is talking to a Configuration Database with an appropriate version, the WFE gets the connection information for the Content Database that is required to respond to the URL request being handled. For more details on the URL Site Map Lookup, see sections 2.1.2.6 and 2.1.2.6.1.

4. When the WFE has the connection string to the Content Database, the WFE connects to the Content Database and verifies that it is the correct version by calling proc_GetVersion with each of the version identifiers specified in table A of [MS-WSSFO] section 3.1.5.39 for Windows SharePoint Services 3.0, or in [MS-WSSFO2] section 3.1.5.44 for SharePoint Foundation 2010, and ensures that the version numbers are within the acceptable range defined in the same table. If one or more versions are not within the acceptable range, the WFE disconnects from the content database.

   - Windows SharePoint Services 3.0 uses only one of the version identifiers. For more information, see the footnote for proc_GetVersion in [MS-WSSFO2] section 3.1.5.44 regarding SharePoint Foundation 2010.

5. The validation result is cached in the WFE process. When the process restarts, the validation will be performed again.

The acceptable range of specified version numbers may change when WSS is updated through a service pack or other release.

2.7 Error Handling

There are no system-level error handling behaviors. In general, for errors returned as part of a protocol in this system, the technical documents for those protocols describe what the error means for the system when they are defined. How these errors are handed, based on the protocol description, is left to the implementer.

2.8 Coherency Requirements

This system has no special coherency requirements.

2.9 Security

This section describes two core aspects of the WSS security model: authentication and authorization.

Authentication is the part of the system that determines the current user's identity. This is the first step in managing the security of the system. WSS uses the authentication mechanism from an underlying platform, such as IIS and ASP.NET, to authenticate users.

WSS supports all of the authentication modes that IIS and ASP.NET support, including Active Directory, Forms, and WebSSO authentication. In Active Directory authentication mode, IIS authenticates the user, using Basic Authentication, Digital Certificate, NTLM, or Kerberos. In other authentication modes, WSS relies on ASP.NET authentication modules to authenticate users, which can also be created by third-party developers such as FormsAuthenticationModule or ADFSAuthenticationModule. For more information about Active Directory authentication, see section 2.9.2.

Authorization in WSS identifies which permissions are granted to which users on a given object. When a Web request (or some object model API code) attempts to access an object inside WSS, and

the caller has been authenticated, the authorization code is invoked to identify whether the access should be granted. In a trusted subsystem model, the WFE uses the IIS application pool account to access the contents in the Content Database, on behalf of the user, to access content rather than the account of the user who is using the site. For more information, see section 2.9.2.2. Therefore, the permissions check has to happen before WSS returns any page content back to the user.

The following sections describe the basic concepts pertaining to authorization.

2.9.1 Authorization for User and Group Administration

After the user has been identified (authenticated), WSS controls the user's authorization and determines which **permissions** are granted to that user on a given object.

WSS supports a number of security-related operations to control access to content stored in WSS. These operations are built around a few core concepts defined in WSS, which are described in detail in the following sections.

2.9.1.1 Individual User Permissions (Rights)

Individual permissions, also known as rights, grant the ability to perform specific actions. For example, the View Items permission grants a user the ability to view items in a list. WSS has a fixed set of permissions that may be granted to users. The full specification of the WSS Rights Mask is provided in the WSS Rights Mask section of [MS-WSSFO] section 2.2.2.13 for Windows® SharePoint® Services 3.0, and in [MS-WSSFO2] section 2.2.3.14 for Microsoft® SharePoint® Foundation 2010. The permissions directly related to file services scenarios are:

- Add Items
- Edit Items
- Delete Items
- View Items
- Open Items
- Browse Directories

In addition, the following set of permissions relate to permissions control:

- Manage Permissions
- Create Groups
- Enumerate Permissions
- Open (site, Web, list, folder)

2.9.1.2 Permission Level (Role)

A Role is a predefined set of permissions that grants users permission to perform related actions. Roles are defined at the Site level, where a site can inherit roles from its parent site or have roles unique to it. All permissions in WSS are managed through roles and all users will have roles. Rights are never directly assigned to a user. The default WSS permission levels, or roles, are:

- Limited Access

- Read, Contribute
- Design
- Full Control

For example, the Limited Access role includes permissions that allow users to view specific lists, document libraries, list items, folders, or documents, when given the appropriate permissions.

It is also possible to add custom role definitions to the collection of roles, to include the specific set of rights required for the role, or to remove role definitions. For example, a specific scenario might require a user role where the user cannot see previous versions of a document. To achieve this, it is possible to create a custom contributor role where the View Versions and Delete Versions rights have been removed.

For more information about creating and removing roles, see [MSDN-SHPTSDK] for Windows® SharePoint® Services 3.0, and [MSDN-SHPTSDK4] for Microsoft® SharePoint® Foundation 2010.

2.9.1.3 User

A User is an identity associated with a user account that can be authenticated to WSS. Permission levels can be directly assigned to users. After a user has been authenticated, that user's identity is represented by a WSS User Token, and their permissions are represented by the roles to which they are assigned, as described in section 2.9.2. Role assignment is per site, where WSS tracks which users (or groups) are assigned to which roles for each site. A user's complete set of permissions is an aggregation of two sets of roles:

- The roles of which the user is a direct member.
- The roles that the user acquires by being a member of a group or site group.

2.9.1.4 Group

A Group is an identity associated with a group of users within Active Directory (Windows security group). As far as account management is concerned, WSS treats groups similarly to user accounts. When a user interacts with a WSS environment, their Active Directory Group membership is determined and their membership within a Group is used to determine the effective role of the user.

2.9.1.5 Site Group

A WSS Site Group is a named logical grouping of user or group accounts. A Site Group can be set to specific roles or have rights granted to it. Each WSS Site Group is assigned a default role, but the role for any Site Group can be changed as necessary. Some predefined WSS Site Groups are as follows:

- Site Owners
- Site Members
- Site Visitors

WSS Group memberships are stored in SQL table named GroupMemberships. Each group is assigned an identifier that is unique within that Site Collection. Use of groups enables easier security management. When a large number of users have to be assigned the same role, administrators can easily create a WSS Group and assign those users as members and simply grant permissions to the

group rather than to each individual. Similarly, administrators can add new users to existing groups as a means of quickly giving users appropriate permissions. For more information about creating WSS Groups and adding users to existing groups, see:

- [MSDN-SHPTSDK] for Windows® SharePoint® Services 3.0
- [MSDN-SHPTSDK4] for Microsoft® SharePoint® Foundation 2010
- proc_SecCreateSiteGroup and proc_SecAddUserToSiteGroup in [MS-WSSFO] for Windows SharePoint Services 3.0
- proc_SecCreateSiteGroup and proc_SecAddUserToSiteGroup in [MS-WSSFO2] for SharePoint Foundation 2010

WSS Groups cannot be nested inside of each other. However, a WSS group can contain Active Directory groups as members.

WSS Groups are themselves a securable object in WSS with specific permissions to manage them, as described in section 2.9.1.2.

2.9.1.6 Securable Object

Users are assigned a permission level for a specific securable object: a Site, Library, Folder, or Document. By default, permissions for a Site, Library, or Document are inherited from the parent Site or Library.

Folders within lists are securable objects in that they derive their permissions from the underlying list items they contain. Pages that do not belong to any list are not securable objects; such pages always share the same permissions as their parent Site. Attached files (Attachments) and thumbnail files are also not securable objects; they always share the same permissions as their associated list item.

Each securable object gets its security permissions from its **access control list (ACL)** and other security metadata (for example owner info, checkout state, and so on). The security permissions can be unique, or could be inherited from the parent of the object.

2.9.1.7 Scope

A security Scope represents a URL subtree in WSS that shares the same permissions. A user creating an item in WSS could choose to give the item its own specific permission requirements or specify that it should inherit permissions from its parent. If the item has its own permissions, the item and its descendants form a scope, with the created item being the root of that scope. If the item inherits permissions, the item belongs to a larger scope that also contains its parent. A scope cannot span more than one site collection; however, it can span multiple sites within a site collection.

2.9.1.8 Inheritance

As mentioned earlier, an item in WSS can have its own specific permissions. However, default permissions for an item within a site are inherited from that site. This inheritance can be broken for any securable object at a lower level in the site hierarchy by creating a unique permissions assignment for that securable object.

For example, editing the permissions for a document library breaks its permission inheritance from its site. However, the inheritance is broken only for that particular document library; the rest of the

permissions for the site remain unchanged. An object may be reverted to inheriting permissions from its parent list or site at any time.

Sites are themselves a securable object to which permissions can be assigned. Sites contained within other sites can be configured to inherit permissions from a parent site or to create unique permissions for that particular site. If a child site inherits permissions from its parent, that set of permissions is shared with the child site. This effectively means that owners of the sites that inherit permissions from a parent site can change the permissions of the parent. To allow control of the permissions for the child site alone, the child site stops inheriting permissions, restricting the owner of the child site to only making changes to the permissions of the child site.

Creating unique permissions for a site stops permission inheritance. The groups, users, and permission levels from the parent site are copied to the child site and then the inheritance is broken. Reverting a site back from unique permissions to inherited permissions causes users, groups, and permission levels to once again be inherited and removes any users, groups, or permission levels that were uniquely defined in the site while inheritance was broken.

Permission levels (roles) can also be inherited. By default, permissions are defined such that the Read permission level is the same irrespective of the object to which it is applied. This type of inheritance can also be broken by editing the permission level. For example, an administrator might not require the Read permission level on a particular site to include the Create Alerts permission.

For more information about inheritance, see:

- [MSDN-SHPTSDK] for Windows® SharePoint® Services 3.0
- [MSDN-SHPTSDK4] for Microsoft® SharePoint® Foundation 2010
- proc_SecChangeToInheritedWeb and proc_SecChangeToUniqueScope in [MS-WSSFO] for Windows SharePoint Services 3.0
- proc_SecChangeToInheritedWeb and proc_SecChangeToUniqueScope in [MS-WSSFO2] for SharePoint Foundation 2010

2.9.1.9 Anonymous

WSS allows a specific type of access where the user is not uniquely authenticated, and thus is unknown to WSS. Such a user is referred to as the "Anonymous" user, or as having "Anonymous" access. The availability of anonymous access is controlled at the Web Application level of WSS. If anonymous access is allowed for the Web Application, then for example, Site administrators can decide whether to:

- Grant anonymous access to a site
- Grant anonymous access only to lists and libraries
- Block anonymous access to a site altogether

Anonymous access relies on the anonymous user account on the Web server. This account is created and maintained by Microsoft Internet Information Services (IIS), not by WSS. By default in IIS, the anonymous user account is IUSR_*ComputerName*. Enabling anonymous access essentially grants the anonymous account access to the WSS site. Allowing access to a site, or to lists and libraries, grants the View Items permission to the anonymous user account. However, even with the View Items permission there are restrictions to what anonymous users can do. For example, anonymous users cannot perform the following actions:

- Upload or edit documents into document libraries, including wiki libraries

- View the site in My Network Places

When the user who is accessing WSS items is anonymous, then the user identifier is null. In authentication modes other than Active Directory, such as Forms authentication, WSS is also impersonating IUSR_*ComputerName*. In those cases, WSS uses a user identifier generated from the ASP.NET Identity.Name value.

2.9.1.10 Anonymous Rights Mask (Anonymous Permissions Mask)

Use of an Anonymous Rights Mask (Anonymous Permissions Mask) is based on the concept of permissions for an anonymous user. The anonymous user can be given permissions via a Role, or can be assigned direct permissions on an item in WSS. Because, unlike other WSS users, the anonymous user's permissions cross Site collection boundaries and the user identifier is null (there is no "user" per se to have permissions), WSS uses an anonymous permission mask at the security scope level to make a security decision regarding whether the anonymous user has access to items within that scope.

2.9.1.11 System Account

Some features in WSS can trigger operations that require the ability to run with full permissions when the current user may not have permission to do so directly. For security reasons, the feature may not reveal the identity of the account with such permissions. This concept is known as "run as system account". When an account runs with "run as system account" permissions, operations performed by this account are recorded as executed by the "system account". The system account has its login name as SHAREPOINT\system. For example, when a list item is created by the application pool identity, it will show as "created by SHAREPOINT\system".

2.9.2 Authentication

WSS supports pluggable authentication, an extensibility mechanism provided by ASP.NET. By default, WSS uses one of three authentication modes against a Windows domain:

- Windows Integrated Authentication
- Basic Authentication
- Anonymous Authentication

Specific deployments may use a custom authentication provider to authenticate end-users against any third-party authentication system.

When used with Active Directory and a Windows Domain, WSS works with Active Directory for authentication of network accounts in the following contexts:

- **Authentication of the requests from the EUC.** The WFE establishes a specific End-User Identity for requests from the EUC. The WFE evaluates that End-User Identity against permissions associated with objects related to the request, to determine whether to execute the action for that request.
- **Authentication of the Process Account from the WFE.** The BEDS establishes an identity for requests from the WFE. The BEDS evaluates whether that identity has permissions to operate as a WSS WFE for content stored in the BEDS.
- **Creating a site collection local user record** for each logged-in user.

- **Updating the site collection local user record** to reflect a change in the user record in the Active Directory.
- **Selecting users and groups from the directory** for the purposes of setting security ACLs, as well as defining SharePoint Groups.
- **Creating Active Directory user accounts** in Account Creation mode to enable the creation of Active Directory accounts for WSS users.

Section 2.9.2.1 specifies how WSS uses the Active Directory Protocol [MS-ADTS] for the two types of authentication previously described.

2.9.2.1 Authentication of the Requests from the EUC

A typical scenario for authentication of the requests from the EUC occurs when a user is logged in using an Active Directory domain user account that allows the user to access a network resource such as a WSS site. When the user makes a page request to WSS, IIS handles the request and authenticates the user. IIS could choose one of the three methods: Windows Integrated Authentication, Basic Authentication, or Anonymous Authentication.

Once IIS has authenticated the user, IIS impersonates that user account for the thread handling the request. At this point, control is handed over to the WSS code to fulfill the request from the EUC. The request contains the unique address of a resource in WSS (a page, a document, a list item, and so on), and its associated ACL. The ACL specifies which **security principal** has what permissions on this object. It is possible that the object inherits its permissions from an object higher in the container hierarchy (for example, see the figure in section 2.1.2).

In addition to the resource's address, the request contains the action that has to be performed on the object, such as Read, Write, Delete, Check-out, and so on. The WSS authorization system performs the following steps to determine whether the requestor can perform the requested action on the request object.

1. When making the determination, the authorization function inspects the user's token (a data structure provided by Active Directory on the thread by IIS) containing the User **SID (Security Identifier)** and Security Group SIDs. It compares this against the list of referenced SIDs in the site collection. As a performance optimization, once this comparison is made, only the SIDs in the user's token that are referenced in the site collection are used.
2. The authorization code then uses the truncated user token and compares it against each **Access Control Entry** in the requested object's ACL. An ACE contains the security principal and the action(s) which the principal can perform on that object. By matching all of the principals in the ACEs against the user's token, the list of actions that the user can perform on the requested object is determined.
3. The final step is to compare the requested action against the list of actions that the user can perform as determined by the authorization algorithm. If the requested action is present in the list of authorized actions for the user, the request is allowed; otherwise, the request is denied.

2.9.2.2 Authentication of the Process Account from the WFE

When the WFE requires data from the BEDS, it makes a request for that data via the appropriate protocol. The BEDS has to validate whether the WFE has appropriate permissions to access the data in the BEDS.

The WSS worker process in the WFE runs under a service account identity defined in Active Directory. This account is assigned by the system or can be set by the administrator. The BEDS

validates the requestor's identity using either Windows Integrated Authentication (WIA) or SQL login. If WIA is chosen, the request made to the BEDS is done using the service account identity.

The BEDS then uses the requestor's identity (from the SPUser object, not the requestor's account) to authorize the request.

2.9.2.3 Creating a Site Collection Local Record of the User

In addition to authorizing user requests, WSS has to know the user's identity for other purposes, such as indicating who created a document, providing an e-mail address to deliver an alert, or associating a picture with a discussion post.

To optimize BEDS page rendering performance, some user information is copied from Active Directory and stored in the Content Database for that site collection. This allows the user's name, e-mail address or SIP address, and picture to be used in rendering a page without making a call to Active Directory.

When the user makes a request to see the list of documents in a document library, the name of the user who last created or updated the file is displayed. To avoid calling Active Directory for each document just to get the appropriate user name, the user names are stored in a User Info table in the Content Database. This allows the BEDS to determine the user name by doing a database Join instead of an off-computer remote call to Active Directory.

This User Info table is populated with an entry containing the User's SID, account name, name, e-mail address, SIP address, Title, and Department. The entry is created when one of the following actions occur:

1. A user is given access using the Site Administration pages.
2. A user is referenced by a list item (for example, Task is assigned to 'User A').
3. A user uploads or creates a document.
4. A user visits the site by requesting a document, page, or item from the site.

All of the fields for the user entry are found in the user's Active Directory User Object record. If the Active Directory user record is not populated with any information, the resulting entry in the WSS User Table also contains nothing. This is a one-time occurrence; the table is not regularly updated against Active Directory.

2.9.2.4 Updating the Site Collection Local User Record (Account Migration)

During a user's life cycle, a number of user attributes can change in Active Directory. Some common changes are name (user got married), domain (the user was migrated from one Active Directory domain to another), or **forest** (user was migrated from one Active Directory forest to another).

Because there is no automatic way to synchronize the user information in the User Info table of the Content Database with the Active Directory, a MigrateUser command line option is provided. The most common use of this command is for updating the user record when the user has been migrated from one domain to another within the same forest, or from one forest to another forest.

A local user record is identified with the user SID. The command to migrate the user takes the original user account name and the new account name and indicates whether to validate the SID history.

In the case of a domain-to-domain transfer within the same forest, validation of SID history is recommended. When the MigrateUser command is issued, the Windows® SharePoint® Services

WFE gets the new user SID from Active Directory from the new user account name, looks up the user table for the record under the old user SID, and updates that row with the new user SID. In effect, this converts the user from one SID to another. When SID history is turned on, the new user token is examined to make sure that it contains the old user SID.

In the case of a forest-to-forest transfer, it is not possible to verify the SID history because the Active Directory forest is the boundary for security principals. In this case, the new SID is looked up from the new Active Directory forest, the user table record that matches the old user SID is located, and the record is updated with the new SID.

2.9.2.5 Selecting Users and Groups from Active Directory

Active Directory is used by WSS as the directory containing the list of users and groups that can be used for securing a container in WSS. End users as well as administrators can look up users and groups, search for users and groups, and add one or more users and/or groups to the site.

Two basic functions are performed against Active Directory in order to select users and groups:

1. Resolve a name or ID.

2. Search for all matching records for a query.

In WSS, a user is able to invoke the user and group selector in the security setting UI. The control contains an entry box, a check names button, and a browse button.

The typical scenario involves a user adding a user name or user ID into the entry box and pressing the Check Names button. The WFE then sends the string to Active Directory and attempts to find a unique User or Security Group object that matches the text entered in the entry box. If Active Directory locates a unique match, the object's SID is returned, and the Resolve call is considered successful. For an example of this operation, see the description of the Active Directory: People Picker Check Name UI in section 3.3.

If a unique object is not found, the resolve call fails, and may return the list of matches for the query. The user interface marks the original text with a red underline, and when selected, will show all the possible matches returned from Active Directory in a drop-down listbox.

The other possibility is for the user to select the Browse button. This displays a pop-up window containing a search box and a results box. The user enters the query and selects the "Go" button to issue the query. The query is then sent to Active Directory, and all results are shown in the results box. The user is then able to select one or more of the results to add to the field. For an example of this operation, see the description of the Active Directory: People Picker Browse Display UI in section 3.2.

If no results are found, an informational message is displayed, indicating that no results matched the query term.

The User and Group selectors can be configured to select just users, just groups, or both.

2.9.2.6 Creating an Active Directory User Account

Windows® SharePoint® Services is often deployed in the intranet with Active Directory. In that case, Active Directory contains the list of users (for example, users of the corporation) who can potentially access the site. In this situation, it is impossible to invite someone who is not listed in Active Directory to view or participate in a WSS site.

WSS is also frequently used in the extranet to enable cross-company and/or cross-domain collaboration. In this case, it is not possible to pre-emptively place an exhaustive list of all users in all companies into Active Directory. In this situation, WSS is deployed in a mode called Active Directory Account Creation Mode. This allows WSS to create an account for a user when that user first attempts to interact with a WSS site.

For example, a team site administrator invites a partner using an e-mail address, username@example.com. WSS sends an e-mail to that address inviting the user to access the resources at that team site. When the prospective user first arrives at the WSS site, the user is asked to create a site account that identifies that user for future visits.

After the user completes the sign-up process, the WSS WFE creates a user account in Active Directory, and stores the new user's name, e-mail address, and other data. The location in Active Directory where these accounts are created, and the permissions to create accounts, are set up when WSS is configured. For an example of this operation, see the description of the Active Directory: Account Creation New UI in section 3.1.

2.10 Additional Considerations

There are no additional considerations.

3 Protocol Examples

These examples describe in detail the process of communication between the various server components involved in the WSS deployment. In conjunction with the technical protocol documents listed in section 2.2, these examples are intended to provide a comprehensive view of how WSS WFEs communicate with EUC, Active Directory Domain Control (DC), and BEDS systems.

3.1 Example 1: Active Directory: Account Creation New UI

This example describes the requests made when the user on an EUC computer fills in the e-mail Address and Display Name, and then clicks the "OK" button on the Create Active Directory Account – Web Page Dialog page to create a new Active Directory user account. The main member protocol used in this sequence is [MS-WSSFO] covering the stored procedures listed in the following steps. The sequence diagram has been broken into three figures for reasons of length. The three figures in this section should be viewed as a single sequence. This specific example is for Active Directory operations involving Windows® SharePoint® Services 3.0.

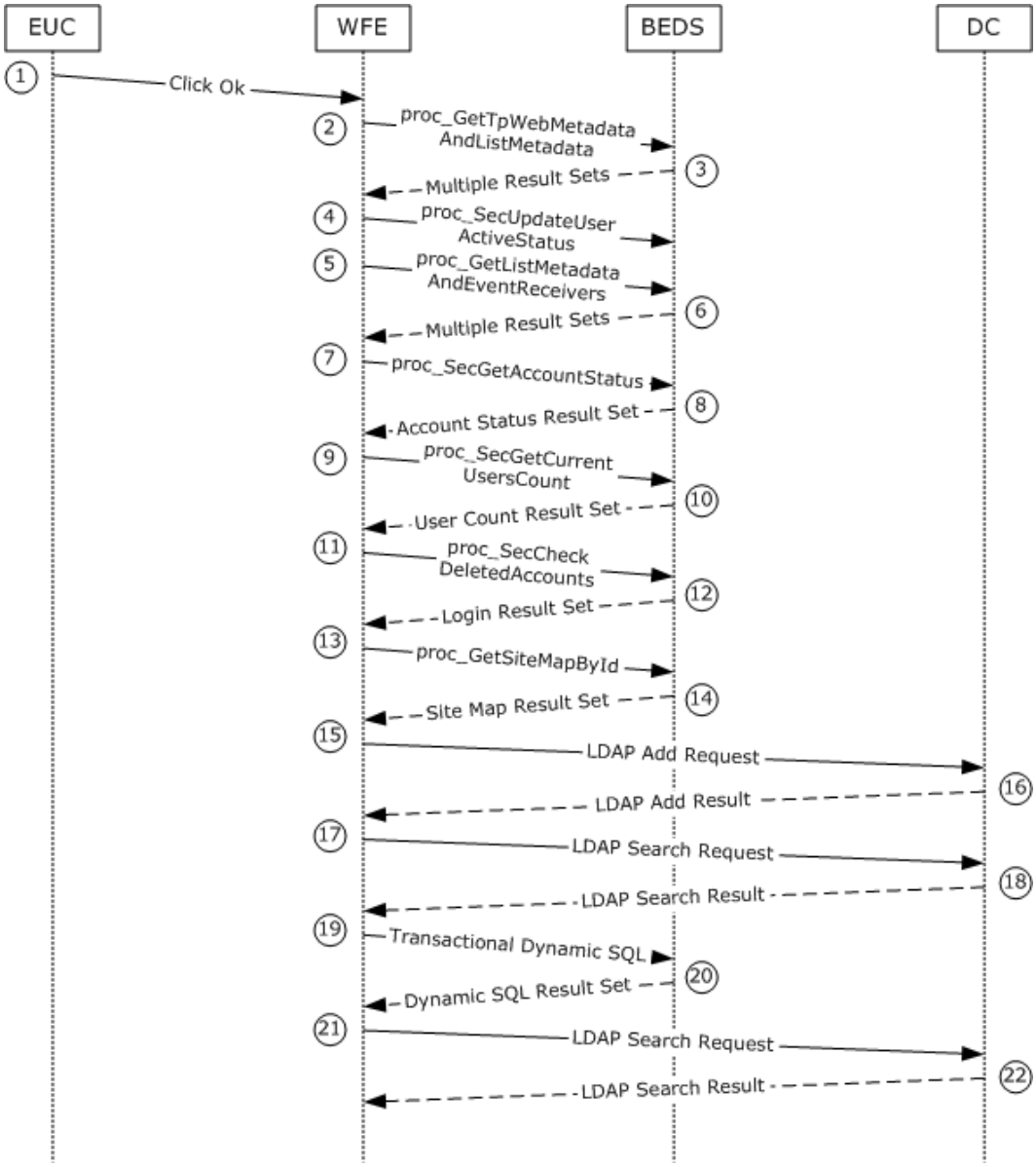

Figure 5: Account Creation New UI, steps 1 through 22

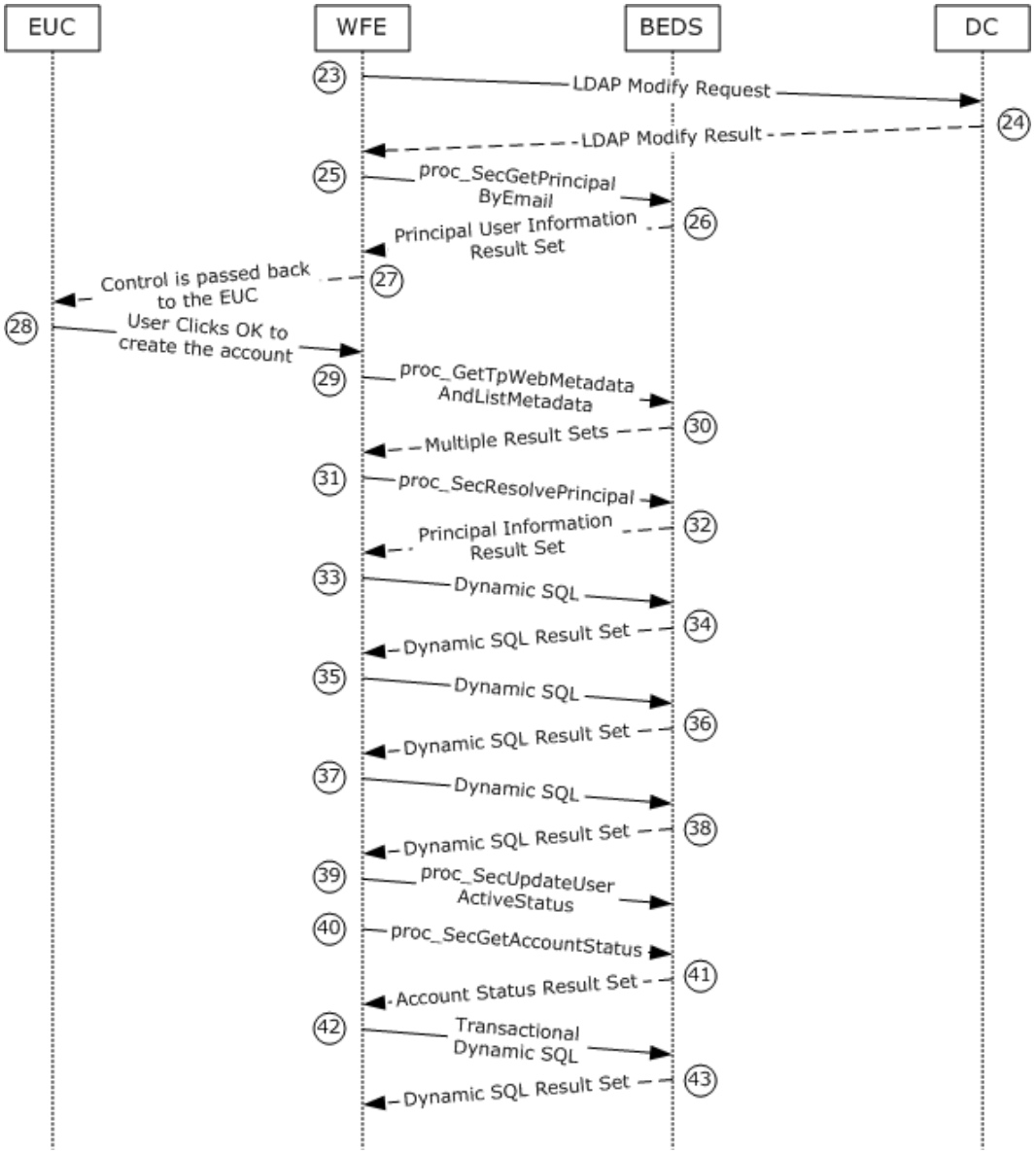

Figure 6: Account Creation New UI, steps 23 through 43

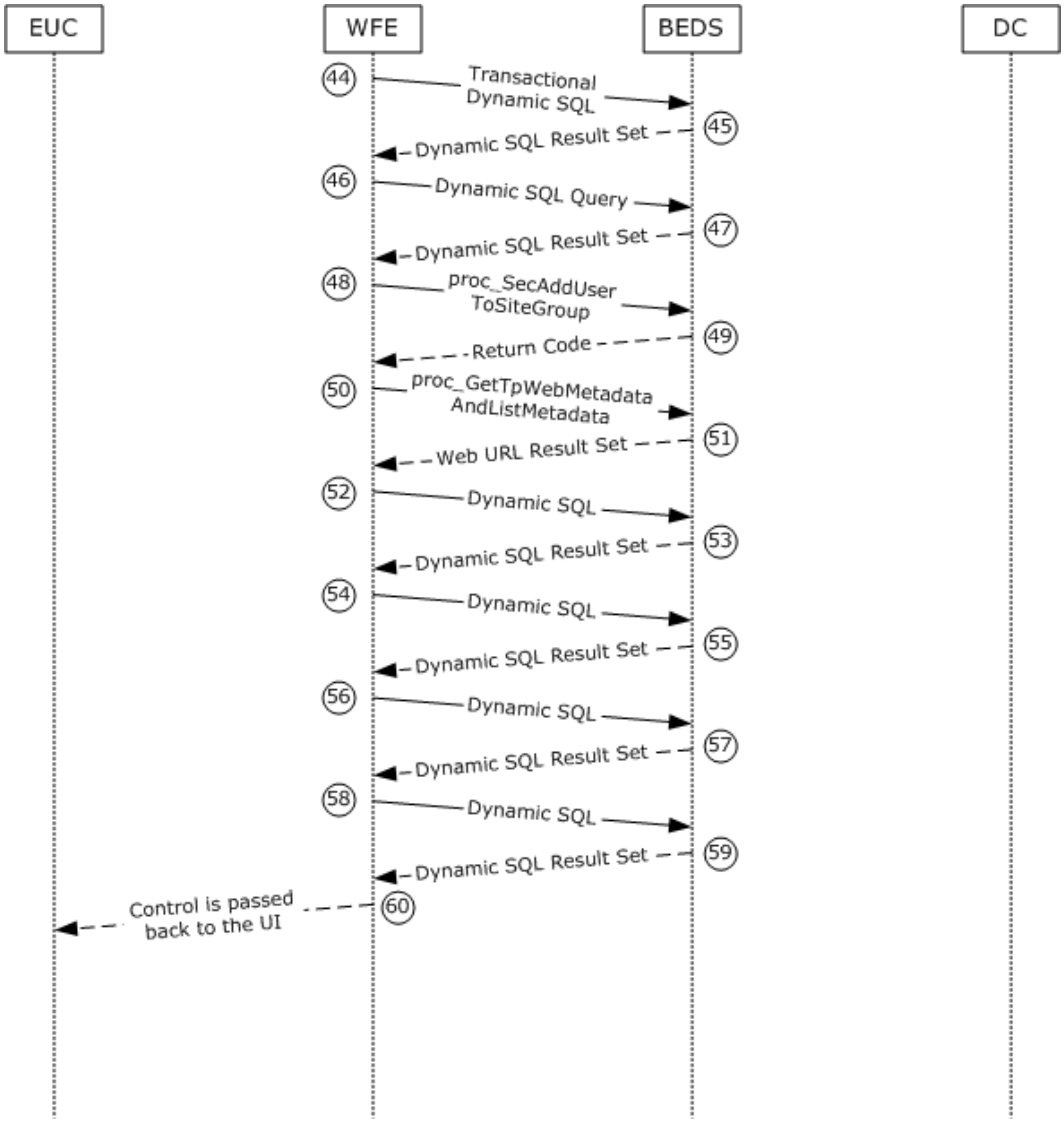

Figure 7: Account Creation New UI, steps 44 through completion

This scenario uses the WSS user interface (UI) to create a new account for a user. It assumes that a user on an EUC computer has selected the Add Users option on the people.aspx page under the New button. The user then clicks the Create button under the Users/Groups: Dialog box, and fills in a valid e-mail address and Display Name for a user who has never existed in this site collection. The following trace is then initiated when the user clicks the "OK" buttons on both the Create Active Directory Account page and the Add Users: Team Site page to add a new user to Active Directory.

The following actions happen:

1. The user clicks OK, which causes the EUC to start the UserAdd process via a page to post back to the WFE to verify that the new user isn't currently defined.

2. The WFE first gathers information about the current environment by calling the proc_GetTpWebMetadataAndListMetadata Stored Procedure.

3. The BEDS returns the following eight Result Sets:

    - Web URL Result Set, which returns the Store-Relative Form URL of the root of the Site.
    - Domain Group Cache Versions Result Set, which returns information about the version numbers associated with the Domain Group map cache for this Site.
    - Domain Group Cache WFE Update Result Set, which returns binary data needed to refresh the Domain Group map cache.
    - Site Metadata Result Set, which returns specialized Site Metadata.
    - Event Receivers Result Set, which returns information about the Event Receivers defined for the Site.
    - Site Feature List Result Set, which returns a List of default Feature identifiers for this Site Collection that contains this Site.
    - Site Feature List Result Set, which returns a List of Feature identifiers for this Site.
    - Empty Result Set, which is a placeholder set returned because the Site has no cached navigation Scope information.

4. The WFE invokes the proc_SecUpdateUserActiveStatus stored procedure to update the list of active users for the Site.
5. The WFE continues collecting information about the current user list by calling the proc_GetListMetadataAndEventReceivers Stored Procedure using TDS.
6. The BEDS returns the following two Result Sets:

    - List Metadata Result Set, which returns the Metadata associated with this List.
    - Event Receivers Result Set, which returns information about the Event Receivers defined for this List.

7. The WFE checks for the existence of the new account by calling the proc_SecGetAccountStatus Stored Procedure.
8. The BEDS returns an Account Status Result Set with zero rows, indicating that the e-mail address has not yet been used in this Site Collection.
9. The WFE gathers further information about the current count of registered users by calling the proc_SecGetCurrentUsersCount Stored Procedure.
10. The BEDS returns the User Count Result Set to indicate the number of users registered with this Site Collection and any quota information.
11. The WFE verifies that the UserID to be created does not exist in the deleted user list by calling the proc_SecCheckDeletedAccounts Stored Procedure.
12. The BEDS returns the Login Result Set with zero results, indicating the UserID does not yet exist.
13. The WFE gathers further information about the current site by calling the proc_GetSiteMapById Stored Procedure.
14. The BEDS returns the Site Map By Id Result Set with one row of data on the current site.

15. The WFE makes an LDAP AddRequest to the DC to add a User object with the supplied information.

16. The DC sends an LDAP AddResponse indicating a successful insertion.

17. The WFE makes an LDAP Search Request to the DC to confirm that the recently added User object exists.

18. The DC sends an LDAP Search Response indicating that the User object exists.

19. The WFE builds a transactional dynamic SQL query which performs the following tasks:

    - The query begins a new SQL transaction.
    - The proc_SecAddUser Stored Procedure is executed to add the requesting user to Active Directory environment.
    - In the event of any error, the transaction is rolled back, and the final select statement is returned.
    - If the new user account does not already exist, it is added to the User List using the proc_AddListItem Stored Procedure.
    - In the event of any error, the transaction is rolled back, and the final select statement is returned.
    - The transaction is committed, and the final select statement is returned with the variables used to create the new User account.

20. The BEDS returns a Dynamic SQL Result Set, indicating that the new user account has been successfully added, and displaying the variables used when creating the new account.

21. The WFE then makes an LDAP Search Request to the DC to request attributes for the recently added User object.

22. The DC sends an LDAP Search Response indicating the additional User object attributes.

23. The WFE then makes an LDAP Modify Request to the DC to modify the mail attribute for the added User object.

24. The DC sends an LDAP Modify Response is response, indicating the successful attribute change.

25. The WFE verifies the availability of the new account's e-mail address by calling the proc_SecGetPrincipalByE-mail Stored Procedure.

26. The BEDS returns the Principal User Information Result Set, containing information about the user associated with the specified e-mail address.

27. Control is passed back to the EUC on the Add Users page, with the new user listed in the Users/Groups dialog box.

28. The User clicks OK on the Add Users page to add the newly created user to the Site Group.

29. The WFE requests status data by calling the proc_GetTpWebMetadataAndListMetadata Stored Procedure.

30. The BEDS returns the following 14 Result Sets:

- Web URL Result Set, which returns the URL of the root of the Site.
- Domain Group Cache Versions Result Set, which returns information about the version numbers associated with the Domain Group map cache for this Site.
- Domain Group Cache WFE Update Result Set, which returns binary data needed to refresh the Domain Group map cache.
- Site Metadata Result Set, which returns specialized Site Metadata.
- Event Receivers Result Set, which returns information about the Event Receivers defined for this Site.
- Site Category Result Set, which returns categories of this Site.
- Site Metainfo Result Set, which returns the specialized Site Metadata.
- Site Feature List Result Set, which returns a List of default Feature identifiers for the Site Collection that contains this Site.
- Site Feature List Result Set, which returns a List of Feature identifiers of this Site.
- Empty Result Set, which is a placeholder set returned because the Site has no cached navigation Scope information.
- List Metadata Result Set, which returns the Metadata associated with the specified Document List.
- NULL Unique Permissions Result Set, which is a placeholder set returned because the List has no individual List Permissions.
- Event Receivers Result Set, which returns information about the Event Receivers defined for the Document List.
- List Web Parts Result Set, which returns information about the List Web Parts defined for this Document List.

31. The WFE requests information about the new account by calling the proc_SecResolvePrincipal Stored Procedure.
32. The BEDS Returns the Principal Information Result Set with a single row containing basic information about the user.
33. The WFE creates a Dynamic SQL query, which selects information from the UserData view joined with the Docs view.
34. The BEDS Returns a Dynamic SQL Result Set, which contains one row of data with the new user account information.
35. The WFE creates a Dynamic SQL query, which selects information from Sec_SiteGroupsView.
36. The BEDS Returns a Dynamic SQL Result Set with all Site Group Membership Levels.
37. The WFE creates a Dynamic SQL query to check the requesting User permissions for this activity by calling the proc_SecGetUserPermissionOnGroup Stored Procedure.
38. The BEDS Returns a Dynamic SQL Result Set representing the permission levels of the calling user.

39. The WFE invokes the proc_SecUpdateUserActiveStatus stored procedure to update the list of active users for the Site.

40. The WFE requests the account status for the new account by calling the proc_SecGetAccountStatus Stored Procedure.

41. The BEDS returns the Account Status Result Set.

42. The WFE builds a Transactional Dynamic SQL query which performs the following tasks:

    - The query begins a new SQL transaction.
    - The proc_SecAddUser Stored Procedure is executed to add the requesting user to the appropriate security groups in the Active Directory environment.
    - In the event of any error, the transaction is rolled back, and the final select statement is returned.
    - If the new user account does not already exist, it is added to the User List using the proc_AddListItem Stored Procedure.
    - In the event of any error, the transaction is rolled back and the final select statement is returned.
    - The transaction is committed, and the final select statement is returned with the variables used to create the new User account.

43. The BEDS returns a Dynamic SQL Result Set indicating the success of the add procedures and the variables used in the new account.

44. The WFE builds a Transactional Dynamic SQL query, which performs the following tasks:

    - The query begins a new SQL transaction.
    - The proc_AddListItem Stored Procedure is executed to add the new User account to the appropriate WSS List.
    - In the event of any error, the transaction is rolled back and the final select statement is returned.
    - The new User account is updated, using the proc_UpdateListItem Stored Procedure with additional User data as necessary.
    - In the event of any error, the transaction is rolled back and the final select statement is returned.
    - The transaction is committed, and the final select statement is returned with the variables used to create the new User account.

45. The BEDS returns a Dynamic SQL Result Set with information about the newly created user.

46. The WFE creates a Dynamic SQL query to either add the User Data by calling the proc_AddListItem Stored Procedure, or to update the User Data by calling the proc_UpdateListItem Stored Procedure.

47. The BEDS returns the following two Result Sets:

    - Item Update Result Set, which returns pertinent information about the update.

- Dynamic SQL Result Set, which returns the output status value from the update.

48. The WFE can now add the new user account to the appropriate site group by calling the proc_SecAddUserToSiteGroup Stored Procedure.

49. The BEDS responds with a return code, but no Result Sets are returned.

50. The WFE requests further status data by calling the proc_GetTpWebMetadataAndListMetadata Stored Procedure.

51. The BEDS returns the following 14 Result Sets:

    - Web URL Result Set, which returns the URL of the root of the Site.
    - Domain Group Cache Versions Result Set, which returns information about the version numbers associated with the Domain Group map cache for this Site.
    - Domain Group Cache WFE Update Result Set, which returns binary data needed to refresh the Domain Group map cache.
    - Site Metadata Result Set, which returns specialized Site Metadata.
    - Event Receivers Result Set, which returns information about the Event Receivers defined for this Site.
    - Site Category Result Set, which returns categories of this Site.
    - Site Metainfo Result Set, which returns the specialized Site Metadata.
    - Site Feature List Result Set, which returns a List of default Feature identifiers for the Site Collection that contains this Site.
    - Site Feature List Result Set, which returns a List of Feature identifiers of this Site.
    - Empty Result Set, which is a placeholder set.
    - List Metadata Result Set, which returns the Metadata associated with the specified Document List.
    - NULL Unique Permissions Result Set, which is a placeholder set.
    - Event Receivers Result Set, which returns information about the Event Receivers defined for the Document List.
    - List Web Parts Result Set, which returns information about the List Web Parts defined for this Document List.

52. The WFE creates a Dynamic SQL query selecting all information from the Sec_SiteGroupsView view for this Site to generate the final Web site display.

53. The BEDS returns a Dynamic SQL Result Set with all Site GroupMembership Levels.

54. The WFE creates a Dynamic SQL query, selecting information from the UserData view, Docs view, and AllUserData view for details on the web site display.

55. The BEDS returns a Dynamic SQL Result Set of User data as it applies to the current display.

56. The WFE creates a Dynamic SQL request for user permission information by calling the proc_SecGetUserPermissionOnGroup Stored Procedure.

57. The BEDS returns a Dynamic SQL Result Set representing the permission levels of the calling user.

58. The WFE creates a Dynamic SQL request for information from the UserData view and Docs view for Web site display.

59. The BEDS returns a Dynamic SQL Result Set of User data as it applies to the current display.

60. Control is then returned to the UI.

3.2 Example 2: Active Directory: People Picker Browse Display UI

This example describes the requests that are made when a search for a valid Active Directory User is made from the EUC computer by entering a search string that matches a User's display name, and when that user is located, that User is added to the current Site. The main member protocol used in this sequence is [MS-WSSFO] covering the stored procedures listed in the steps. The sequence diagram has been broken into three figures for reasons of length. The three figures in this section should be viewed as a single sequence. This specific example is for Active Directory operations involving Windows® SharePoint® Services 3.0.

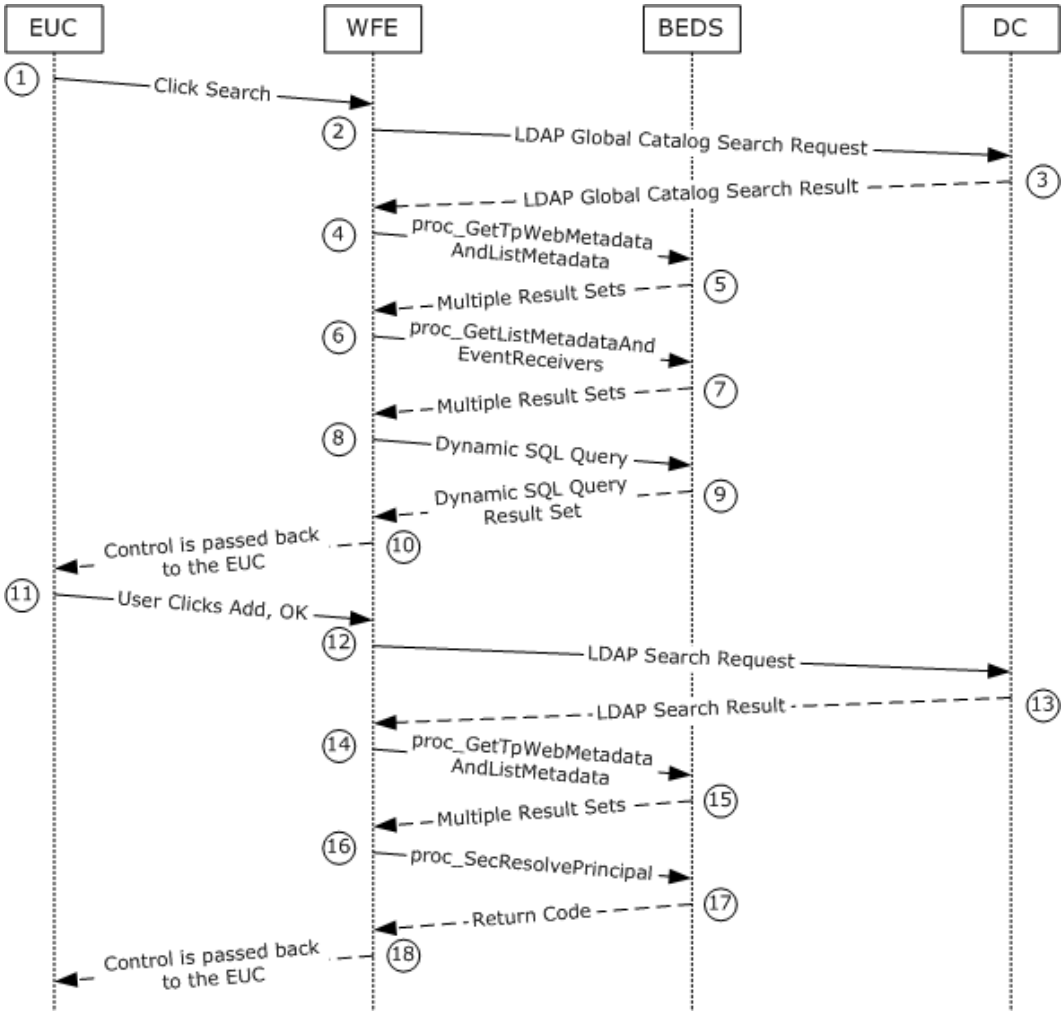

Figure 8: People Picker Browse Display UI, steps 1 through 18

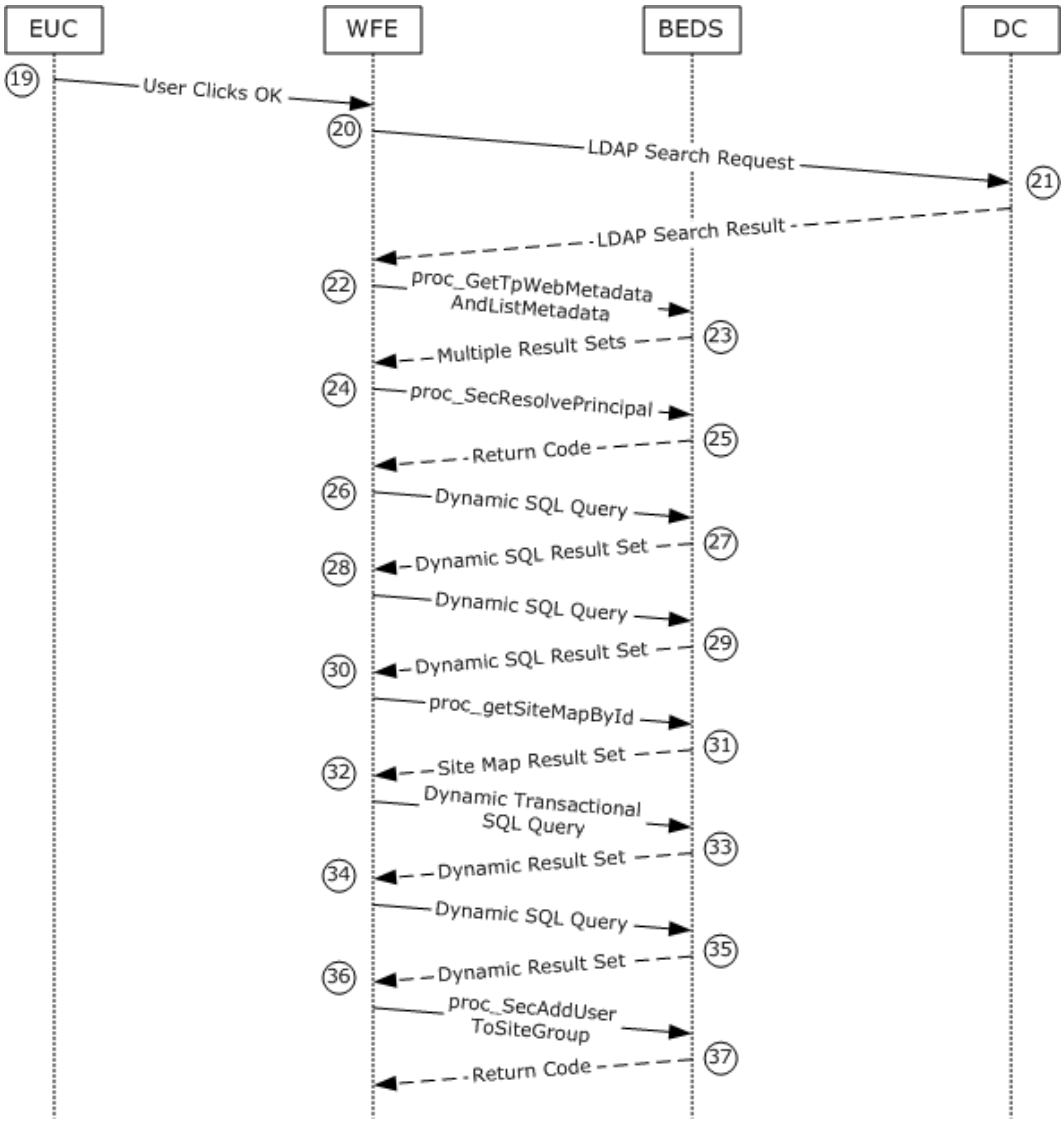

Figure 9: People Picker Browse Display UI, steps 19 through 37

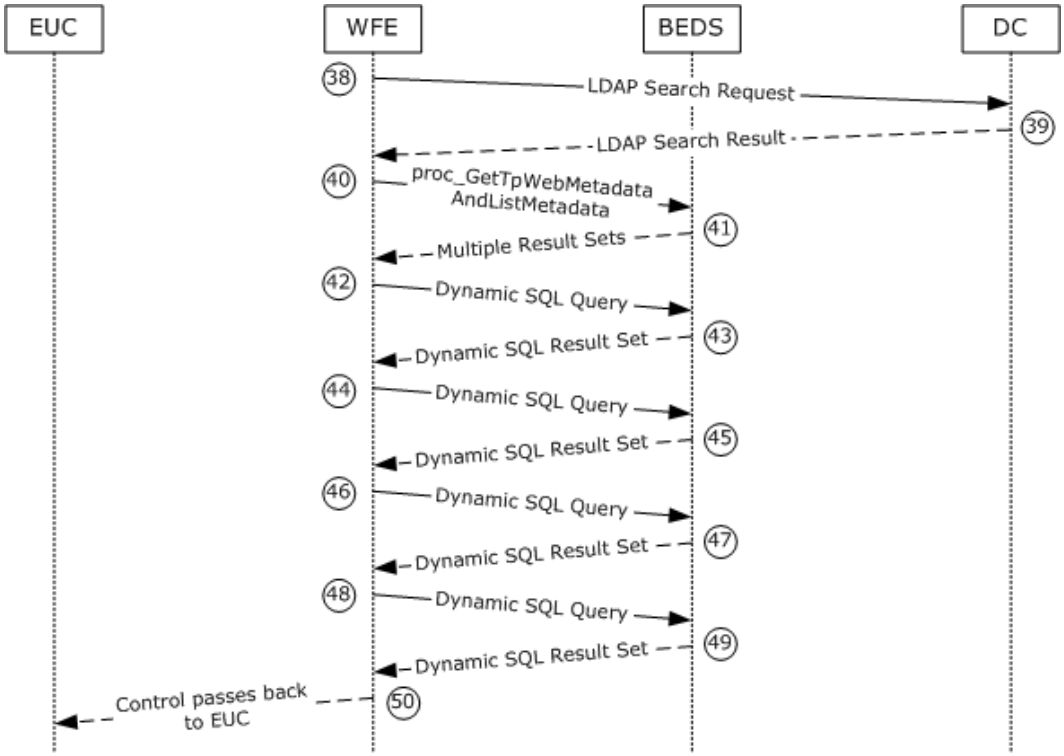

Figure 10: People Picker Browse Display UI, steps 38 through completion

This scenario is initiated from the "Select People and Groups – Web page Dialog." A user enters a search string in the "Find" text field and then clicks the search icon. For the sake of simplicity, it is assumed that the user has Add privileges for the current Site Group.

The following actions happen:

1. The EUC first sends a request to the WFE to search for the desired User display name.
2. The WFE sends an LDAP Global Catalog Search Request to the DC asking for any match in the whole subtree for user or group objects with attributes that contain the search string (a wildcard search version of the User display name) in one of the following attributes:
   - User objects: 'name', 'displayName', 'cn', 'sn', 'SamAccountName', 'mail', SMTP or sip 'proxyAddresses' attributes.
   - Group objects: 'name', 'displayName', 'cn', or 'SamAccountName' attributes.
3. The DC responds with an LDAP Global Catalog Search Response containing both user and group objects that match the search string.
4. The WFE initializes information about the Site and its Users by calling the proc_GetTpWebMetadataAndListMetadata Stored Procedure using TDS.
5. The BEDS returns five Result Sets:
   - Web URL Result Set, which returns the Store-Relative URL of the root of the Site.

- Domain Group Cache Versions Result Set, which returns information about the version numbers associated with the Domain Group map cache for this Site.
- Domain Group Cache WFE Update Result Set, which returns information to be used in recomputing the Domain Group map cache for the Site.
- Site Metadata Result Set, which returns specialized Site Metadata.
- Event Receivers Result Set, which returns information about the Event Receivers defined for the Site.

6. The WFE continues collecting information about the Site's User List by calling the proc_GetListMetadataAndEventReceivers Stored Procedure.
7. The BEDS returns the following four Result Sets:
   - The List Metadata Result Set, which returns the permissions associated with the User List.
   - The NULL Unique Permissions Result Set, which indicates that unique permissions do not exist for the List.
   - The List Event Receivers Result Set, which is empty because there are no Event Receivers defined for this List.
   - The List Web Parts Result Set, which contains information about the List view pages.
8. The WFE creates a Dynamic SQL query that searches for the submitted search string in the user information list, looking for a match in the display name, account name or e-mail address columns.
9. The BEDS returns one empty Dynamic SQL Result Set, indicating that a match was not found.
10. The WFE displays the display name received from the DC as a candidate for selection.
11. The end user clicks "Add," then "OK". The EUC closes the dialog and redirects the user to the User Information List Web page.
12. The WFE negotiates authentication with the DC and then sends an LDAP search request to the DC for an object that has a SID attribute equal to the value obtained from the DC in Step 3.
13. The DC sends an LDAP Search Result containing the attributes of the Active Directory User object.
14. The WFE again initializes by gathering information about the Site by calling the proc_GetTpWebMetadataAndListMetadata Stored Procedure.
15. The BEDS returns five Result Sets:
    - The Web URL Result Set, which contains the Store-Relative URL of the root of the Site.
    - The Domain Group Cache Versions Result Set, which contains information about the version numbers associated with the Domain Group map cache for this Site.
    - The Domain Group Cache WFE Update Result Set, which contains information to be used in recomputing the Domain Group map cache for the Site.
    - The Site Metadata Result Set, which contains Site Metadata.

- The Event Receivers Result Set, which contains information about the Event Receivers defined for the Site.

16. The WFE sends a request to the BEDS to find security principals that might have login name, display name, or e-mail information matching the user account name returned from the DC. It does so by calling the proc_SecResolvePrincipal Stored Procedure.

17. The BEDS responds with a return code, but no Result Sets are returned, indicating that no matches were found.

18. The WFE renders the name as resolved.

19. The end user clicks "OK" on the "Add Users" page, sending a request to the WFE to add the user to the Site and Site Group.

20. The WFE negotiates authentication with the DC, and then sends an LDAP search request to the DC for an object that has a SID attribute equal to the value obtained from the DC in Step 3.

21. The DC sends an LDAP Search Result containing the attributes of the Active Directory User object.

22. The WFE initializes again by calling the proc_GetTpWebMetadataAndListMetadata Stored Procedure.

23. The BEDS returns the following 14 Result Sets:

    - The Web URL Result Set, which contains the URL of the Site.
    - The Domain Group Cache Versions Result Set, which contains information about the version numbers associated with the Domain Group map cache for this Site.
    - The Domain Group Cache WFE Update Result Set, which contains binary data needed to refresh the Domain Group map cache.
    - The Site Metadata Result Set, which contains Site Metadata.
    - The Event Receivers Result Set, which contains information about the Event Receivers that are defined for this Site.
    - The Site Category Result Set, which contains the categories of this Site.
    - The Site Metainfo Result Set, which contains the specialized Site Metadata.
    - The Site Feature List Result Set, which contains the List of default Feature identifiers for the Site Collection that contains this Site.
    - The Site Feature List Result Set, which contains the List of Feature identifiers of this Site.
    - An Empty Result Set, which is a Placeholder set.
    - The List Metadata Result Set, which contains the Metadata associated with the specified Document List.
    - The NULL Unique Permissions Result Set, which indicates that there are no special permissions set on the User information list.
    - The Event Receivers Result Set, which contains information about the Event Receivers defined for the Document List.

- The List Web Parts Result Set, which contains information about the List view pages defined for the user information List.

24. The WFE sends a request to resolve the selected user names by calling the proc_SecResolvePrincipal Stored Procedure.

25. The BEDS responds with a Return Code, but no Result Sets are returned, indicating that the user was not found.

26. The WFE creates a Dynamic SQL query that selects information from the Sec_SiteGroupsView.

27. The BEDS Returns a Dynamic SQL Result Set with all Site Group Membership Levels signifying the owner of all groups.

28. The WFE builds a Dynamic Query to determine whether the current user has permission to add a user to the group. It does this by calling the proc_SecGetUsersPermissionsOnGroup Stored Procedure.

29. The BEDS returns one Dynamic SQL Result Set, which contains one record for the current group, indicating that the current user does not directly have permission to add a user to the group, and is not the owner of the group.

30. The WFE requests the site map by calling the proc_getSiteMapById Stored Procedure.

31. The BEDS returns the Site Map By Id Result Set.

32. The WFE builds a Dynamic Transactional SQL Query to add the User to the Site Collection. The following actions happen:

    1. The transaction begins.

    2. An attempt to add a user to the UserInfo table is performed by calling the proc_SecAddUser Stored Procedure.

    3. If adding the user succeeded, then an attempt to add a person List Item to the User Information List is performed. It does so by calling the proc_AddListItem stored procedure.

    4. If either adding the User to the Site Collection or adding the List Item to the User Information List failed, then the transaction is rolled back; otherwise, the transaction is committed.

33. One result is returned from the BEDS, containing the Return Code and information about the added User.

34. The WFE constructs a Dynamic SQL query, selecting full User information about the added User.

35. The BEDS returns a Dynamic Result Set with the requested information.

36. The WFE requests the BEDS to add the User to the current Site Group by calling the proc_SecAddUserToSiteGroup Stored Procedure.

37. The BEDS responds with a Return Code, but no Result Sets are returned.

38. The WFE negotiates authentication with the DC, and then sends an LDAP search request to the DC for an object that has a SID attribute equal to the value obtained from the DC in Step 3.

39. The DC sends an LDAP Search Result containing the attributes of the Active Directory User object.

40. The WFE again initializes its information about the Site by calling the proc_GetTpWebMetadataAndListMetadata Stored Procedure.

41. The BEDS returns the following 14 Result Sets:

    - Web URL Result Set, which returns the URL of the root of the Site.
    - Domain Group Cache Versions Result Set, which returns information about the version numbers associated with the Domain Group map cache for this Site.
    - Domain Group Cache WFE Update Result Set, which returns binary data needed to refresh the Domain Group map cache.
    - Site Metadata Result Set, which returns specialized Site Metadata.
    - Event Receivers Result Set, which returns information about the Event Receivers defined for this Site.
    - Site Category Result Set, which returns the Categories of the Site.
    - Site Metainfo Result Set, which returns the specialized Site Metadata.
    - Site Feature List Result Set, which returns the List of default Feature identifiers for the Site Collection that contains this Site.
    - Site Feature List Result Set, which returns the List of Feature identifiers of this Site.
    - Empty Result Set, which is a placeholder set.
    - List Metadata Result Set, which returns the Metadata associated with the specified Document List.
    - NULL Unique Permissions Result Set, which is a placeholder set.
    - Event Receivers Result Set, which returns information about the Event Receivers defined for the Document List.
    - List Web Parts Result Set, which returns information about the List Web Parts defined for this Document List.

42. The WFE creates a Dynamic SQL query that selects information from the Sec_SiteGroupsView view.

43. The BEDS returns a Dynamic SQL Result Set with all Site Group Membership Levels, signifying the owner of all groups.

44. The WFE builds a Dynamic SQL Query to obtain updated information about the Site Group to which the User was added.

45. The BEDS returns one Dynamic SQL Result Set containing information about the Site Group.

46. The WFE builds a Dynamic Query to determine whether the current user has permission to add a user to the group. It does this by calling the proc_SecGetUsersPermissionsOnGroup Stored Procedure.

47. The BEDS returns one Dynamic SQL Result Set, which contains one record for the current group, indicating that the current User does not directly have permission to add a user to the group and is also not the owner of the group.

48. The WFE builds a Dynamic SQL Query to obtain more User information for the Site Group to which the User has been added.

49. The BEDS returns one Dynamic SQL Result Set of information about the newly added User.

50. Control is passed back to the EUC.

3.3 Example 3: Active Directory: People Picker Check Name UI

This example describes the requests made when the User is adding a new Member to a SharePoint List and uses the "Check Names" function to confirm the existence of the new Member in Active Directory. The main member protocol used in this sequence is [MS-WSSFO] covering the stored procedures listed in the steps. The sequence diagram has been broken into three figures for reasons of length. The three figures in this section should be viewed as a single sequence. This specific example is for Active Directory operations involving Windows® SharePoint® Services 3.0.

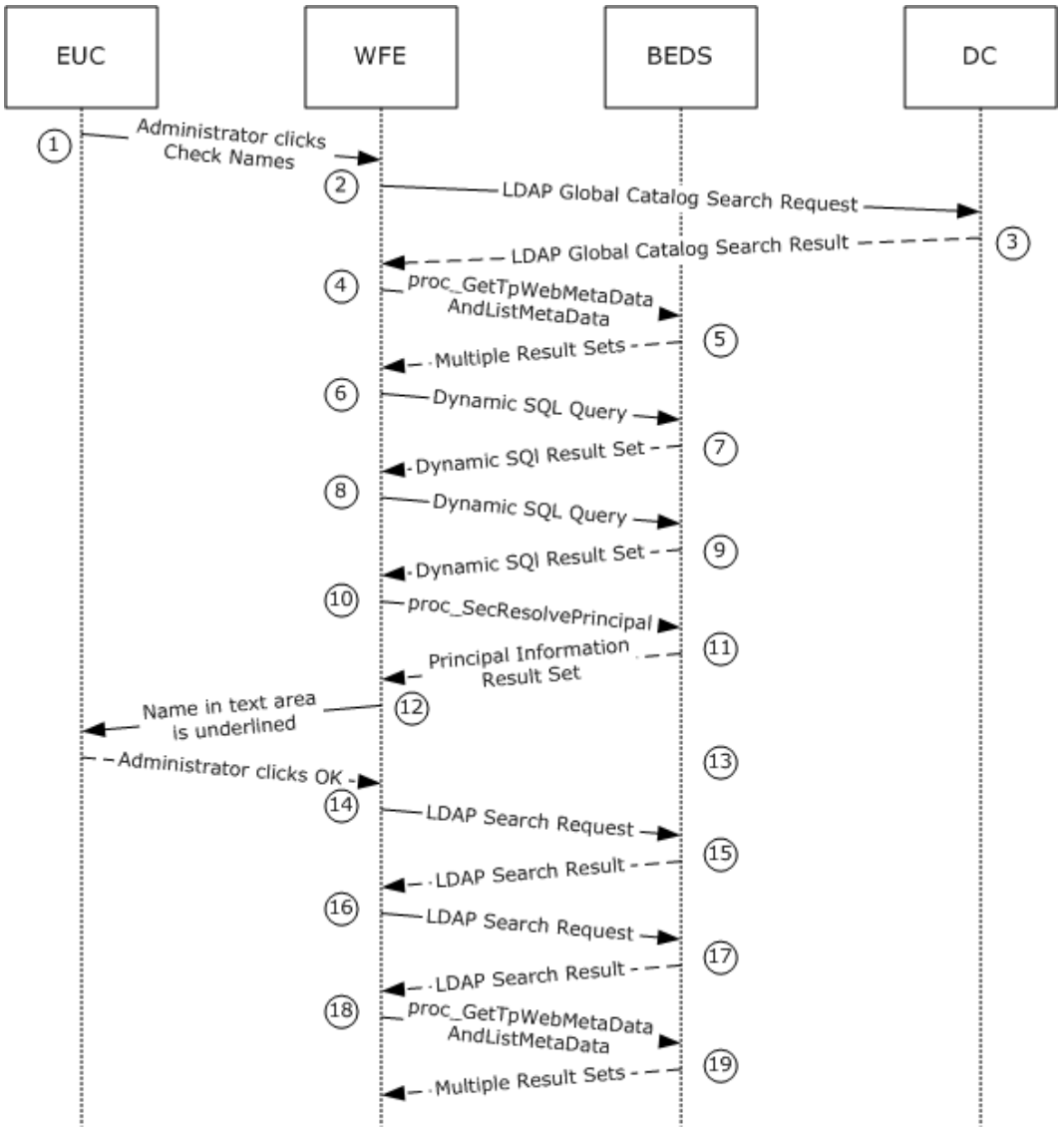

Figure 11: People Picker Check Name UI, steps 1 through 19

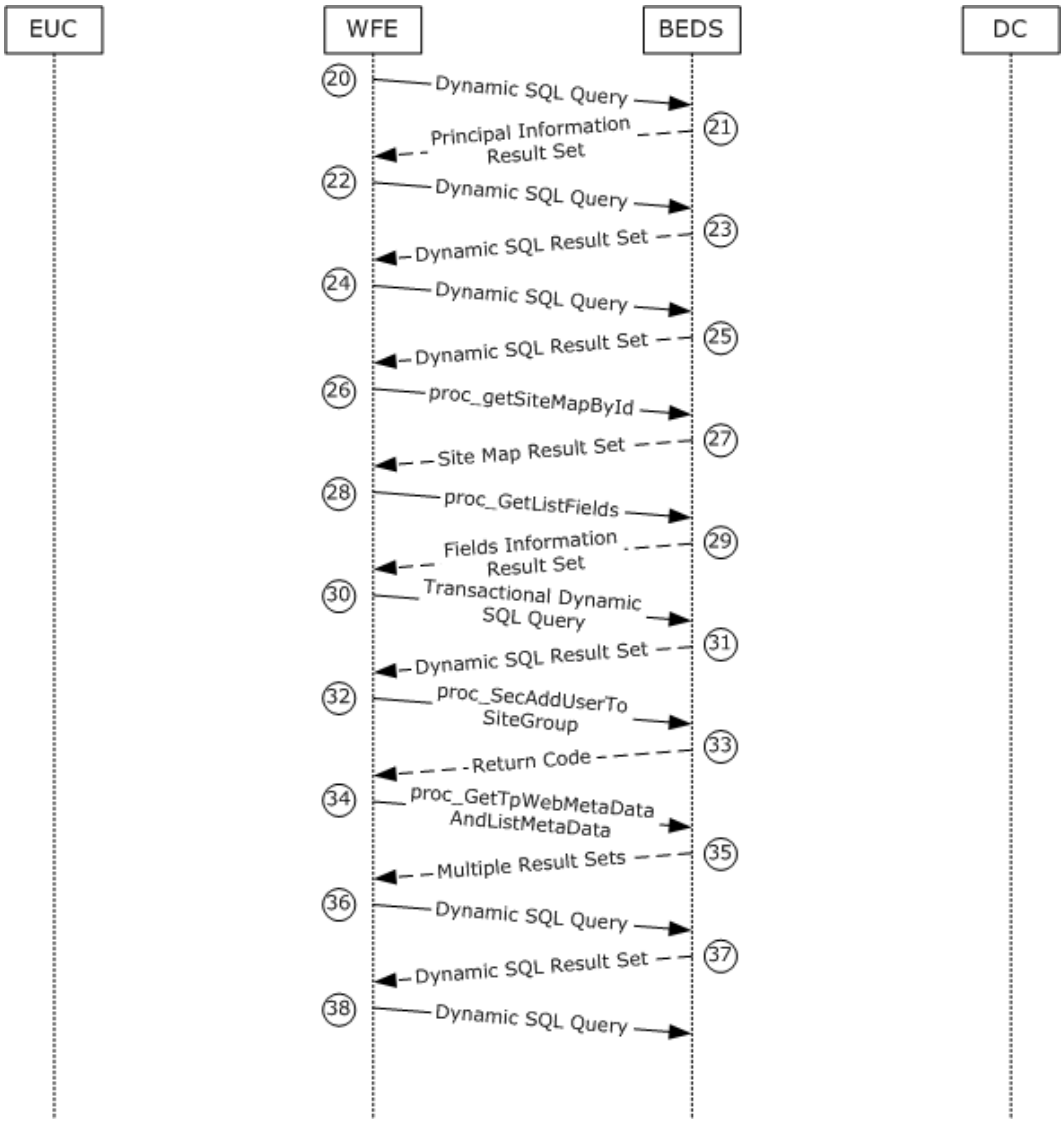

Figure 12: People Picker Check Name UI, steps 20 through 38

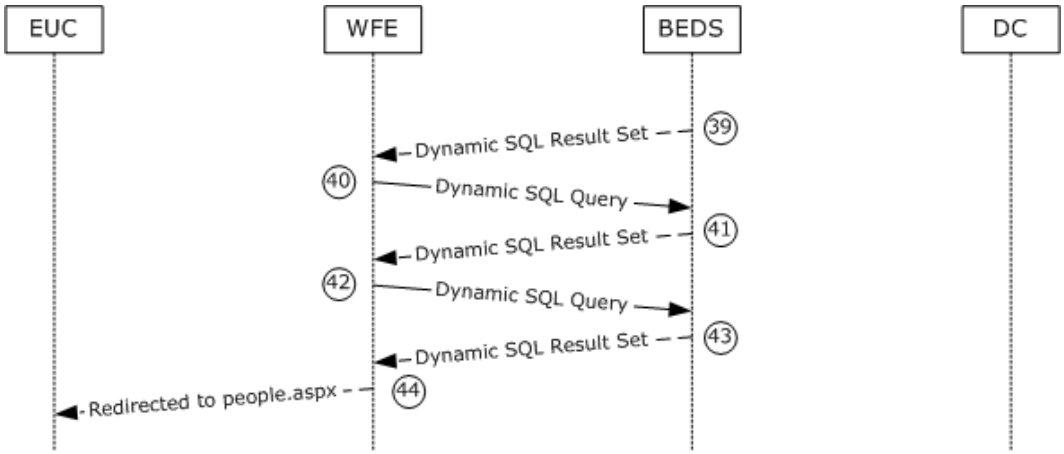

Figure 13: People Picker Check Name UI, steps 39 through completion

This scenario is initiated when the User clicks the "Check Names" icon. The following assumptions are made for the purposes of this example:

- The WSS Site is already established.
- The user making the request has permission to do so.
- The Name to be added already exists in Active Directory.

The following actions happen:

1. The User clicks the "Check Names" icon.
2. The WFE sends an LDAP Global Catalog Search Request to the DC, asking for any match in the whole subtree for user or group objects with attributes that contain the search string (a wildcard search version of the User display name) in one of the following attributes:
   - User objects: 'name', 'displayName', 'cn', 'samAccountName', 'mail', SMTP or sip 'proxyAddresses' attributes.
   - Group objects: 'name', 'displayName', 'cn', or 'samAccountName' attributes.
3. The DC responds with an LDAP Global Catalog Search Response containing both user and group objects that match the search string.
4. The WFE first collects Metadata for the Site and the Document List by calling the proc_GetTpWebMetaDataAndListMetaData Stored Procedure using TDS.
5. The BEDS returns 14 Result Sets:
   - Web Url Result Set, which returns the Store-Relative Form URL of the root of the Site.
   - Domain Group Cache Versions Result Set, which returns information about the version numbers associated with the Domain Group map cache for the Site.
   - Domain Group Cache WFE Update Result Set, which returns the binary data needed to refresh the Domain Group map cache.
   - Site Metadata Result Set, which returns specialized Site Metadata.

- Event Receivers Result Set, which returns information about the Event Receivers defined for the Site.
- Site Category Result Set, which returns the Categories of the Site.
- Site MetaInfo Result Set, which returns the specialized Site Metadata.
- Site Feature List Result Set, which returns a list of Feature identifiers for the Site Collection that contains this Site.
- Site Feature List Result Set, which returns a list of Feature identifiers for the Site.
- Empty Result Set, which is returned because the METADATA_WEB and METADATA_WEB_NAVSTRUCT flags are set and the Site has no cached Scope information.
- List Metadata Result Set, which returns Metadata associated with the Document List.
- NULL Unique Permissions Result Set, which is returned because the METADATA_PREFETCH_SCOPES flag is set, the METADATA_LISTMETADATA_NOFETCH flag has not been set, and Permissions for the Document List do not exist.
- Event Receivers Result Set, which returns information about the Event Receivers defined for the Document List.
- List Web Parts Result Set, which returns information about the List Web Parts defined for the Document List.

6. The WFE builds a Dynamic SQL query to retrieve information from the Sec_SiteGroupsView view. This query is sent to the SQL server using TDS.
7. The BEDS returns a Dynamic SQL Result Set containing information about the Team Site Owners, the Team Site Visitors, and the Team Site Members.
8. The WFE builds a Dynamic SQL query regarding which permissions the requesting User has within the Site Group. It does this by calling the proc_SecGetUserPermissionOnGroup Stored Procedure using TDS.
9. The BEDS returns a Dynamic SQL Result Set consisting of the output variables of the Stored Procedure.
10. The WFE then retrieves information about the User to be added by calling the proc_SecResolvePrincipal Stored Procedure using TDS.
11. The BEDS returns the Principal Information Result Set, consisting of a single row of information about the User to be added.
12. When the information about the User to be added has been confirmed, the User's name in the text area is underlined.
13. The User clicks the "OK" button.
14. The WFE negotiates authentication with the DC and then sends an LDAP search request to the DC for an object that has a SID attribute equal to the value obtained from the DC earlier.
15. The DC sends an LDAP Search Result containing the attributes of the Active Directory User object.

16. The WFE again negotiates authentication with the DC, and then sends an LDAP search request to the DC for an object that has a SID attribute equal to the value obtained from the DC in Step 3.

17. The DC sends an LDAP Search Result containing the attributes of the Active Directory User object.

18. The WFE then collects Metadata for the Site and the Document List by calling the proc_GetTpWebMetaDataAndListMetaData Stored Procedure using TDS.

19. The BEDS returns 14 Result Sets:

    - Web Url Result Set, which returns the Store-Relative Form URL of the root of the Site.
    - Domain Group Cache Versions Result Set, which returns information about the version numbers associated with the Domain Group map cache for the Site.
    - Domain Group Cache WFE Update Result Set, which returns the binary data needed to refresh the Domain Group map cache.
    - Site Metadata Result Set, which returns specialized Site Metadata.
    - Event Receivers Result Set, which returns information about the Event Receivers defined for the Site.
    - Site Category Result Set, which returns the Categories of the Site.
    - Site MetaInfo Result Set, which returns the specialized Site Metadata.
    - Site Feature List Result Set, which returns a list of Feature identifiers for the Site Collection that this contains Site.
    - Site Feature List Result Set, which returns a list of Feature identifiers for the Site.
    - Empty Result Set, which is returned because the METADATA_WEB and METADATA_WEB_NAVSTRUCT flags are set, and the Site has no cached Scope information.
    - List Metadata Result Set, which returns Metadata associated with the Document List.
    - NULL Unique Permissions Result Set, which is returned because the METADATA_PREFETCH_SCOPES flag is set, the METADATA_LISTMETADATA_NOFETCH flag has not been set, and Permissions for the Document List do not exist.
    - Event Receivers Result Set, which returns information about the Event Receivers defined for the Document List.
    - List Web Parts Result Set, which returns information about the List Web Parts defined for the Document List.

20. The WFE then retrieves information about the User to be added by calling the proc_SecResolvePrincipal Stored Procedure using TDS.

21. The BEDS returns the Principal Information Result Set, consisting of a single row of information about the User to be added.

22. The WFE builds a Dynamic SQL query to retrieve information from the Sec_SiteGroupsView view. This query is sent to the SQL server using TDS.

23. The BEDS returns a Dynamic SQL Result Set containing information about the Team Site Owners, the Team Site Visitors, and the Team Site Members.

24. The WFE builds a Dynamic SQL query regarding what permissions the requesting User has within the Site Group. It does this by calling the proc_SecGetUserPermissionOnGroup Stored Procedure using TDS.

25. The BEDS returns a Dynamic SQL Result Set consisting of the output variables of the proc_SecGetUserPermissionOnGroup Stored Procedure.

26. The WFE then retrieves the database and URL mapping information for the Site Collection. It does this by calling the proc_getSiteMapById Stored Procedure from the WSS Configuration Database using TDS.

27. The BEDS returns the Site Map By Id Result Set.

28. The WFE then retrieves the mapping of Fields in the Document List by calling the proc_GetListFields Stored Procedure using TDS.

29. The BEDS returns the Fields Information Result Set, consisting of a single row containing a single column of a WSS implementation-specific version string followed by an XML representation of the field definitions.

30. The WFE builds a Transactional Dynamic SQL Query to add an entry for the new User to the List of User information stored in the BEDS. This query is sent to the SQL server using TDS. On the SQL server, the following actions occur:

    - The query begins a new SQL transaction.
    - The query attempts to add the new User to the List of User information in the BEDS by calling the proc_SecAddUser Stored Procedure using TDS.
    - The query rolls back the SQL transaction if the proc_SecAddUser Stored Procedure's Return Code is not "0", or it checks to see if the new User's UserId exists in the UserData table.
    - If the User's UserId is not found in the UserData table, the query attempts to add the List Item to the Document List by calling the proc_AddListItem Stored Procedure using TDS.
    - The query rolls back the SQL transaction if the proc_AddListItem Stored Procedure's Return Code is not "0", or it commits the transaction if the proc_SecAddUser's Return Code of is "0" and proc_AddListItem's Return Code (if it runs) is "0".

31. The BEDS returns a Dynamic SQL Result Set that consists of information about the new User.

32. The WFE then attempts to add the new User to the Site Group. It does this by calling the proc_SecAddUserToSiteGroup Stored Procedure using TDS.

33. The BEDS responds with a Return Code, but no Result Sets are returned.

34. The WFE then collects Metadata for the Site and the Document List by calling the proc_GetTpWebMetaDataAndListMetaData Stored Procedure using TDS.

35. The BEDS returns 14 Result Sets:

    - Web Url Result Set, which returns the Store-Relative Form URL of the root of the Site.
    - Domain Group Cache Versions Result Set, which returns information about the version numbers associated with the Domain Group map cache for the Site.

- Domain Group Cache WFE Update Result Set, which returns the binary data needed to refresh the Domain Group map cache.
- Site Metadata Result Set, which returns specialized Site Metadata.
- Event Receivers Result Set, which returns information about the Event Receivers defined for the Site.
- Site Category Result Set, which returns the Categories of the Site.
- Site MetaInfo Result Set, which returns the specialized Site Metadata.
- Site Feature List Result Set, which returns a list of Feature identifiers for the Site Collection that this contains Site.
- Site Feature List Result Set, which returns a list of Feature identifiers for the Site.
- Empty Result Set, which is returned because the METADATA_WEB and METADATA_WEB_NAVSTRUCT flags are set and the Site has no cached Scope information.
- List Metadata Result Set, which returns Metadata associated with the Document List.
- NULL Unique Permissions Result Set, which is returned because the METADATA_PREFETCH_SCOPES flag is set, the METADATA_LISTMETADATA_NOFETCH flag has not been set, and Permissions for the Document List do not exist.
- Event Receivers Result Set, which returns information about the Event Receivers defined for the Document List.
- List Web Parts Result Set, which returns information about the List Web Parts defined for the Document List.

36. The WFE builds a Dynamic SQL Query to retrieve information from the Sec_SiteGroupsView view. This query is sent to the SQL server using TDS.
37. The BEDS returns a Dynamic SQL Result Set containing information about the Team Site Owners, the Team Site Visitors, and the Team Site Members.
38. The WFE builds a Dynamic SQL Query to retrieve information about the Site and Document List as it relates to the requesting User. This query is sent to the SQL server using TDS.
39. The BEDS returns a Dynamic SQL Result Set containing information about the current Site and Document List.
40. The WFE builds a Dynamic SQL Query regarding what permissions the requesting User has within the Site Group. It does this by calling the proc_SecGetUserPermissionOnGroup Stored Procedure using TDS.
41. The BEDS returns a Dynamic SQL Result Set consisting of the output variables of the Stored Procedure.
42. The WFE builds a Dynamic SQL Query to retrieve information about the Site and Document List as it relates to the newly added User. This query is sent to the SQL server using TDS.
43. The BEDS returns a Dynamic SQL Result Set containing information about the current Site and Document List.

44.The WFE redirects the UEC to the "http://<YourSharePointServer>/_layouts/people.aspx?MembershipGroupId=5" page.

3.4 Example 4: Create a SharePoint Document Library File from the Client Console

This example describes a simple method for creating a file using the protocols covered in this system. This example uses the Creating a SharePoint Document Library File from the Client Console use case described in section 2.5.1.

Note The following steps and diagram consolidate multiple WFE to BEDS actions, and multiple WFE to Active Directory actions into single flows. The step descriptions indicate where multiple actions are occurring and specify examples that provide more detail about those actions. In addition, the diagram and steps do not describe some initial interactions between the client and server that optionally happen on some clients, and which can also depend on whether the client has connected to the site previously to verify that the server is able to support WebDAV.

The main member protocols used in this sequence are [MS-WSSFO2] covering the stored procedures listed in the steps and [MS-WDV].

The following assumptions are made for the purposes of this example:

- The user has read/write access permissions to an existing SharePoint Foundation 2010 Document Library called "http://server/site/doclib".
- The user is logged on to a client computer running Windows® 7 operating system with an authenticated Windows session, and can access the WSS site containing the Document Library.
- From a command prompt window on the End User Client machine, the user types the following command:

```
echo hello >\\server\site\doclib\hello.txt
```

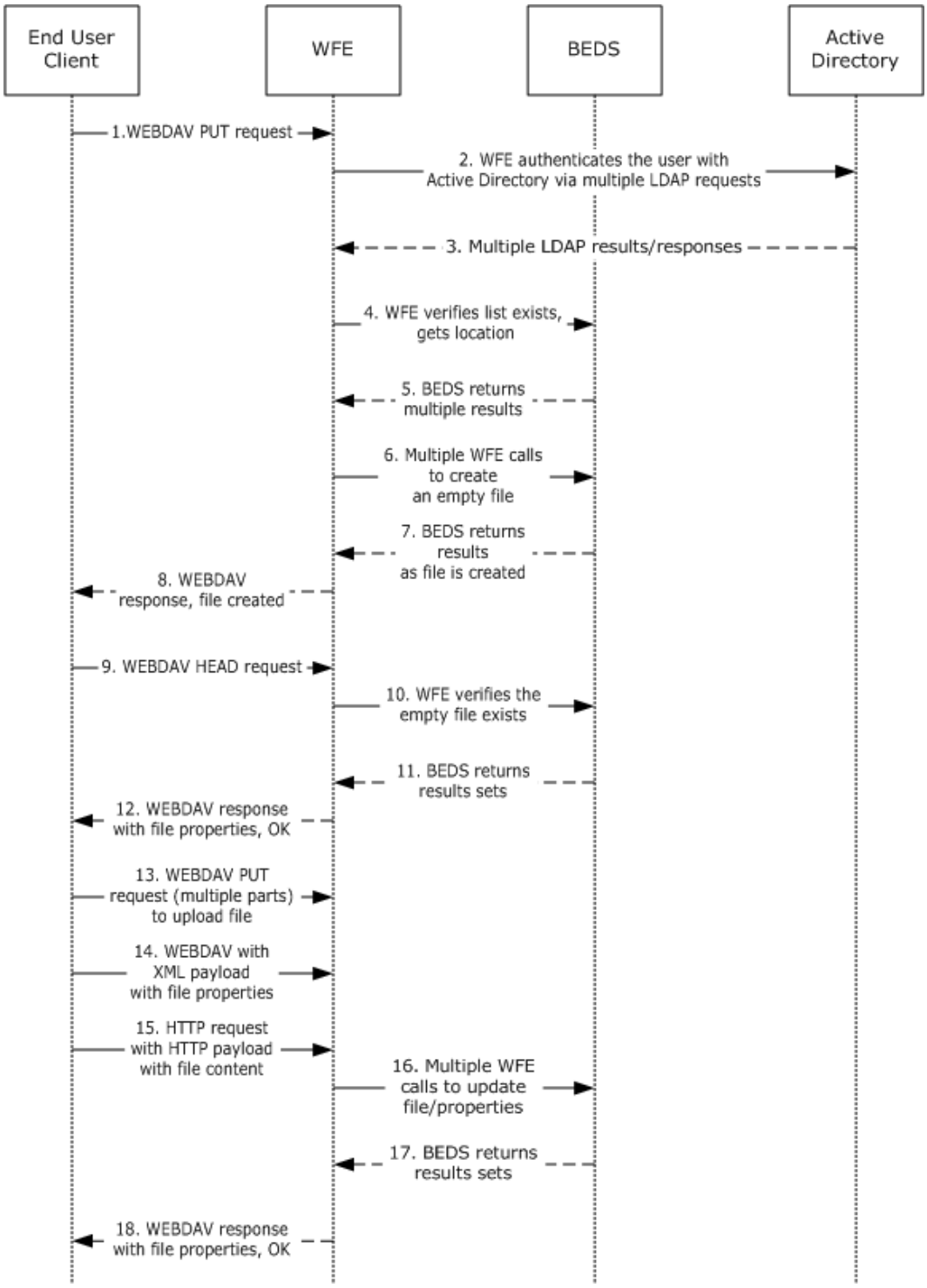

Figure 14: Create a SharePoint Document Library file from the client console

The [MS-WSSFO2] examples referenced for more details in the following steps use the SharePoint programming API; the actual steps may vary when the request is generated by user interaction with the front-end Web server, as in this case.

The following actions happen:

1. The user initiates the "echo" command and the client sends a WebDAV request to the server asking it to perform a PUT operation on the hello.txt file in the document library.

2. The front-end Web server (IIS) authenticates the user with Active Directory. In practice this may involve multiple LDAP requests with the Active Directory, especially if the user has not previously visited the site.

3. Active Directory responds with multiple LDAP results.

   For more information about Authentication, see section 2.9.2.1.

   For more information about the scenario when the user has not previously visited the site, see [MS-WSSFO2] section 4.2 for Microsoft® SharePoint® Foundation 2010.

4. In multiple roundtrips with the BEDS, the WFE locates the content database for the document library, and verifies that the library exists.

5. The BEDS server returns multiple objects for the site collection, Web site and library to the WFE.

   For more information on steps 4 and 5, see [MS-WSSFO2] section 4.6 for SharePoint Foundation 2010.

6. In multiple roundtrips with the BEDS, the WFE creates an empty file in the document library, and then, if successful, the WFE also verifies that the user has permissions to access and write to the document library.

7. The BEDS server returns multiple results sets as part of the process to create the file.

   For more information about file creation in steps 6 and 7, see [MS-WSSFO2] section 4.9 for SharePoint Foundation 2010.

8. The WFE returns a WebDAV response saying the file was created successfully.

9. The client sends a WebDAV HEAD request to WFE with the URL to the hello.txt file in the Document Library, to verify the success of the previous call.

10. In multiple roundtrips with the BEDS, the WFE retrieves the file.

11. The BEDS server returns multiple results sets as part of the process to retrieve the file.

    For more information about file retrieval in steps 10 and 11, see [MS-WSSFO2] section 4.1 for SharePoint Foundation 2010.

12. In response to the HEAD request, the WFE sends a response reporting that the request was successful.

13. Then the client sends a WebDAV PUT request to the WFE containing multiple parts, to upload the file and to update the file properties.

14. The client sends a WebDAV request to the WFE with an XML payload that has the file properties from the client.

15. The client sends a HTTP request to the WFE with an HTTP payload that has the file content; in this example that content is simply the text "hello".

16. In multiple roundtrips with the BEDS, the WFE updates the file and its properties in the document library.

17. The BEDS server returns multiple results sets as part of the process to update the files.

    For more information about file retrieval and update in Steps 16 and 17, see [MS-WSSFO2] sections 4.1 and 4.9 for SharePoint Foundation 2010. There is overlap with previous steps because stored procedures such as proc_FetchDocForUpdate ([MS-WSSFO2] section 3.1.5.21) are part of file updates as well as file creation.

18. Upon completing the update, the WFE sends a WebDAV Response reporting that the request was successful.

4 Microsoft Implementations

The information in this specification is applicable to the following product versions. References to product versions include released service packs.

- Windows Server® 2003 operating system
- Windows Server® 2008 operating system
- Windows Server® 2008 R2 operating system

Exceptions, if any, are noted in the following section.

4.1 Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Windows Server® 2003 operating system
- Windows Server® 2008 operating system
- Windows Server® 2008 R2 operating system

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

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

<1> Section 2.5.1: Operating system versions other than Windows 7 might require different steps than those specified in this use case.

5 Change Tracking

This section identifies changes that were made to the [MS-WSSO] protocol document between the May 2011 and June 2011 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- An extensive rewrite, addition, or deletion of major portions of content.
- The removal of a document from the documentation set.
- Changes made for template compliance.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the language and formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.

- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated.**

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact protocol@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
1.2 References	Added explanatory statement regarding the removal of the publishing year from Microsoft Open Specification document references.	N	Content updated.

6 Index

V