[MS-WSSO]:
Windows SharePoint Services Overview

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.

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.
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.

Date	Revision History	Revision Class	Comments
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.
09/23/2011	7.3	Minor	Clarified the meaning of the technical content.
12/16/2011	7.4	Minor	Clarified the meaning of the technical content.
03/30/2012	7.4	No change	No changes to the meaning, language, or formatting of the technical content.
07/12/2012	7.5	Minor	Clarified the meaning of the technical content.
10/25/2012	7.5	No change	No changes to the meaning, language, or formatting of the technical content.
01/31/2013	7.5	No change	No changes to the meaning, language, or formatting of the technical content.
08/08/2013	7.5	No change	No changes to the meaning, language, or formatting of the technical content.
11/14/2013	7.5	No change	No changes to the meaning, language, or formatting of the technical content.
04/30/2014	7.5	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction 6
1.1 Glossary 7
1.2 References 9

2 Functional Architecture 11
2.1 Overview 11
2.1.1 Scale-out Technologies 12
2.1.2 Storage Architecture 12
2.1.2.1 Non-File System Objects 13
2.1.2.1.1 Farm 14
2.1.2.1.2 Web Application 14
2.1.2.1.3 Site Collection 14
2.1.2.1.4 Site 14
2.1.2.1.5 List 14
2.1.2.1.6 List Item 14
2.1.2.2 File System Objects 14
2.1.2.2.1 Document Library 15
2.1.2.2.2 Folder 15
2.1.2.2.3 Document 15
2.1.2.3 Advanced Storage Concepts 15
2.1.2.3.1 Attachment 15
2.1.2.3.2 Thickets 15
2.1.2.3.3 Ghosting 15
2.1.2.3.4 Versioning 15
2.1.2.3.5 Publishing 16
2.1.2.3.6 Document Property Promotion 16
2.1.2.3.7 Large File Access 16
2.1.2.3.8 BLOB Storage Outside the Content Database 16
2.1.2.4 SQL Databases 16
2.1.2.5 Content Databases 17
2.1.2.6 Configuration Database 17
2.1.2.6.1 Site Map 17
2.1.2.6.1.1 Site Collection Lookup 18
2.2 Protocol Summary 19
2.3 Environment 20
2.3.1 Dependencies on this System 21
2.3.2 Dependencies on Other Systems/Components 21
2.3.2.1 Domain Controller/Directory Service 21
2.4 Assumptions and Preconditions 21
2.5 Use Cases 22
2.5.1 Creating a SharePoint Document Library File from the Client Console 22
2.6 Versioning, Capability Negotiation, and Extensibility 24
2.7 Error Handling 25
2.8 Coherency Requirements 25
2.9 Security 25
2.9.1 Authorization for User and Group Administration 25
2.9.1.1 Individual User Permissions (Rights) 26
2.9.1.2 Permission Level (Role) 26
2.9.1.3 User 27
2.9.1.4 Group 27

2.9.1.5 Site Group 27
2.9.1.6 Securable Object 28
2.9.1.7 Scope 28
2.9.1.8 Inheritance 28
2.9.1.9 Anonymous 29
2.9.1.10 Anonymous Rights Mask (Anonymous Permissions Mask) 30
2.9.1.11 System Account 30
2.9.2 Authentication 30
2.9.2.1 Authentication of the Requests from the End-User Client 31
2.9.2.2 Authentication of the Process Account from the Front-End Web Server 32
2.9.2.3 Creating a Site Collection Local Record of the User 32
2.9.2.4 Updating the Site Collection Local User Record (Account Migration) 32
2.9.2.5 Selecting Users and Groups from Active Directory 33
2.9.2.6 Creating an Active Directory User Account 34
2.10 Additional Considerations 34

3 Examples 35
3.1 Example 1: Active Directory: Account Creation New UI 35
3.2 Example 2: Active Directory: People Picker Browse Display UI 44
3.3 Example 3: Active Directory: People Picker Check Name UI 52
3.4 Example 4: Create a SharePoint Document Library File from the Client Console 60

4 Microsoft Implementations 66
4.1 Product Behavior 66

5 Change Tracking 67

6 Index 68

1 Introduction

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

- A ready-to-use, team-oriented **website (2)** for collaboration.
- A development platform for building web-based experiences that take advantage of the collaboration features of Windows SharePoint Services.
- A framework for deploying and managing the Windows SharePoint Services Team Site and applications built on the Windows SharePoint Services platform.

As part of the collaboration services, Windows SharePoint Services 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 Windows SharePoint Services infrastructure is designed to ensure that Windows SharePoint Services sites (2) 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 Windows SharePoint Services. In order to deliver these file services capabilities, Windows SharePoint Services uses three major sets of protocols:

- File-oriented communication between the **end-user client** and the Windows SharePoint Services **front-end web server** using the **Web Distributed Authoring and Versioning Protocol (WebDAV)** as described in [RFC2518], [MS-WDV], and [MS-WDVSE]. The end-user client 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.
- Web pages presented to the client using standard **Hypertext Transfer Protocol (HTTP)**.
- Communication between the front-end web server and the Windows SharePoint Services **back-end database server** 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 front-end web server and back-end database server is described in [MS-WSSFOB] for Windows SharePoint Services 2.0, [MS-WSSFO] for Windows SharePoint Services 3.0, and [MS-WSSFO2] for Microsoft SharePoint Foundation 2010.

This document provides an overview for protocols for Windows SharePoint Services 2.0, Windows SharePoint Services 3.0, and SharePoint Foundation 2010. It generally refers to Windows SharePoint Services when the subject applies to all versions, and explicitly calls out the version when necessary for clarity.

Note This document will not be updated to reflect new releases. For more information, see the SharePoint Products and Technologies Protocols Overview [MS-SPO]. This document provides an informative overview of the front- and back-end protocols that are implemented by all SharePoint Products and Technologies for communicating with client and server applications. Unlike this overview, this document is not limited only to those protocols that provide file, print, and user/group administration capabilities. In addition, [MS-SPO] covers Windows SharePoint Services 3.0 and Microsoft Office SharePoint Server 2007 as well as SharePoint Foundation 2010 and Microsoft SharePoint Server 2010. Going forward, this document will also cover future versions of SharePoint Foundation 2010 and SharePoint Server 2010.

Note The **Transact-Structured Query Language (T-SQL)** based protocols change significantly in their function between Windows SharePoint Services 2.0, Windows SharePoint Services 3.0, and SharePoint Foundation 2010. Separate versions of the specification documents for these protocols target each release.

- Windows SharePoint Services 2.0: [MS-WSSFOB] Windows SharePoint Services (Windows SharePoint Services): File Operations Database Communications Base Protocol Specification.
- Windows SharePoint Services 3.0: [MS-WSSFO] Windows SharePoint Services (Windows SharePoint Services): File Operations Database Communications Protocol Specification
- SharePoint Foundation 2010: [MS-WSSFO2] Windows SharePoint Services (Windows SharePoint Services): File Operations Database Communications Version 2 Protocol Specification

1.1 Glossary

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

access control entry (ACE)
access control list (ACL)
Active Directory
anonymous authentication
anonymous user
authentication
authorization
digital certificate
directory service (DS)
domain
domain controller (DC)
domain user
file system
forest
group object
Hypertext Transfer Protocol (HTTP)
Kerberos
LDAP
Lightweight Directory Access Protocol (LDAP)
NT LAN Manager (NTLM) Authentication Protocol
security identifier (SID)
security principal
topology
user object
XML

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

Active Directory account creation mode
attachment
back-end database server
basic authentication scheme
binary large object (BLOB)
Central Administration site
column
configuration database
connection string
content database

deployment
display name
document
document library
domain group
email address
event receiver
farm
feature identifier
field
file
folder
forms authentication
front-end web server
group
hierarchy
Integrated Windows authentication
list
list item
list view
login name
major version
member
membership
parent site
path component
permission
permission level
placeholder
pluggable security authentication
query
result set
return code
rights
role
role definition
search provider
search query
securable object
security group
security group identifier
security scope
server-relative URL
Session Initiation Protocol (SIP)
Simple Mail Transfer Protocol (SMTP)
site
site collection
site collection administrator
site collection identifier
SQL authentication
stored procedure
store-relative URL
thicket
thumbnail
transaction

Transact-Structured Query Language (T-SQL)
trusted subsystem
Uniform Resource Locator (URL)
user identifier
user name
web application
web application identifier
Web Distributed Authoring and Versioning Protocol (WebDAV)
Web Part
web server
website

The following terms are specific to this document:

anonymous access: A mechanism that does not require users to specify a user name or password for authentication (1).

end-user client: 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.

1.2 References

References to Microsoft Open Specifications documentation 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.

We conduct frequent surveys of the informative references to assure their continued availability. If you have any issue with finding an informative 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".

[MSDN-SHPTSDK] Microsoft Corporation, "Windows SharePoint Services 3.0 SDK", December 2007, http://msdn.microsoft.com/en-us/library/ms441339.aspx

[MSDN-SHPTSDK4] Microsoft Corporation, "Microsoft SharePoint 2010 SDK", http://msdn.microsoft.com/en-us/library/office/ee557253(v=office.14).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", SharePoint Services 3.0 SDK, http://msdn.microsoft.com/en-us/library/bb802976.aspx

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

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

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

[MS-SPO] Microsoft Corporation, "SharePoint Products and Technologies Protocols Overview".

[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".

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

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

[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

2 Functional Architecture

Windows SharePoint Services provides team-oriented collaboration websites (2) , a platform for building web-based applications that use the Windows SharePoint Services 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

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

- The end-user client is a computer on which an individual user is requesting specific **file** and user operations. These requests are communicated using Hypertext Transfer Protocol (HTTP)-based protocols, including HTTP, WebDAV, and Microsoft FrontPage Server Extensions.
- The front-end web server is a computer that receives requests from an end-user client and provides Windows SharePoint Services capabilities by manipulating information stored in a database. These database requests are expressed in the T-SQL language and communicated using the Tabular Data Stream (TDS) protocol, as specified in [MS-TDS].
- The back-end database server is a computer that runs Microsoft SQL Server and responds to requests from the front-end web server.
- An **Active Directory** server responds to authentication requests from the end-user client, front-end web server, and back-end database server. These components could use an alternate authentication mechanism besides Active Directory.

These systems are shown in the following diagram.

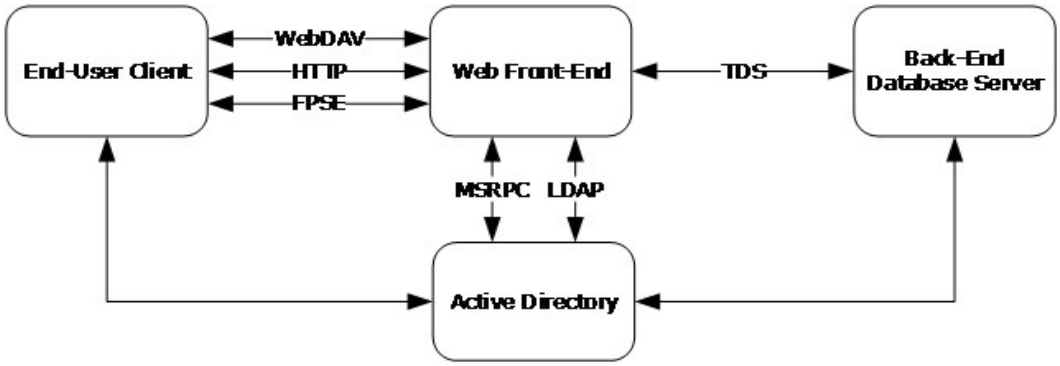

Figure 1: Intersystem protocol relationships

The front-end web server and back-end database server are considered part of the Windows SharePoint Services deployment; the end-user client is normally a user's desktop or laptop computer that connects to Windows SharePoint Services, and Active Directory is part of the generally available infrastructure. Both front-end web server and back-end database server can run on a single computer, so the simplest Windows SharePoint Services deployment could be just a

single server. Alternatively, multiple instances of front-end web server and back-end database server computers can be installed for greater throughput and redundancy.

The following sections describe the architectural concepts pertaining to how Windows SharePoint Services uses the protocols specified by the member protocol specifications identified in section 2.2. These sections describe:

- How Windows SharePoint Services deployments can be scaled out with multiple computers, called a SharePoint **farm**. This **scale-out** is transparent to individual front-end web server computers as they respond to individual requests from an end-user client.
- The Windows SharePoint Services storage model, which allows for a variety of data management and organization techniques:
  - Storage for non-**file system** objects, such as **sites (2)**, **site collections**, **lists (1)**, 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 Windows SharePoint Services deployment.

2.1.1 Scale-out Technologies

When a Windows SharePoint Services **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 the** front-end web server**:** Windows SharePoint Services supports network load-balancing technologies that distribute client requests across multiple servers in a farm. These individual front-end web servers are stateless; any front-end web server in the farm is prepared to handle any client request in the same way as any other front-end web server in the farm. By eliminating state or session information about the front-end web server, overall operational throughput can be increased simply by adding more front-end web servers. This stateless operation also makes the end-user experience more robust, because if a front-end web server fails, another front-end web server in the farm can handle future requests from the user. Front-end web servers 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 front-end web server, that front-end web server will fetch the site (2) 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 (2) into separate content databases.

The network load-balancing technology creates the need for each front-end web server in the server farm to behave identically to all other front-end web servers. 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

Windows SharePoint Services 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 Windows SharePoint Services, and also form the core **securable object**.

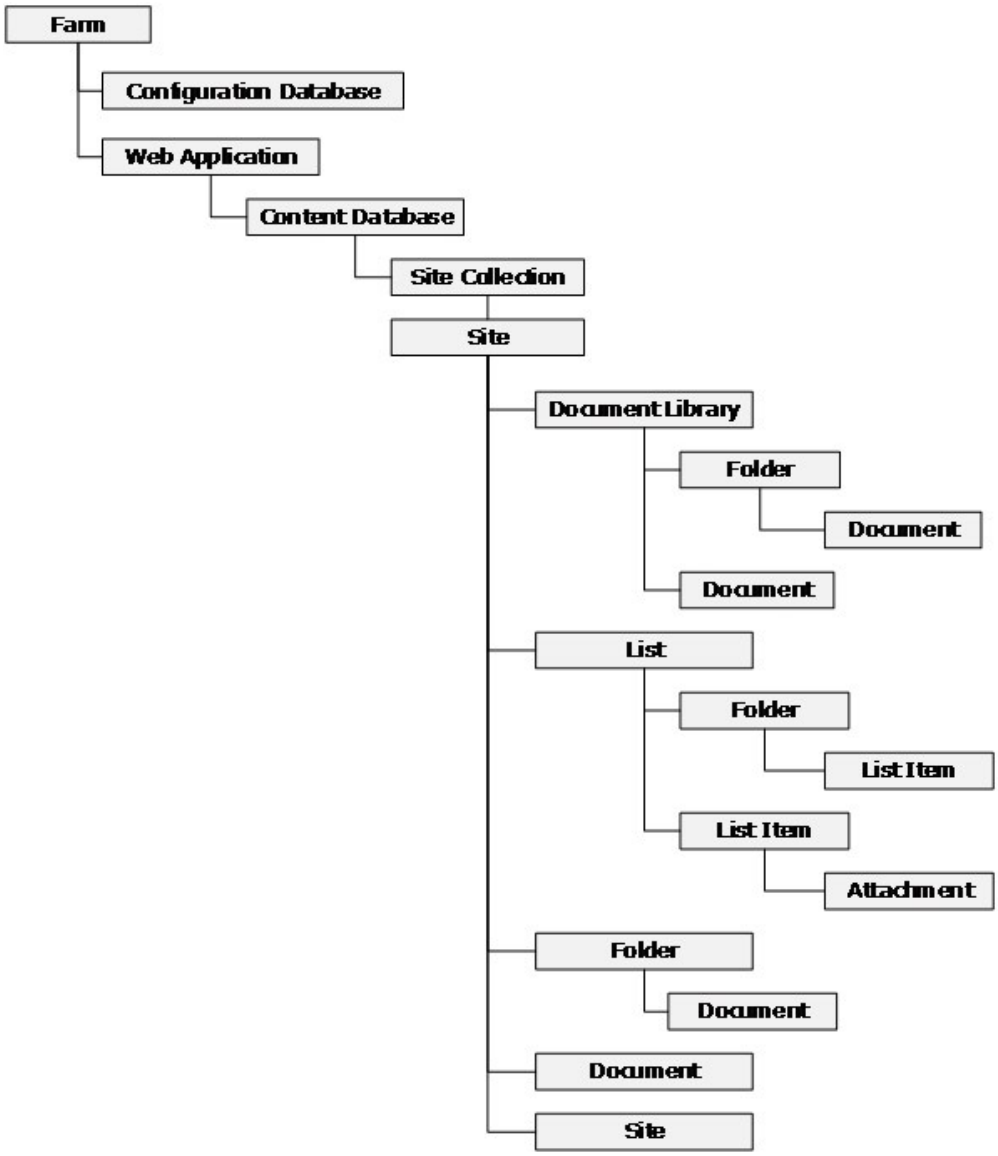

Figure 2: SharePoint storage object hierarchy

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 Windows SharePoint Services 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 **Uniform Resource Locator (URL)**, which can be used to directly reference that object.

2.1.2.1.2 Web Application

The **web application (1)** is the container for grouping site collections within the context of the **web server**. A web application (1) 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 website (2) within Windows SharePoint Services 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 Windows SharePoint Services.
- 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 for defining site groups (section 2.9.1.5) that can be given access to specific resources inside Windows SharePoint Services.

2.1.2.1.4 Site

A site (2) is a container within a site collection that allows for delegated administration where certain actions, such as managing **permissions** to content within the site (2), can be delegated by the **site collection administrator** to the site (2) 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 (2) at the root, but can have many nested sites (2) under the root site (2).

2.1.2.1.5 List

A list (1) is a location within a site (2) that maintains a configurable data structure where end-user data can be stored. A list (1) is composed of multiple columns that define the structure for the data stored within the list (1).

2.1.2.1.6 List Item

A **list item** is a unique row within a list (1) where individual end-user data elements are stored. A list item follows the structure of **columns (1)** as defined by the list (1) that contains the list item.

2.1.2.2 File System Objects

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

2.1.2.2.1 Document Library

A **document library** is a type of list (1) specifically designed to be a location within a site (2) where end-user **documents** are stored.

2.1.2.2.2 Folder

A folder is an organizational tool used within a site (2) 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 (2), document library, folder, or as an attachment.

2.1.2.3 Advanced Storage Concepts

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

2.1.2.3.1 Attachment

Many list (1) structures within Windows SharePoint Services 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, Windows SharePoint Services 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 (2) and lists (1) tend to share a relatively small set of common system files across many instances within the back-end database server. To avoid having to store these files repeatedly for every site (2) or list in the back-end database server, Windows SharePoint Services allows files to be ghosted, meaning that the content of the file is not actually stored in the back-end database server. Instead, a reference to a location on the front-end web server where the source of the file can be found is stored in the file metadata.

2.1.2.3.4 Versioning

Windows SharePoint Services can be configured on a per-list (1) 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

Windows SharePoint Services can be configured on a per-list (1) 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 (1) 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 (1) 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, Windows SharePoint Services 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 back-end database server provides functionality to return files larger than a specified size to the front-end web server in a series of smaller chunks that can be processed more smoothly.

2.1.2.3.8 BLOB Storage Outside the Content Database

Windows SharePoint Services provides the ability to allow file metadata to be stored by Windows SharePoint Services 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 Microsoft SQL Server concept of unstructured, binary data streams that are commonly associated with files.

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

1. **External BLOB Storage:** For more information about External BLOB Storage, see [MSDN-WSSEBS]. Windows SharePoint Services 3.0 and wss4 support External BLOB Storage providers.
2. **Remote BLOB Storage:** For more information about Remote BLOB Storage, see [MSDN-SQLRBS]. Microsoft 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 Windows SharePoint Services deployment. These databases are considered "internal" to Windows SharePoint Services, which does not support users, developers, or system administrators directly accessing or manipulating content in these databases. This is true regardless of whether the Windows SharePoint Services deployment is running on a single server or across multiple computers.

Instead, Windows SharePoint Services 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 (1) will have one or more content databases. The Windows SharePoint Services content database stores the data and documents that end users have associated with their SharePoint site (2). The contents of this database are fully normalized to efficiently perform the kinds of operations required for the high-scale, highly configurable system previously described. The role of a Windows SharePoint Services front-end web server 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 end-user client. The queries and stored procedures are specified in [MS-WSSFOB] for Windows SharePoint Services 2.0, [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 Windows SharePoint Services deployment, for either a single instance of Windows SharePoint Services 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 site**, while run-time web applications (1) 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.

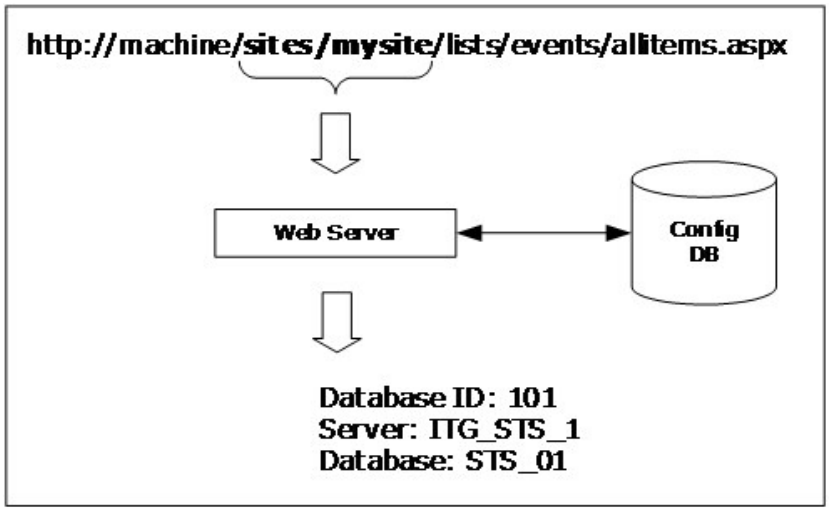

Figure 3: Determining the Site Collection URL

This mapping can use a section of the URL (in this case "sites/mysite") to map to an individual back-end database server and content database. This information is stored as a **server-relative URL**, 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, Windows SharePoint Services determines 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 (1) URLs. If one of the URLs in the list matches the incoming URL, the associated web application (1) is used for the remainder of the operation.

2. **Prefix Matching:** Web applications (1) 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 (1) 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 (1) 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 (1) 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>
http://example.com/a/b.htm	""	"/a"

1. **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 of 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 (1) 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.

2. **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:

   - 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.
   - 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.
   - The identifier of the server is passed to the configuration database, which returns the address of the server.
   - 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	This protocol specifies the communication between the front-	[MS-

Protocol name	Description	Short name
Services (WSS): File Operations Database Communications Base Protocol	end web server and the back-end database server that is used to satisfy requests involving file access and the administration of users and groups within Windows SharePoint Services 2.0.	WSSFOB]
Windows SharePoint Services (WSS): File Operations Database Communications Protocol	This protocol specifies the communication between the front-end web server and the back-end database server used to satisfy requests involving file access and administration of users and groups within Windows SharePoint Services 3.0.	[MS-WSSFO]
Windows SharePoint Services (WSS): File Operations Database Communications Version 2 Protocol	This protocol specifies the communication between the front-end web server and the back-end database server used to satisfy requests involving file access and administration of users and groups within 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 website (2) 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 website (2).	[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 in Windows Server 2003 operating system, Windows Server 2008 operating system with Service Pack 2 (SP2), or Windows Server 2008 R2 operating system 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:

- 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 with Service Pack 2 (SP2), Windows Server 2003 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, Windows SharePoint Services can communicate with an Active Directory **domain controller (DC)** to provide **authentication (1)** services that enable the User/Group Administration functions described in this document. This domain controller provides an **LDAP**-enabled **directory service (DS)** that stores user information, such as name and **email address**.

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 Windows SharePoint Services.

- The Windows SharePoint Services server(s) is reachable by external clients via an established IP address (or IP addresses).
- The Windows SharePoint Services server(s) functional components are started collectively and the Windows SharePoint Services server(s) accepts client requests.
- The Windows SharePoint Services front-end web servers can reach back-end database servers and have appropriate permissions to access data in the content database and the configuration database.
- The Windows SharePoint Services front-end web server and back-end database server 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 (1), the directory service (DS) is accessible to the Windows SharePoint Services 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 Windows SharePoint Services 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.

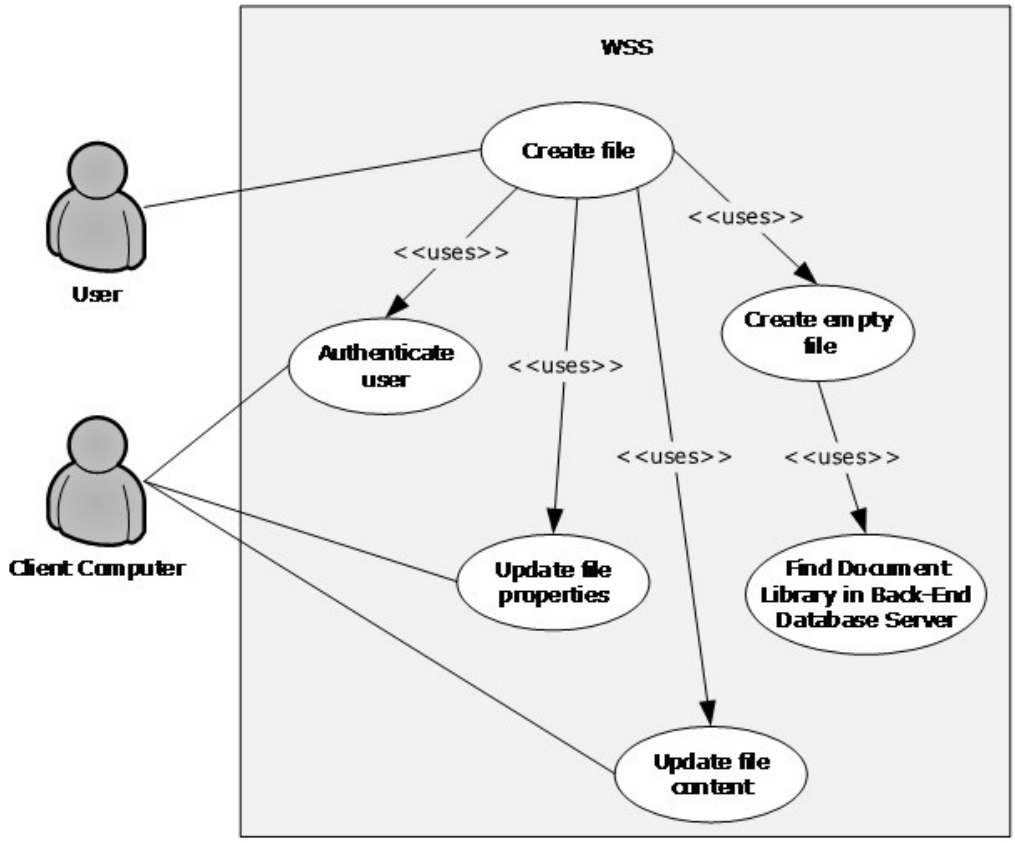

Figure 4: Create file from client computer

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 the Windows 7 operating system<1> with an authenticated Windows session, and can access the Windows SharePoint Services site (2) 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 types the `echo` command and presses **Enter**.
2. The Windows SharePoint Services front-end web server authenticates the user.
3. Windows SharePoint Services finds the location of the document library and verifies that the user has the appropriate access permissions.
4. Windows SharePoint Services 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

- The user does not have the appropriate access permissions: The client notifies the user that access is denied.
- The client cannot connect to the Windows SharePoint Services front-end web server: The client notifies the user of the connection error.
- 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 Windows SharePoint Services front-end web server and Windows SharePoint Services back-end database server perform explicit version verifications in Windows SharePoint Services 3.0 and Microsoft SharePoint Foundation 2010.

The client calls the proc_GetVersion stored procedure to retrieve version information from the server and to decide whether it ought to connect to the database. The proc_GetVersion stored procedure 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 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 2.2.7.11 for SharePoint Foundation 2010.

Windows SharePoint Services front-end web server initializes the connection to the back-end database server according to the following steps:

1. When the front-end web server was added to the SharePoint farm, the administrator gave a connection string to the configuration database.
2. In SharePoint Foundation 2010, the front-end web server 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 front-end web server 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 front-end web server 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 front-end web server verifies that it is connected to a configuration database with an appropriate version, the front-end web server 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 front-end web server has the connection string to the content database, the front-end web server 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 front-end web server disconnects from the content database.

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

5. The validation result is cached in the front-end web server process. When the process restarts, the validation will be performed again.

The acceptable range of specified version numbers can change when Windows SharePoint Services 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 Windows SharePoint Services security model: authentication (1) and **authorization**.

Authentication (1) 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. Windows SharePoint Services uses the authentication mechanism from an underlying platform, such as Internet Information Services (IIS) and Microsoft ASP.NET, to authenticate users.

Windows SharePoint Services supports all of the authentication modes that IIS and ASP.NET support, including Active Directory, **forms authentication**, and WebSSO authentication. In Active Directory authentication mode, IIS authenticates the user, using **basic authentication scheme**, **digital certificate**, **NT LAN Manager (NTLM) Authentication Protocol**, or **Kerberos**. In other authentication modes, Windows SharePoint Services 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 Windows SharePoint Services 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 Windows SharePoint Services, and the caller has been authenticated, the authorization code is called to identify whether the access can be granted. In a **trusted subsystem** model, the front-end web server uses the IIS application identity 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 (2). For more information, see section 2.9.2.2. Therefore, the permissions check has to happen before Windows SharePoint Services 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), Windows SharePoint Services controls the user's authorization and determines which permissions are granted to that user on a given object.

Windows SharePoint Services supports a number of security-related operations to control access to content stored in Windows SharePoint Services. These operations are built around a few core concepts defined in Windows SharePoint Services, 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 (1). Windows SharePoint Services has a fixed set of permissions that can be granted to users. The full specification of the Windows SharePoint Services Rights Mask is provided in the Windows SharePoint Services Rights Mask section of [MS-WSSFOB] section 2.2.2.10 for Windows SharePoint Services 2.0, [MS-WSSFO] section 2.2.2.13 for Windows SharePoint Services 3.0, and in [MS-WSSFO2] section 2.2.2.14 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 (2), web, list (1), 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 (2) level, where a site (2) can inherit roles from its **parent site** or have roles unique to it. All permissions in Windows SharePoint Services are managed through roles and all users will have roles. Rights are never directly assigned to a user. The default Windows SharePoint Services 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 (1), 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 Windows SharePoint Services. **Permission levels** can be directly assigned to users. After a user has been authenticated, that user's identity is represented by a Windows SharePoint Services 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 (2), where Windows SharePoint Services tracks which users (or groups, as described in section 2.9.1.4) are assigned to which roles for each site (2). 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 (3)**.
- The roles that the user acquires by being a member (3) of a **group (2)** or site group (section 2.9.1.5).

2.9.1.4 Group

A group (2) is an identity associated with a group of users within Active Directory (Windows **security group**). As far as account management is concerned, Windows SharePoint Services treats groups (2) similarly to user accounts. When a user interacts with a Windows SharePoint Services environment, their Active Directory group (2) **membership** is determined and their membership within a group (2) is used to determine the effective role of the user.

2.9.1.5 Site Group

A Windows SharePoint Services 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 Windows SharePoint Services site group is assigned a default role, but the role for any site group can be changed as necessary. Some predefined Windows SharePoint Services site groups are as follows:

- Site Owners
- Site Members
- Site Visitors

Windows SharePoint Services group (2) memberships are stored in SQL table named GroupMemberships. Each group is assigned an identifier that is unique within that site collection. The use of groups can enable easier security management. When a large number of users have to be assigned the same role, administrators can easily create a Windows SharePoint Services group (2) and assign those users as members (3) 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 Windows SharePoint Services groups (2) 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-WSSFOB] for Windows SharePoint Services 2.0
- 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

Windows SharePoint Services groups (2) cannot be nested inside of each other. However, a Windows SharePoint Services group (2) can contain Active Directory groups as members (3).

Windows SharePoint Services groups (2) are themselves a securable object in Windows SharePoint Services 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 (2), library, folder, or document. By default, permissions for a site (2), library, or document are inherited from the parent site or library.

Folders within lists (1) are securable objects in that they derive their permissions from the underlying list items they contain. Pages that do not belong to any list (1) are not securable objects; such pages always share the same permissions as their parent site (2). 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 can be inherited from the parent of the object.

2.9.1.7 Scope

A **security scope** represents a URL subtree in Windows SharePoint Services that shares the same permissions. A user creating an item in Windows SharePoint Services could choose to give the item its own specific permission requirements or specify that it can 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 (2) within a site collection.

2.9.1.8 Inheritance

As mentioned earlier, an item in Windows SharePoint Services can have its own specific permissions. However, default permissions for an item within a site (2) are inherited from that site (2). This inheritance can be broken for any securable object at a lower level in the site (2) **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 (2). However, the inheritance is broken only for that particular document library; the rest of the permissions for the site (2) remain unchanged. An object can be reverted to inheriting permissions from its parent list (1) or site (2) at any time.

Sites (2) are themselves a securable object to which permissions can be assigned. Sites (2) contained within other sites (2) can be configured to inherit permissions from a parent site or to create unique permissions for that particular site (2). If a child site (2) inherits permissions from its parent, that set of permissions is shared with the child site (2). This effectively means that owners of the sites (2) that inherit permissions from a parent site can change the permissions of the parent. To allow control of the permissions for the child site (2) alone, the child site (2) stops inheriting permissions, restricting the owner of the child site (2) to only making changes to the permissions of the child site (2).

Creating unique permissions for a site (2) stops permission inheritance. The groups (2), users, and permission levels from the parent site are copied to the child site (2) and then the inheritance is broken. Reverting a site (2) back from unique permissions to inherited permissions causes users, groups (2), and permission levels to once again be inherited and removes any users, groups (2), or permission levels that were uniquely defined in the site (2) 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 regardless 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 (2) 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_SecChangeToUniqueWeb in [MS-WSSFOB] for Windows SharePoint Services 2.0
- 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

Windows SharePoint Services allows a specific type of access where the user is not uniquely authenticated, and thus is unknown to Windows SharePoint Services. 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 (1) level of Windows SharePoint Services. If anonymous access is allowed for the web application (2), then for example, site (2) administrators can decide whether to:

- Grant anonymous access to a site (2)
- Grant anonymous access only to lists (1) and libraries
- Block anonymous access to a site (2) 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 Windows SharePoint Services. By default in IIS, the anonymous user account is IUSR_ComputerName. Enabling anonymous access essentially grants the anonymous account access to the Windows SharePoint Services site (2). Allowing access to a site (2), or to lists (1) 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 (2) in My Network Places

When the user who is accessing Windows SharePoint Services items is anonymous, then the user identifier is null. In authentication (1) modes other than Active Directory, such as forms authentication, Windows SharePoint Services is also impersonating IUSR_ComputerName. In those cases, Windows SharePoint Services 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 Windows SharePoint Services. Because, unlike other Windows SharePoint Services 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), Windows SharePoint Services 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 Windows SharePoint Services can trigger operations that require the ability to run with full permissions if the current user does not have permission to do so directly. For security reasons, the feature cannot 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 login name for the system account is SHAREPOINT\system. For example, when a list item is created by the application pool identity account, it will show as "created by SHAREPOINT\system".

2.9.2 Authentication

Windows SharePoint Services supports **pluggable security authentication**, an extensibility mechanism provided by ASP.NET. By default, Windows SharePoint Services uses one of three authentication (1) modes against a Windows **domain**:

- **Integrated Windows authentication**
- Basic authentication scheme
- **Anonymous authentication**

Specific deployments can use a custom authentication (1) provider to authenticate end users against any third-party authentication system.

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

- Authentication (1) of the requests from the end-user client. The front-end web server establishes a specific end-user identity for requests from the end-user client. The front-end web server 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 (1) of the Process Account from the front-end web server. The back-end database server establishes an identity for requests from the front-end web server. The back-end database

server evaluates whether that identity has permissions to operate as a Windows SharePoint Services front-end web server for content stored in the back-end database server.

- 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 Active Directory.
- Selecting users and groups (2) from the directory for the purposes of setting security access control lists (ACLs), as well as defining SharePoint groups (2).
- Creating Active Directory user accounts in **Active Directory account creation mode** to enable the creation of Active Directory accounts for Windows SharePoint Services users.

Section 2.9.2.1 specifies how Windows SharePoint Services uses the Active Directory Protocol [MS-ADTS] for the two types of authentication (1) previously described.

2.9.2.1 Authentication of the Requests from the End-User Client

A typical scenario for authentication (1) of the requests from the end-user client 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 Windows SharePoint Services site (2). When the user makes a page request to Windows SharePoint Services, IIS handles the request and authenticates the user. IIS could choose one of the three methods: Integrated Windows authentication, basic authentication scheme, 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 Windows SharePoint Services code to fulfill the request from the end-user client. The request contains the unique address of a resource in Windows SharePoint Services (a page, a document, a list item, and so on), and its associated **ACL**. The ACL specifies which security principal (3) 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 Windows SharePoint Services 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 and **security group identifier**. 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 (ACE)** in the requested object's ACL. An ACE contains the security principal (3) 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 Front-End Web Server

When the front-end web server requires data from the back-end database server, it makes a request for that data via the appropriate protocol. The back-end database server has to validate whether the front-end web server has appropriate permissions to access the data in the back-end database server.

The Windows SharePoint Services worker process in the front-end web server 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 back-end database server validates the requestor's identity using either Integrated Windows authentication or SQL login. If Integrated Windows authentication is chosen, the request made to the back-end database server is done using the service account identity.

The back-end database server 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, Windows SharePoint Services has to know the user's identity for other purposes, such as indicating who created a document, providing an email address to deliver an alert, or associating a picture with a discussion post.

To optimize back-end database server 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, email address or **Session Initiation Protocol (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 back-end database server 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, email address, SIP address, title, and department. The entry is created when one of the following actions occur:

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

All of the fields for the user entry are found in the user's Active Directory **user object** record. If the Active Directory user object record is not populated with any information, the resulting entry in the Windows SharePoint Services 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 has legally had their name changed), domain (the user was migrated from one Active Directory domain to another), or **forest (2)** (user was migrated from one Active Directory forest (2) 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 (2), or from one forest (2) to another forest (2).

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 (2), validation of SID history is recommended. When the MigrateUser command is issued, the Windows SharePoint Services front-end web server 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 (2) is the boundary for security principals (3). In this case, the new SID is looked up from the new Active Directory forest (2), 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 Windows SharePoint Services as the directory containing the list of users and groups (2) that can be used for securing a container in Windows SharePoint Services. End users as well as administrators can look up users and groups (2), search for users and groups (2), and add one or more users and/or groups (2) to the site (2).

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

- Resolve a name or ID.
- Search for all matching records for a query.

In Windows SharePoint Services, a user is able to call the user and group selector in the security setting UI. The control contains a text box, a **Check Names** button, and a **Browse** button.

The typical scenario involves a user adding a user name or user ID into the text box and pressing the **Check Names** button. The front-end web server 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 text 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 list.

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 search string and selects the **Go** button to issue the query. The search string 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 on an 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 (2). In this situation, it is impossible to invite someone who is not listed in Active Directory to view or participate in a Windows SharePoint Services site (2).

Windows SharePoint Services 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, Windows SharePoint Services is deployed in a mode called Active Directory account creation mode. This allows Windows SharePoint Services to create an account for a user when that user first attempts to interact with a Windows SharePoint Services site (2).

For example, a Team Site administrator invites a partner using an email address, username@example.com. Windows SharePoint Services sends an email to that address inviting the user to access the resources at that Team Site. When the prospective user first arrives at the Windows SharePoint Services site (2), the user is asked to create a site (2) account that identifies that user for future visits.

After the user completes the sign-up process, the Windows SharePoint Services front-end web server creates a user account in Active Directory, and stores the new user's name, email address, and other data. The location in Active Directory where these accounts are created, and the permissions to create accounts, are set up when Windows SharePoint Services 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 Examples

These examples describe in detail the process of communication between the various server components involved in the Windows SharePoint Services deployment. In conjunction with the technical protocol documents listed in section 2.2, these examples are intended to provide a comprehensive view of how Windows SharePoint Services front-end web servers communicate with end-user client, Active Directory domain controller (DC), and back-end database server systems.

3.1 Example 1: Active Directory: Account Creation New UI

This example describes the requests made when the user on an end-user client computer fills in the email address and **display name**, and then clicks the "OK" button on the Create Active Directory Account dialog 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 because of size limitations. The three figures in this section represent a single sequence. This specific example is for Active Directory operations involving Windows SharePoint Services 3.0.

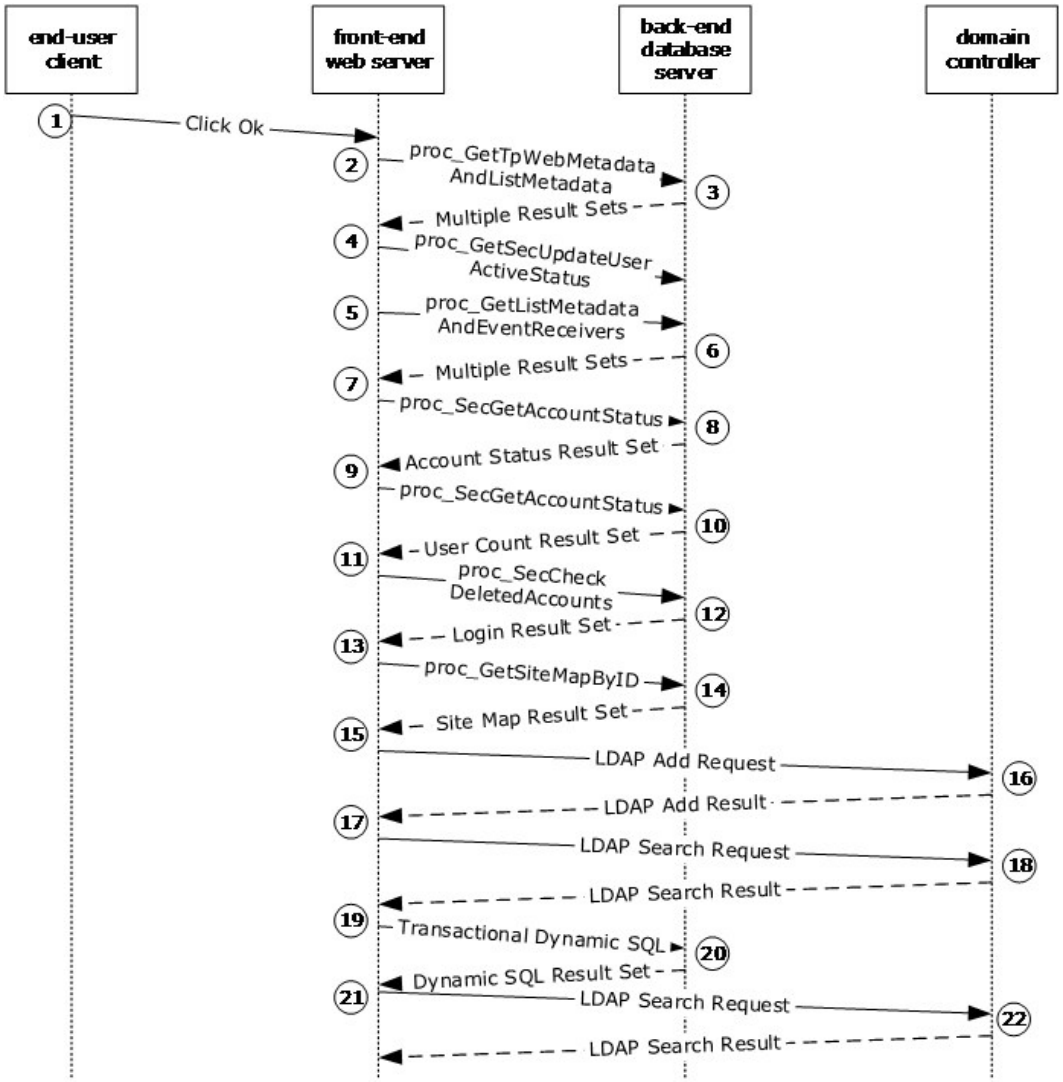

Figure 5: Account Creation New UI, steps 1 through 22

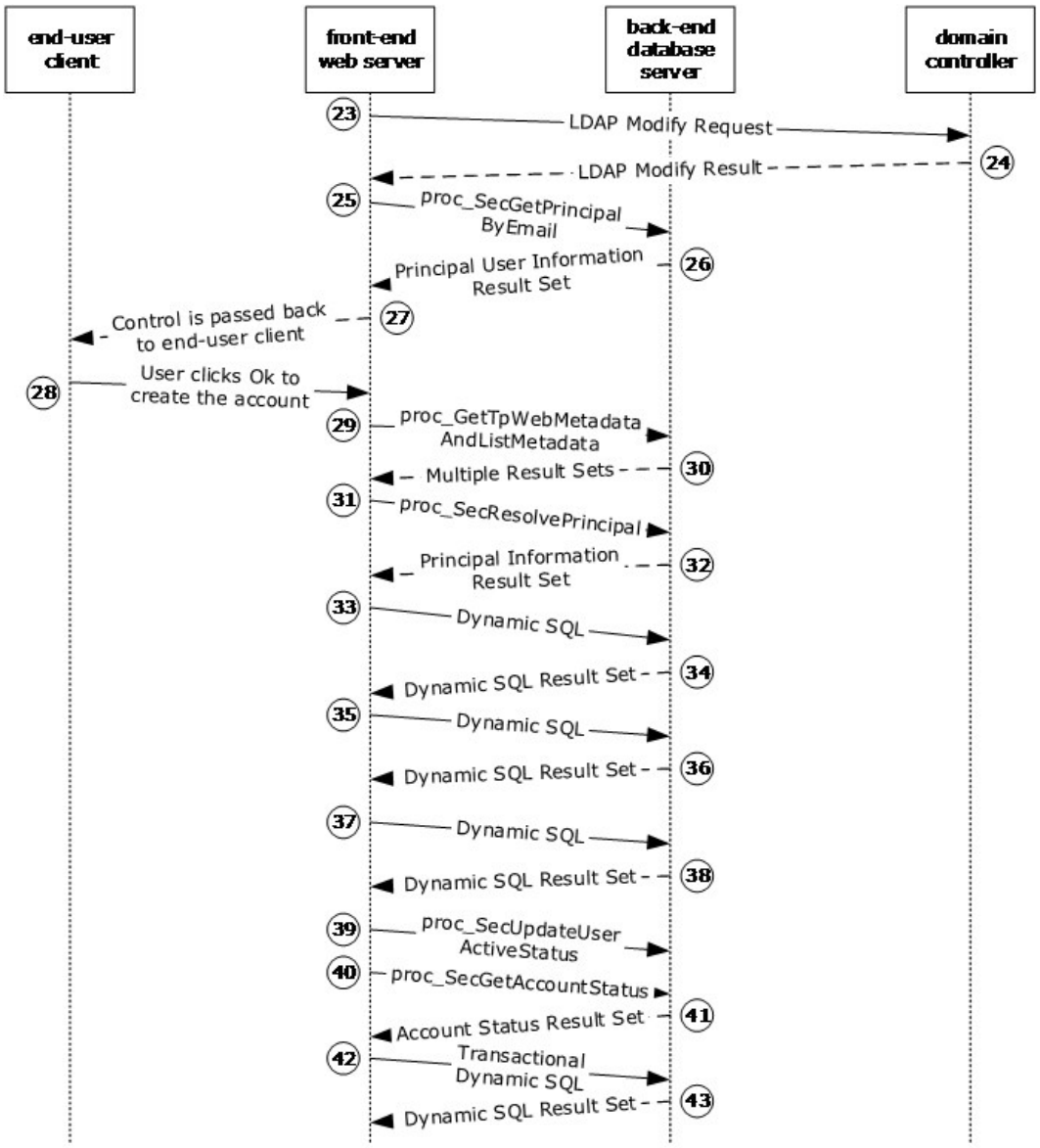

Figure 6: Account Creation New UI, steps 23 through 43

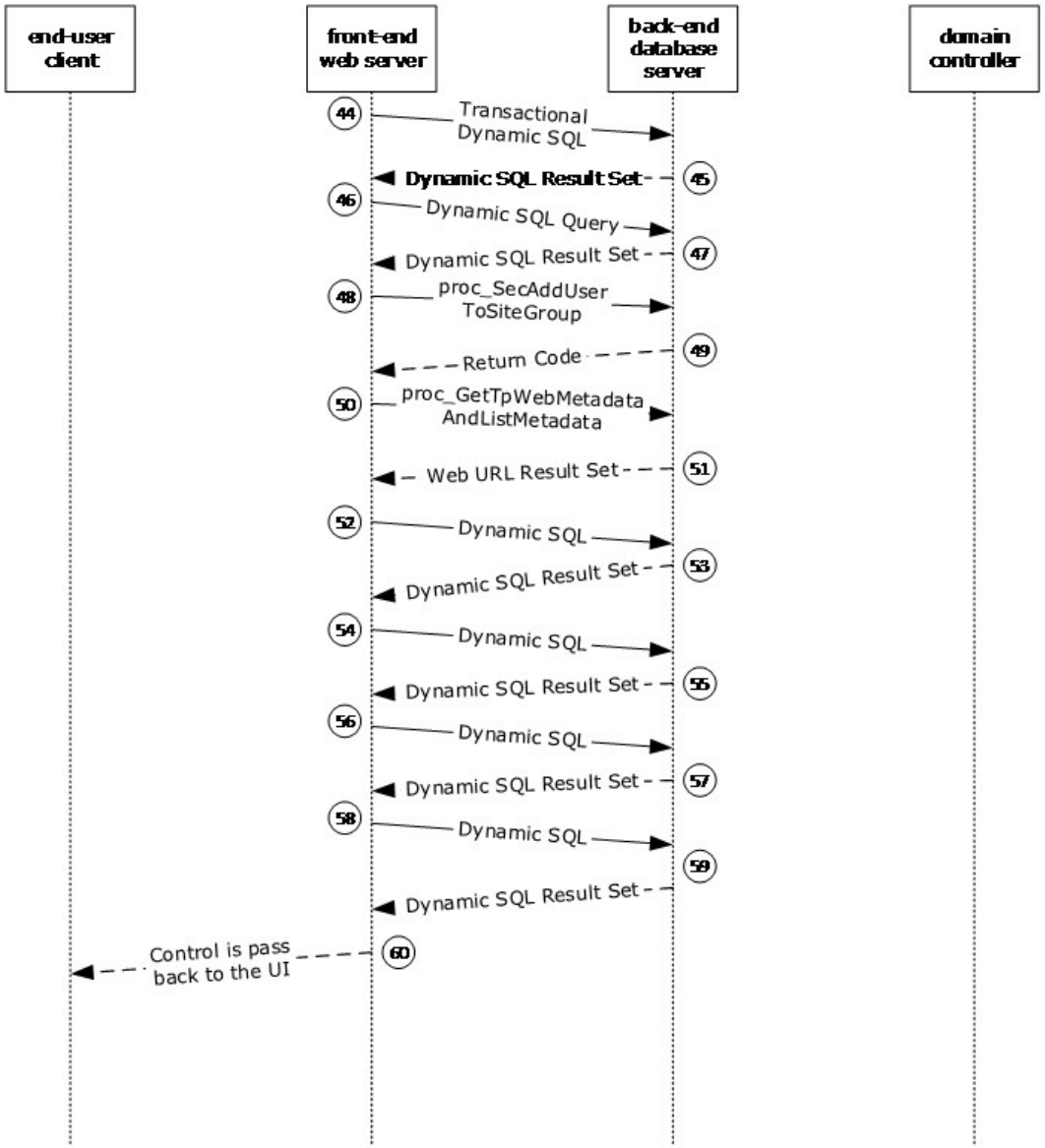

Figure 7: Account Creation New UI, steps 44 through completion

This scenario uses the Windows SharePoint Services user interface (UI) to create a new account for a user. It assumes that a user on an end-user client 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, and fills in a valid email 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 the **OK** button, which causes the end-user client to start the **UserAdd** process via a page to post back to the front-end web server to verify that the new user isn't currently defined.

2. The front-end web server first gathers information about the current environment by calling the proc_GetTpWebMetadataAndListMetadata stored procedure.

3. The back-end database server returns the following eight **result sets**:

   - **Web URL Result Set**, which returns the **store-relative URL** of the root of the site (2).
   - **Domain Group Cache Versions Result Set**, which returns information about the version numbers associated with the **domain group** map cache for this site (2).
   - **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 (2) metadata.
   - **Event Receivers Result Set**, which returns information about the **event receivers** defined for the site (2).
   - **Site Feature List Result Set**, which returns a list of default **feature identifiers** for the site collection that contains this site (2).
   - **Site Feature List Result Set**, which returns a list of feature identifiers for this site (2).
   - **Empty Result Set**, which is a **placeholder** set returned because the site (2) has no cached navigation scope information.

4. The front-end web server calls the proc_SecUpdateUserActiveStatus stored procedure to update the list of active users for the site (2).

5. The front-end web server continues collecting information about the current user list by calling the proc_GetListMetadataAndEventReceivers stored procedure using the Tabular Data Stream (TDS) protocol, as specified in [MS-TDS].

6. The back-end database server returns the following two result sets:

   - **List Metadata Result Set**, which returns the metadata associated with this list (1).
   - **Event Receivers Result Set**, which returns information about the event receivers defined for this list (1).

7. The front-end web server checks for the existence of the new account by calling the proc_SecGetAccountStatus stored procedure.

8. The back-end database server returns an **Account Status Result Set** with zero rows, indicating that the email address has not yet been used in this site collection.

9. The front-end web server gathers further information about the current count of registered users by calling the proc_SecGetCurrentUsersCount stored procedure.

10. The back-end database server returns the **User Count Result Set** to indicate the number of users registered with this site collection and any quota information.

11. The front-end web server verifies that the UserID to be created does not exist in the deleted user list by calling the proc_SecCheckDeletedAccounts stored procedure.

12. The back-end database server returns the **Login Result Set** with zero results, indicating the UserID does not yet exist.

13. The front-end web server gathers further information about the current site (2) by calling the proc_GetSiteMapById stored procedure.

14. The back-end database server returns the **Site Map By Id Result Set** with one row of data on the current site (2).

15. The front-end web server makes an **Lightweight Directory Access Protocol (LDAP) AddRequest** to the domain controller (DC) to add a user object with the supplied information.

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

17. The front-end web server 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 front-end web server builds a transactional dynamic SQL query that performs the following tasks:

    1. The query begins a new SQL **transaction (3)**.

    2. The proc_SecAddUser stored procedure is executed to add the requesting user to Active Directory environment.

    3. In the event of any error, the transaction is rolled back, and the final select statement is returned.

    4. If the new user account does not already exist, it is added to the user list using the proc_AddListItem stored procedure.

    5. In the event of any error, the transaction is rolled back, and the final select statement is returned.

    6. The transaction is committed, and the final select statement is returned with the variables used to create the new User account.

20. The back-end database server 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 front-end web server 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 front-end web server 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 indicating the successful attribute change.

25. The front-end web server verifies the availability of the new account's email address by calling the proc_SecGetPrincipalByE-mail stored procedure.

26. The back-end database server returns the **Principal User Information Result Set**, containing information about the user associated with the specified email address.

27. Control is passed back to the end-user client 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 (section 2.9.1.5).

29. The front-end web server requests status data by calling the proc_GetTpWebMetadataAndListMetadata stored procedure.

30. The back-end database server returns the following 14 result sets:

    - **Web URL Result Set**, which returns the URL of the root of the site (2).
    - **Domain Group Cache Versions Result Set**, which returns information about the version numbers associated with the domain group map cache for this site (2).
    - **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 (2).
    - **Site Category Result Set**, which returns categories of this site (2).
    - **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 (2).
    - **Site Feature List Result Set**, which returns a list of feature identifiers of this site (2).
    - **Empty Result Set**, which is a placeholder set returned because the site (2) has no cached navigation scope information.
    - **List Metadata Result Set**, which returns the metadata associated with the specified document list (1).
    - **NULL Unique Permissions Result Set**, which is a placeholder set returned because the list (1) has no individual list (1) permissions.
    - **Event Receivers Result Set**, which returns information about the event receivers defined for this document list (1).permissions.
    - **List Web Parts Result Set**, which returns information about the list (1) **Web Parts** defined for this document list (1).

31. The front-end web server requests information about the new account by calling the proc_SecResolvePrincipal stored procedure.

32. The back-end database server returns the **Principal Information Result Set** with a single row containing basic information about the user.

33. The front-end web server creates a dynamic SQL query, which selects information from the UserData view joined with the Docs view.

34. The back-end database server returns a dynamic SQL result set, which contains one row of data with the new user account information.

35. The front-end web server creates a dynamic SQL query, which selects information from Sec_SiteGroupsView.

36. The back-end database server returns a dynamic SQL result set with all site group membership levels.

37. The front-end web server creates a dynamic SQL query to check the requesting user permissions for this activity by calling the proc_SecGetUserPermissionOnGroup stored procedure.

38. The back-end database server returns a dynamic SQL result set representing the permission levels of the calling user.

39. The front-end web server calls the proc_SecUpdateUserActiveStatus stored procedure to update the list of active users for the site (2).

40. The front-end web server requests the account status for the new account by calling the proc_SecGetAccountStatus stored procedure.

41. The back-end database server returns the **Account Status Result Set**.

42. The front-end web server builds a transactional dynamic SQL query which performs the following tasks:

    1. The query begins a new SQL transaction (3).

    2. The proc_SecAddUser stored procedure is executed to add the requesting user to the appropriate security groups in the Active Directory environment.

    3. In the event of any error, the transaction (3) is rolled back, and the final select statement is returned.

    4. If the new user account does not already exist, it is added to the user list using the proc_AddListItem stored procedure.

    5. In the event of any error, the transaction is rolled back and the final select statement is returned.

    6. The transaction is committed, and the final select statement is returned with the variables used to create the new user account.

43. The back-end database server returns a dynamic SQL result set indicating the success of the add procedures and the variables used in the new account.

44. The front-end web server builds a transactional dynamic SQL query, which performs the following tasks:

    1. The query begins a new SQL transaction (3).

    2. The proc_AddListItem stored procedure is executed to add the new user account to the appropriate Windows SharePoint Services list (1).

    3. In the event of any error, the transaction (3) is rolled back and the final select statement is returned.

4. The new user account is updated, using the proc_UpdateListItem stored procedure with additional user data as necessary.

5. In the event of any error, the transaction is rolled back and the final select statement is returned.

6. The transaction is committed, and the final select statement is returned with the variables used to create the new user account.

45. The back-end database server returns a dynamic SQL result set with information about the newly created user.

46. The front-end web server 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 back-end database server 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 front-end web server can now add the new user account to the appropriate site group by calling the proc_SecAddUserToSiteGroup stored procedure.

49. The back-end database server responds with a **return code**, but no result sets are returned.

50. The front-end web server requests further status data by calling the proc_GetTpWebMetadataAndListMetadata stored procedure.

51. The back-end database server returns the following 14 result sets:

- **Web URL Result Set**, which returns the URL of the root of the site (2).
- **Domain Group Cache Versions Result Set**, which returns information about the version numbers associated with the domain group map cache for this site (2).
- **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 (2).
- **Site Category Result Set**, which returns categories of this site (2).
- **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 (2).
- **Site Feature List Result Set**, which returns a list of feature identifiers of this site (2).
- **Empty Result Set**, which is a placeholder set.
- **List Metadata Result Set**, which returns the metadata associated with the specified document list (1).

- **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 (1).
- **List Web Parts Result Set**, which returns information about the list (1) Web Parts defined for this document list (1).

52. The front-end web server creates a dynamic SQL query selecting all information from the Sec_SiteGroupsView view for this site (2) to generate the final website (2) display.
53. The back-end database server returns a dynamic SQL result set with all site group membership levels.
54. The front-end web server creates a dynamic SQL query, selecting information from the UserData view, Docs view, and AllUserData view for details on the website (2) display.
55. The back-end database server returns a dynamic SQL result set of user data as it applies to the current display.
56. The front-end web server creates a dynamic SQL request for user permission information by calling the proc_SecGetUserPermissionOnGroup stored procedure.
57. The back-end database server returns a dynamic SQL result set representing the permission levels of the calling user.
58. The front-end web server creates a dynamic SQL request for information from the UserData view and Docs view for website (2) display.
59. The back-end database server 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 end-user client 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 (2). 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 because of size limitations. The three figures in this section represent a single sequence. This specific example is for Active Directory operations involving Windows SharePoint Services 3.0.

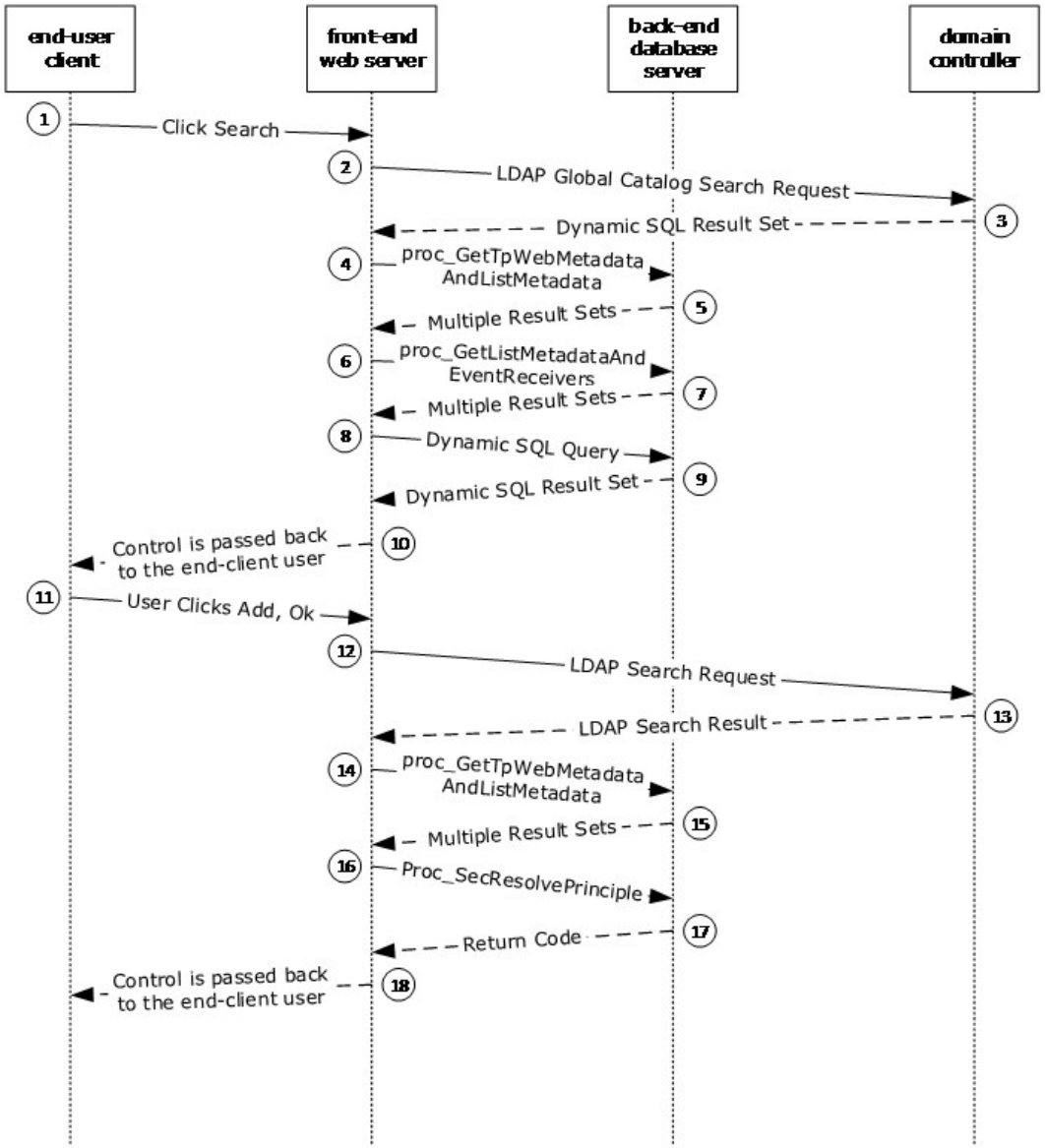

Figure 8: People Picker Browse Display UI, steps 1 through 18

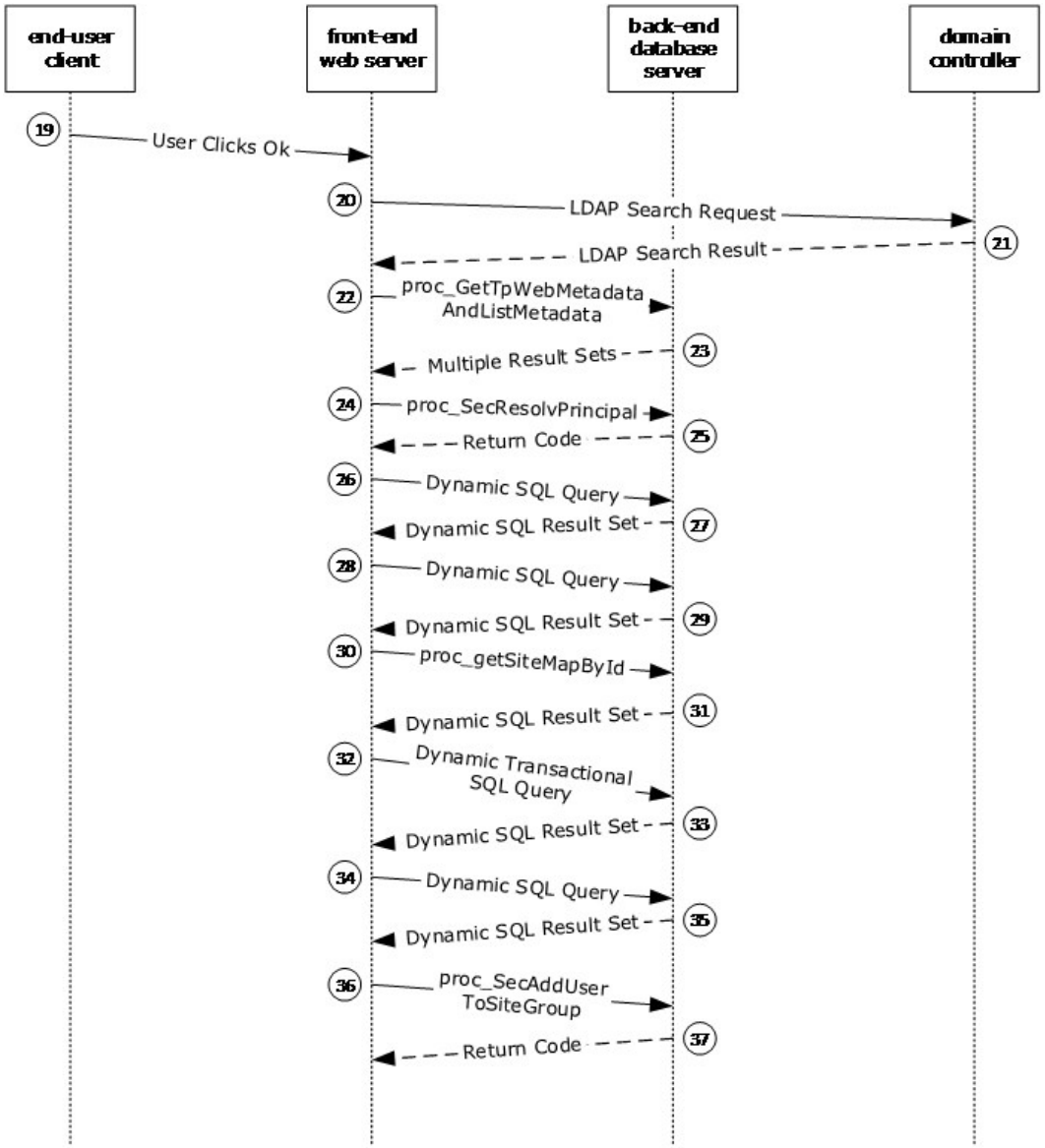

Figure 9: People Picker Browse Display UI, steps 19 through 37

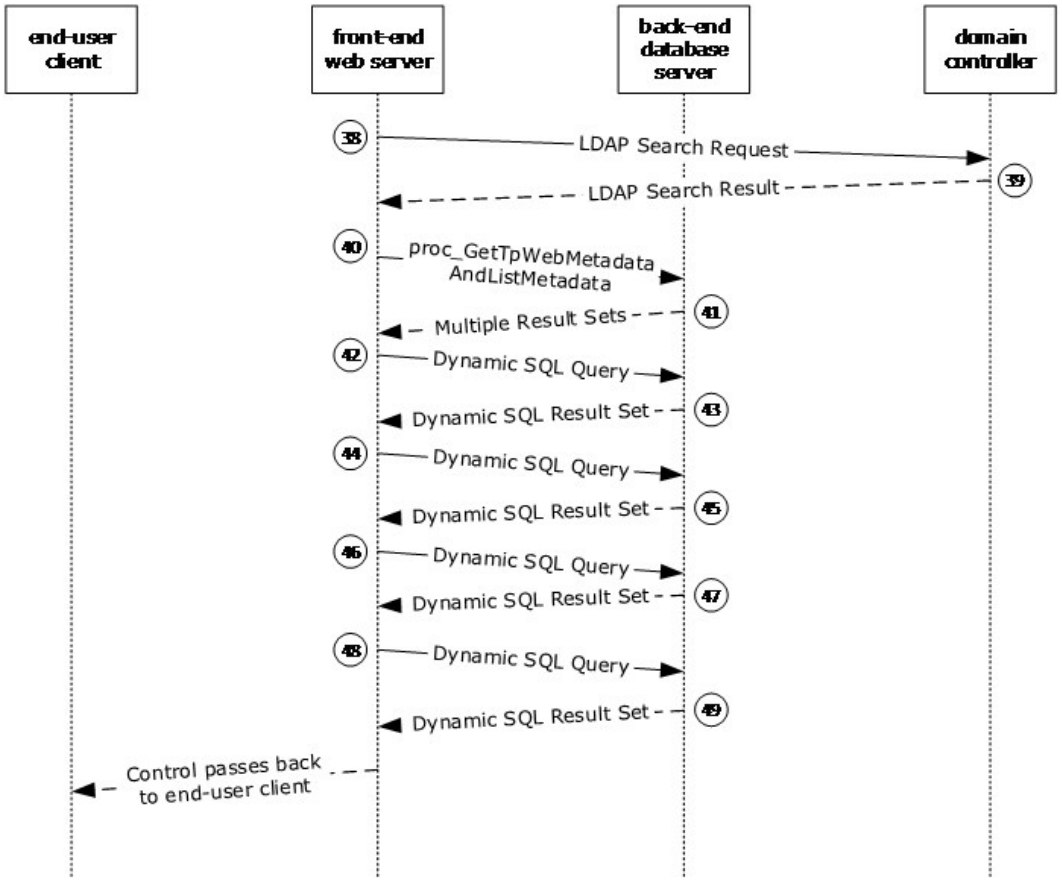

Figure 10: People Picker Browse Display UI, steps 38 through completion

This scenario is initiated from the "Select People and Groups" 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 (section 2.9.1.5).

The following actions happen:

1. The end-user client first sends a request to the front-end web server to search for the desired User display name.

2. The front-end web server sends a Lightweight Directory Access Protocol (LDAP) **Global Catalog Search** request to the domain controller (DC) asking for any match in the whole subtree for user objects or **group objects (1)** 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", **Simple Mail Transfer Protocol (SMTP)** or Session Initiation Protocol (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 objects and group objects that match the search string.

4. The front-end web server initializes information about the site (2) and its users by calling the proc_GetTpWebMetadataAndListMetadata stored procedure using the Tabular Data Stream (TDS) protocol, as specified in [MS-TDS].

5. The back-end database server returns five result sets:

   - **Web URL Result Set**, which returns the store-relative URL of the root of the site (2).
   - **Domain Group Cache Versions Result Set**, which returns information about the version numbers associated with the domain group map cache for this site (2).
   - **Domain Group Cache WFE Update Result Set**, which returns information to be used in recomputing the domain group map cache for the site (2).
   - **Site Metadata Result Set**, which returns specialized site metadata.
   - **Event Receivers Result Set**, which returns information about the event receivers defined for the site (2).

6. The front-end web server continues collecting information about the site's user list by calling the proc_GetListMetadataAndEventReceivers stored procedure.

7. The back-end database server returns the following four result sets:

   - **List Metadata Result Set**, which returns the permissions associated with the user list.
   - **NULL Unique Permissions Result Set**, which indicates that unique permissions do not exist for the list (1).
   - **List Event Receivers Result Set**, which is empty because there are no event receivers defined for this list (1).
   - **List Web Parts Result Set**, which contains information about the **list view** pages.

8. The front-end web server 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 email address columns.

9. The back-end database server returns one empty dynamic SQL result set, indicating that a match was not found.

10. The front-end web server displays the display name received from the DC as a candidate for selection.

11. The end user clicks **Add**, then **OK**. The end-user client closes the dialog and redirects the user to the User Information List web page.

12. The front-end web server negotiates authentication with the DC and then sends an LDAP **Search** request to the DC for an object that has a security identifier (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 front-end web server again initializes by gathering information about the site (2) by calling the proc_GetTpWebMetadataAndListMetadata stored procedure.

15. The back-end database server returns five result sets:

- **Web URL Result Set**, which contains the store-relative URL of the root of the site (2).
- **Domain Group Cache Versions Result Set**, which contains information about the version numbers associated with the domain group map cache for this site (2).
- **Domain Group Cache WFE Update Result Set**, which contains information to be used in recomputing the domain group map cache for the site (2).
- **Site Metadata Result Set**, which contains site metadata.
- **Event Receivers Result Set**, which contains information about the event receivers defined for the site (2).

16. The front-end web server sends a request to the back-end database server to find security principals (3) that might have **login name**, display name, or email address information matching the user account name returned from the DC. It does so by calling the proc_SecResolvePrincipal stored procedure.
17. The back-end database server responds with a return code, but no result sets are returned, indicating that no matches were found.
18. The front-end web server renders the name as resolved.
19. The end user clicks **OK** on the Add Users page, sending a request to the front-end web server to add the user to the site (2) and site group.
20. The front-end web server 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 front-end web server initializes again by calling the proc_GetTpWebMetadataAndListMetadata stored procedure.
23. The back-end database server returns the following 14 result sets:
    - **Web URL Result Set**, which contains the URL of the site (2).
    - **Domain Group Cache Versions Result Set**, which contains information about the version numbers associated with the domain group map cache for this site (2).
    - **Domain Group Cache WFE Update Result Set**, which contains binary data needed to refresh the domain group map cache.
    - **Site Metadata Result Set**, which contains site metadata.
    - **Event Receivers Result Set**, which contains information about the event receivers that are defined for this site (2).
    - **Site Category Result Set**, which contains the categories of this site (2).
    - **Site Metainfo Result Set**, which contains the specialized site metadata.
    - **Site Feature List Result Set**, which contains the list of default feature identifiers for the site collection that contains this site (2).

- **Site Feature List Result Set**, which contains the list of feature identifiers of this site (2).
- **Empty Result Set**, which is a placeholder set.
- **List Metadata Result Set**, which contains the metadata associated with the specified document list (1).
- **NULL Unique Permissions Result Set**, which indicates that there are no special permissions set on the user information list.
- **Event Receivers Result Set**, which contains information about the event receivers defined for the document list (1).
- **List Web Parts Result Set**, which contains information about the list view pages defined for the user information list (1).

24. The front-end web server sends a request to resolve the selected user names by calling the proc_SecResolvePrincipal stored procedure.

25. The back-end database server responds with a return code, but no result sets are returned, indicating that the user was not found.

26. The front-end web server creates a dynamic SQL query that selects information from the Sec_SiteGroupsView.

27. The back-end database server returns a dynamic SQL result set with all site group membership levels signifying the owner of all groups.

28. The front-end web server builds a dynamic SQL 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 back-end database server 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 front-end web server requests the site map by calling the proc_getSiteMapById stored procedure.

31. The back-end database server returns the **Site Map by Id Result Set**.

32. The front-end web server builds a dynamic transactional SQL Query to add the user to the site collection. The following actions happen:

    1. The transaction (3) 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 back-end database server, containing the return code and information about the added user.

34. The front-end web server constructs a dynamic SQL query, selecting full user information about the added user.

35. The back-end database server returns a dynamic SQL result set with the requested information.

36. The front-end web server sends a request to the back-end database server to add the user to the current site group by calling the proc_SecAddUserToSiteGroup stored procedure.

37. The back-end database server responds with a return code, but no result sets are returned.

38. The front-end web server 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 front-end web server again initializes its information about the site (2) by calling the proc_GetTpWebMetadataAndListMetadata stored procedure.

41. The back-end database server returns the following 14 result sets:

    - **Web URL Result Set**, which returns the URL of the root of the site (2).
    - **Domain Group Cache Versions Result Set**, which returns information about the version numbers associated with the domain group map cache for this site (2).
    - **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 (2).
    - **Site Category Result Set**, which returns the categories of the site (2).
    - **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 (2).
    - **Site Feature List Result Set**, which returns the list of feature identifiers of this site (2).
    - **Empty Result Set**, which is a placeholder set.
    - **List Metadata Result Set**, which returns the metadata associated with the specified document list (1).
    - **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 (1).
    - **List Web Parts Result Set**, which returns information about the list (1) Web Parts defined for this document list (1).

42. The front-end web server creates a dynamic SQL query that selects information from the Sec_SiteGroupsView view.

43. The back-end database server returns a dynamic SQL result set with all site group membership levels, signifying the owner of all groups.

44. The front-end web server builds a dynamic SQL query to obtain updated information about the site group to which the user was added.

45. The back-end database server returns one dynamic SQL result set containing information about the site group.

46. The front-end web server 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 back-end database server 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 front-end web server builds a dynamic SQL query to obtain more user information for the site group to which the user has been added.

49. The back-end database server returns one dynamic SQL result set of information about the newly added user.

50. Control is passed back to the end-user client.

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 (3) to a SharePoint list (1) and uses the Check Names function to confirm the existence of the new member (3) 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 because of size limitations. The three figures in this section represent a single sequence. This specific example is for Active Directory operations involving Windows SharePoint Services 3.0.

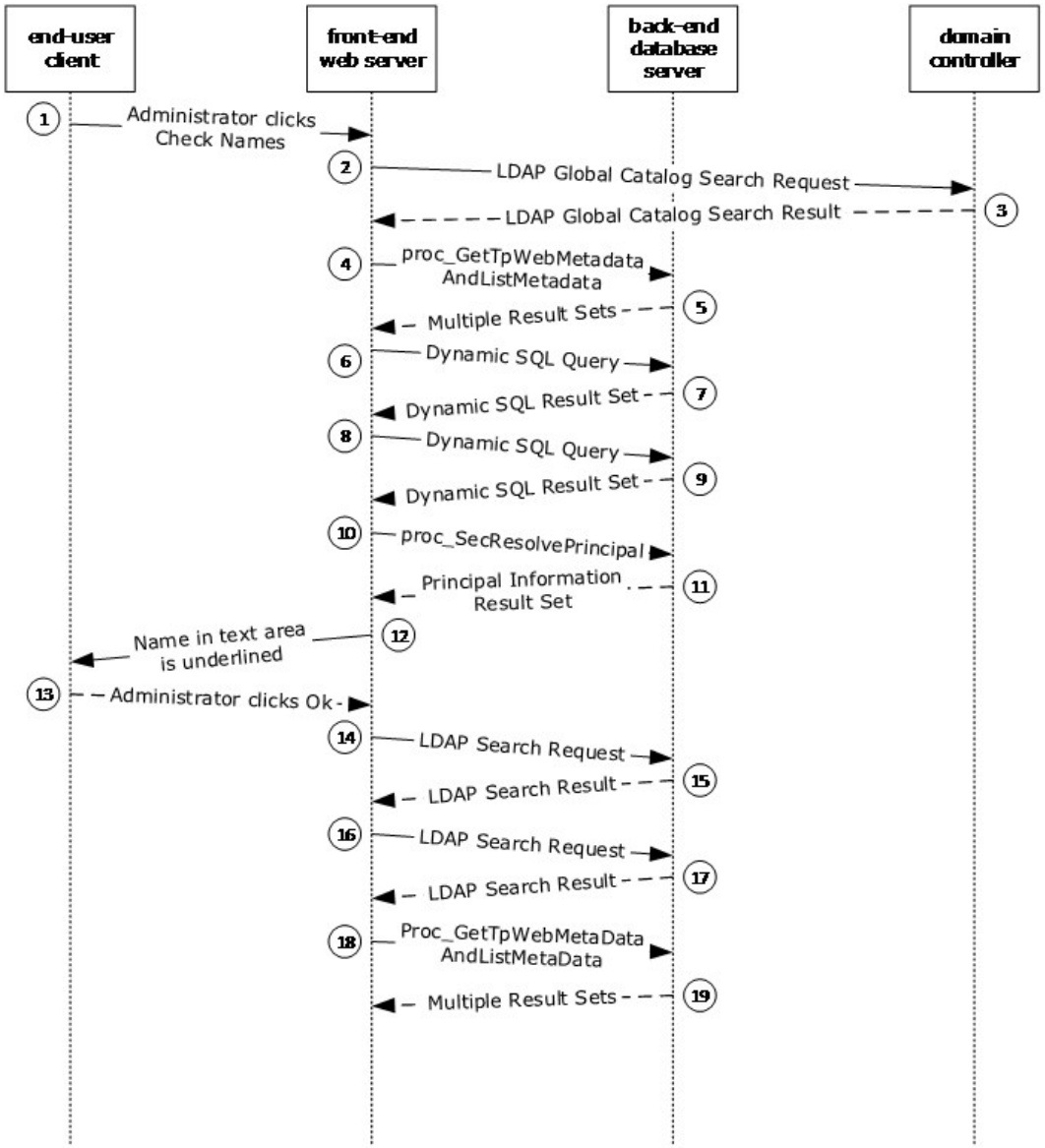

Figure 11: People Picker Check Name UI, steps 1 through 19

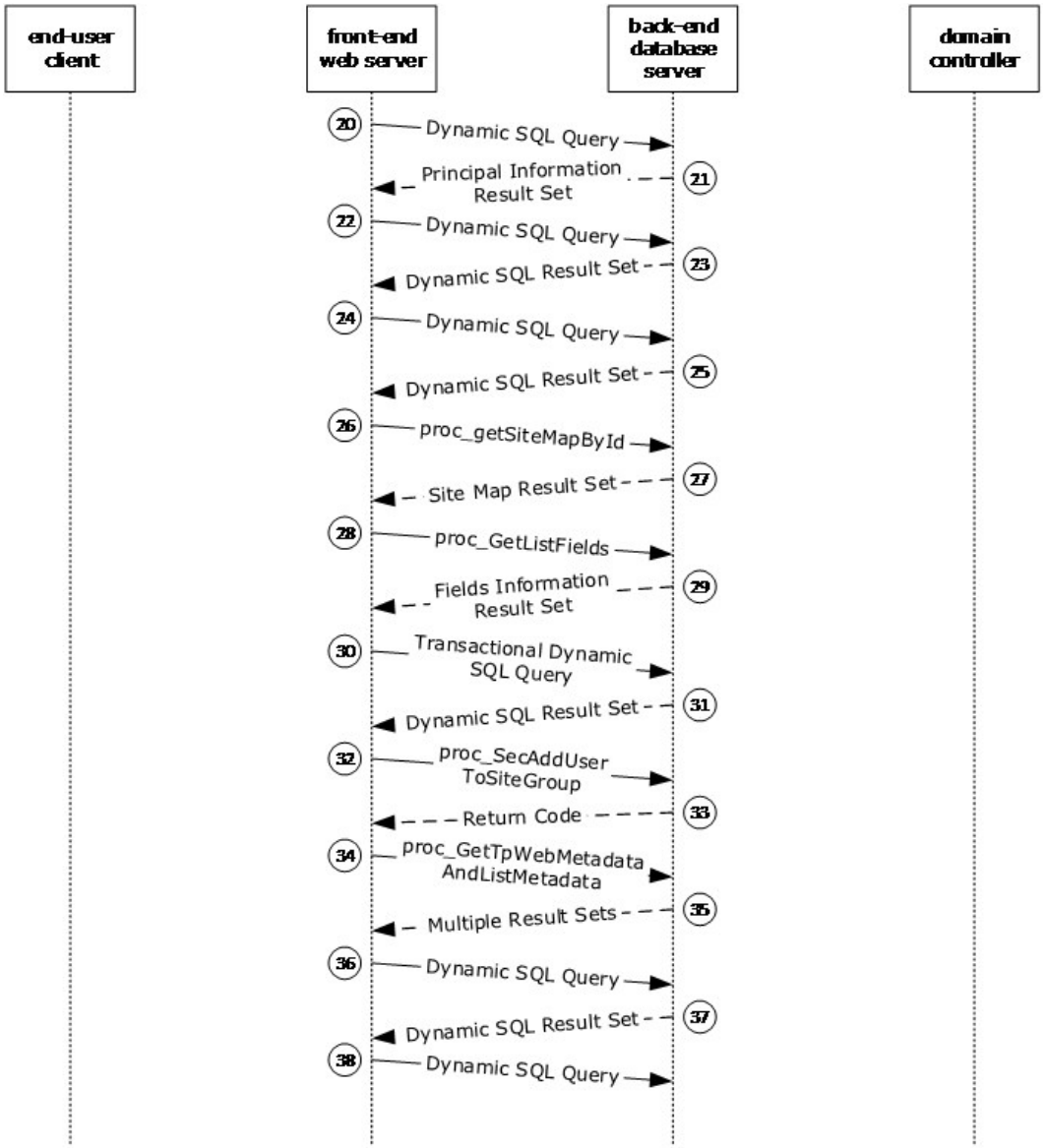

Figure 12: People Picker Check Name UI, steps 20 through 38

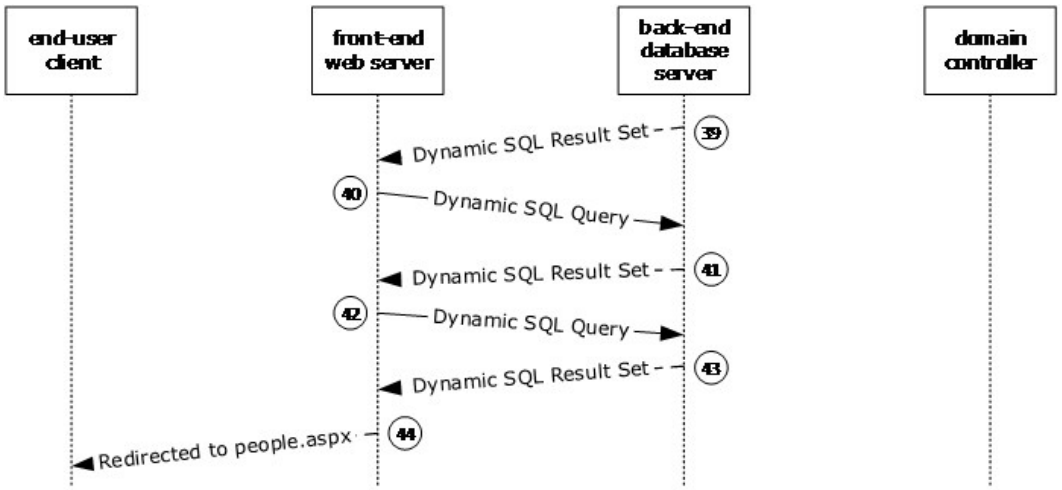

Figure 13: People Picker Check Name UI, steps 39 through completion

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

- The Windows SharePoint Services site (2) 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. From the end-user client, the user clicks the "Check Names" icon.
2. The front-end web server sends an Lightweight Directory Access Protocol (LDAP) **Global Catalog Search** request to the domain controller (DC), asking for any match in the whole subtree for user or group objects (1) 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", Simple Mail Transfer Protocol (SMTP) or Session Initiation Protocol (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 objects and group objects (1) that match the search string.
4. The front-end web server first collects metadata for the site (2) and the document list (1) by calling the proc_GetTpWebMetaDataAndListMetaData stored procedure using the Tabular Data Stream (TDS) protocol, as specified in [MS-TDS].
5. The back-end database server returns 14 result sets:
   - **Web Url Result Set**, which returns the store-relative URL of the root of the site (2).
   - **Domain Group Cache Versions Result Set**, which returns information about the version numbers associated with the domain group map cache for the site (2).

- **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 (2) metadata.
- **Event Receivers Result Set**, which returns information about the event receivers defined for the site (2).
- **Site Category Result Set**, which returns the categories of the site (2).
- **Site MetaInfo Result Set**, which returns the specialized site (2) metadata.
- **Site Feature List Result Set**, which returns a list of feature identifiers for the site collection that contains this site (2).
- **Site Feature List Result Set**, which returns a list of feature identifiers for the site (2).
- **Empty Result Set**, which is returned because the **METADATA_WEB** and **METADATA_WEB_NAVSTRUCT** flags are set and the site (2) has no cached scope information.
- **List Metadata Result Set**, which returns metadata associated with the document list (1).
- **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 (1) do not exist.
- **Event Receivers Result Set**, which returns information about the event receivers defined for the document list (1).
- **List Web Parts Result Set**, which returns information about the list (1) Web Parts defined for the document list (1).

6. The front-end web server 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 back-end database server returns a dynamic SQL result set containing information about the Team Site owners, the Team Site visitors, and the Team Site members (3).
8. The front-end web server builds a dynamic SQL query regarding which permissions the requesting user has within the site group (section 2.9.1.5). It does this by calling the proc_SecGetUserPermissionOnGroup stored procedure using TDS.
9. The back-end database server returns a dynamic SQL result set consisting of the output variables of the stored procedure.
10. The front-end web server then retrieves information about the user to be added by calling the proc_SecResolvePrincipal stored procedure using TDS.
11. The back-end database server 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 front-end web server negotiates authentication (1) with the DC and then sends an LDAP **Search** request to the DC for an object that has a security identifier (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 front-end web server again negotiates authentication (1) 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 front-end web server then collects metadata for the site (2) and the document list (1) by calling the proc_GetTpWebMetaDataAndListMetaData stored procedure using TDS.

19. The back-end database server returns 14 result sets:

    - **Web Url Result Set**, which returns the store-relative URL of the root of the site (2).
    - **Domain Group Cache Versions Result Set**, which returns information about the version numbers associated with the domain group map cache for the site (2).
    - **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 (2) metadata.
    - **Event Receivers Result Set**, which returns information about the event receivers defined for the site (2).
    - **Site Category Result Set**, which returns the categories of the site (2).
    - **Site MetaInfo Result Set**, which returns the specialized site (2) metadata.
    - **Site Feature List Result Set**, which returns a list of feature identifiers for the site collection that contains this site (2).
    - **Site Feature List Result Set**, which returns a list of feature identifiers for the site (2).
    - **Empty Result Set**, which is returned because the **METADATA_WEB** and **METADATA_WEB_NAVSTRUCT** flags are set, and the site (2) has no cached scope information.
    - **List Metadata Result Set**, which returns metadata associated with the document list (1).
    - **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 (1) do not exist.
    - **Event Receivers Result Set**, which returns information about the event receivers defined for the document list (1).
    - **List Web Parts Result Set**, which returns information about the list (1) Web Parts defined for the document list (1).

20. The front-end web server then retrieves information about the user to be added by calling the proc_SecResolvePrincipal stored procedure using TDS.

21. The back-end database server returns the **Principal Information Result Set**, consisting of a single row of information about the user to be added.

22. The front-end web server 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 back-end database server returns a dynamic SQL result set containing information about the Team Site owners, the Team Site visitors, and the Team Site members (3).

24. The front-end web server 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 back-end database server returns a dynamic SQL result set consisting of the output variables of the proc_SecGetUserPermissionOnGroup stored procedure.

26. The front-end web server then retrieves the database and URL mapping information for the site collection. It does this by calling the proc_getSiteMapById stored procedure from the Windows SharePoint Services configuration database using TDS.

27. The back-end database server returns the **Site Map by Id Result Set**.

28. The front-end web server then retrieves the mapping of **fields (2)** in the document list (1) by calling the proc_GetListFields stored procedure using TDS.

29. The back-end database server returns the **Fields Information Result Set**, consisting of a single row containing a single column of a Windows SharePoint Services implementation-specific version string followed by an **XML** representation of the field definitions.

30. The front-end web server builds a transactional dynamic SQL query to add an entry for the new user to the list of user information stored in the back-end database server. This query is sent to the SQL server using TDS. On the SQL server, the following actions occur:

    1. The query begins a new SQL transaction (3).

    2. The query attempts to add the new user to the list of user information in the back-end database server by calling the proc_SecAddUser stored procedure using TDS.

    3. 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.

    4. If the user's UserId is not found in the UserData table, the query attempts to add the list item to the document list (1) by calling the proc_AddListItem stored procedure using TDS.

    5. 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 is "0" and proc_AddListItem's return code (if it runs) is "0".

31. The back-end database server returns a dynamic SQL result set that consists of information about the new user.

32. The front-end web server 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 back-end database server responds with a return code, but no result sets are returned.

34. The front-end web server then collects metadata for the site (2) and the document list (1) by calling the proc_GetTpWebMetaDataAndListMetaData stored procedure using TDS.

35. The back-end database server returns 14 result sets:

    - **Web Url Result Set**, which returns the store-relative URL of the root of the site (2).
    - **Domain Group Cache Versions Result Set**, which returns information about the version numbers associated with the domain group map cache for the site (2).
    - **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 (2) metadata.
    - **Event Receivers Result Set**, which returns information about the event receivers defined for the site (2).
    - **Site Category Result Set**, which returns the categories of the site (2).
    - **Site MetaInfo Result Set**, which returns the specialized site (2) metadata.
    - **Site Feature List Result Set**, which returns a list of feature identifiers for the site collection that contains this site (2).
    - **Site Feature List Result Set**, which returns a list of feature identifiers for the site (2).
    - **Empty Result Set**, which is returned because the **METADATA_WEB** and **METADATA_WEB_NAVSTRUCT** flags are set and the site (2) has no cached scope information.
    - **List Metadata Result Set**, which returns metadata associated with the document list (1).
    - **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 (1) do not exist.
    - **Event Receivers Result Set**, which returns information about the event receivers defined for the document list (1).
    - **List Web Parts Result Set**, which returns information about the list (1) Web Parts defined for the document list (1).

36. The front-end web server 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 back-end database server returns a dynamic SQL result set containing information about the Team Site owners, the Team Site visitors, and the Team Site members (3).

38. The front-end web server builds a dynamic SQL query to retrieve information about the site (2) and document list (1) as it relates to the requesting user. This query is sent to the SQL server using TDS.

39. The back-end database server returns a dynamic SQL result set containing information about the current site (2) and document list (1).

40. The front-end web server 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 back-end database server returns a dynamic SQL result set consisting of the output variables of the stored procedure.

42. The front-end web server builds a dynamic SQL query to retrieve information about the site (2) and document list (1) as it relates to the newly added user. This query is sent to the SQL server using TDS.

43. The back-end database server returns a dynamic SQL result set containing information about the current site (2) and document list (1).

44. The front-end web server redirects the end-user client 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. This example is specifically for operations involving Microsoft SharePoint Foundation 2010.

Note The following steps and diagram consolidate multiple front-end web server to back-end database server actions, and multiple front-end web server 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 (2) 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 Windows SharePoint Services site (2) containing the document library.
- From a command prompt window on the end-user client machine, the user has typed the following command:

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

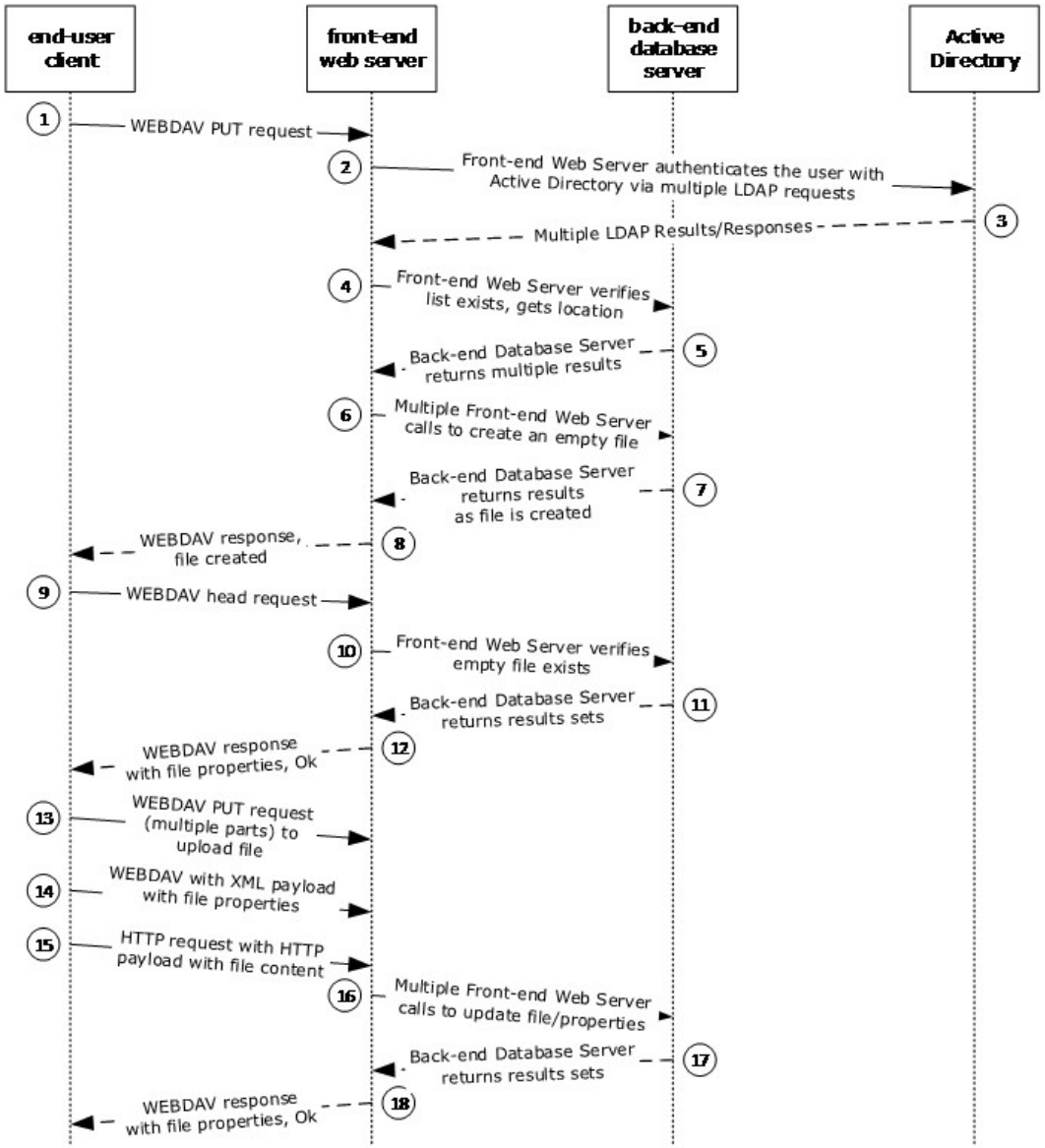

Figure 14: Create a SharePoint Document Library file from the client console

The [MS-WSSFO2] examples referenced for more details in the following steps use the SharePoint programming API; the actual steps can 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 end-user 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 can involve multiple LDAP requests with the Active Directory, especially if the user has not previously visited the site (2).

3. Active Directory responds with multiple LDAP results.

4. In multiple roundtrips with the back-end database server, the front-end web server locates the content database for the document library, and verifies that the library exists.

5. The back-end database server returns multiple objects for the site collection, website (2) and library to the front-end web server.

6. In multiple roundtrips with the back-end database server, the front-end web server creates an empty file in the document library, and then, if successful, the front-end web server also verifies that the user has permissions to access and write to the document library.

7. The back-end database server returns multiple result sets as part of the process to create the file.

8. The front-end web server returns a WebDAV response indicating the file was created successfully.

9. The client sends a WebDAV **HEAD** request to front-end web server 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 back-end database server, the front-end web server retrieves the file.

11. The back-end database server returns multiple results sets as part of the process to retrieve the file.

12. In response to the **HEAD** request, the front-end web server sends a response reporting that the request was successful.

13. Then the client sends a WebDAV **PUT** request to the front-end web server containing multiple parts, to upload the file and to update the file properties.

14. The client sends a WebDAV request to the front-end web server with an XML payload that has the file properties from the client.

15. The client sends a HTTP request to the front-end web server 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 back-end database server, the front-end web server updates the file and its properties in the document library.

17. The back-end database server returns multiple result sets as part of the process to update the files.

For more information about authentication (1), see section 2.9.2.1.

For more information about the scenario when the user has not previously visited the site (2), see [MS-WSSFO2] section 4.2 for SharePoint Foundation 2010.

1. In multiple roundtrips with the back-end database server, the front-end web server locates the content database for the document library, and verifies that the library exists.

2. The back-end database server returns multiple objects for the site collection, website (2) and library to the front-end web server.

3. In multiple roundtrips with the back-end database server, the front-end web server creates an empty file in the document library, and then, if successful, the front-end web server also verifies that the user has permissions to access and write to the document library.

4. The back-end database server returns multiple result sets as part of the process to create the file.

5. The front-end web server returns a WebDAV response indicating the file was created successfully.

6. The client sends a WebDAV **HEAD** request to front-end web server with the URL to the hello.txt file in the document library, to verify the success of the previous call.

7. In multiple roundtrips with the back-end database server, the front-end web server retrieves the file.

8. The back-end database server returns multiple results sets as part of the process to retrieve the file.

9. In response to the **HEAD** request, the front-end web server sends a response reporting that the request was successful.

10. Then the client sends a WebDAV **PUT** request to the front-end web server containing multiple parts, to upload the file and to update the file properties.

11. The client sends a WebDAV request to the front-end web server with an XML payload that has the file properties from the client.

12. The client sends a HTTP request to the front-end web server with an HTTP payload that has the file content; in this example, that content is simply the text "hello".

13. In multiple roundtrips with the back-end database server, the front-end web server updates the file and its properties in the document library.

14. The back-end database server returns multiple result sets as part of the process to update the files.

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

1. In multiple roundtrips with the back-end database server, the front-end web server creates an empty file in the document library, and then, if successful, the front-end web server also verifies that the user has permissions to access and write to the document library.

2. The back-end database server returns multiple result sets as part of the process to create the file.

3. The front-end web server returns a WebDAV response indicating the file was created successfully.

4. The client sends a WebDAV **HEAD** request to front-end web server with the URL to the hello.txt file in the document library, to verify the success of the previous call.

5. In multiple roundtrips with the back-end database server, the front-end web server retrieves the file.

6. The back-end database server returns multiple results sets as part of the process to retrieve the file.

7. In response to the **HEAD** request, the front-end web server sends a response reporting that the request was successful.

8. Then the client sends a WebDAV **PUT** request to the front-end web server containing multiple parts, to upload the file and to update the file properties.

9. The client sends a WebDAV request to the front-end web server with an XML payload that has the file properties from the client.

10. The client sends a HTTP request to the front-end web server with an HTTP payload that has the file content; in this example, that content is simply the text "hello".

11. In multiple roundtrips with the back-end database server, the front-end web server updates the file and its properties in the document library.

12. The back-end database server returns multiple result sets as part of the process to update the files.

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

1. The front-end web server returns a WebDAV response indicating the file was created successfully.

2. The client sends a WebDAV **HEAD** request to front-end web server with the URL to the hello.txt file in the document library, to verify the success of the previous call.

3. In multiple roundtrips with the back-end database server, the front-end web server retrieves the file.

4. The back-end database server returns multiple results sets as part of the process to retrieve the file.

5. In response to the **HEAD** request, the front-end web server sends a response reporting that the request was successful.

6. Then the client sends a WebDAV **PUT** request to the front-end web server containing multiple parts, to upload the file and to update the file properties.

7. The client sends a WebDAV request to the front-end web server with an XML payload that has the file properties from the client.

8. The client sends a HTTP request to the front-end web server with an HTTP payload that has the file content; in this example, that content is simply the text "hello".

9. In multiple roundtrips with the back-end database server, the front-end web server updates the file and its properties in the document library.

10. The back-end database server returns multiple result sets as part of the process to update the files.

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

1. In response to the **HEAD** request, the front-end web server sends a response reporting that the request was successful.

2. Then the client sends a WebDAV **PUT** request to the front-end web server containing multiple parts, to upload the file and to update the file properties.

3. The client sends a WebDAV request to the front-end web server with an XML payload that has the file properties from the client.

4. The client sends a HTTP request to the front-end web server with an HTTP payload that has the file content; in this example, that content is simply the text "hello".

5. In multiple roundtrips with the back-end database server, the front-end web server updates the file and its properties in the document library.

6. The back-end database server returns multiple result 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.

Upon completing the update, the front-end web server sends a WebDAV response reporting that the request was successful.

4 Microsoft Implementations

- Windows Server 2003 operating system
- Windows Server 2008 operating system with Service Pack 2 (SP2)
- Windows Server 2008 R2 operating system

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:

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

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

6 Index

A

Active Directory account creation new UI 35
Active Directory People Picker Browse display UI 44
Active Directory People Picker Check Name UI 52
Additional considerations 34
Applicable protocols 19
Architecture 11
Assumptions 21
Authentication – security considerations 30
Authorization for user and group administration – security considerations 25

C

Capability negotiation 24
Change tracking 67
Coherency requirements 25
Communications 20
  with other systems 21
  within the system 21
Component dependencies 21
Concepts 11
Considerations
  additional 34
  security 25
Create a document library file from the client console 60
Creating a sharepoint document library file from the client console
  overview 22

D

Dependencies
  with other systems 21
  within the system 21
Design intent
  creating a sharepoint document library file from the client console 22
  overview 22

E

Environment 20
Error handling 25
Examples 35
  Active Directory account creation new UI 35
  Active Directory People Picker Browse display UI 44
  Active Directory People Picker Check Name UI 52
  create a document library file from the client console 60
Extensibility
  Microsoft implementations 66
  overview 24
External dependencies 21

F

Functional architecture 11
Functional requirements - overview 11

G

Glossary 7

H

Handling requirements 25

I

Implementations - Microsoft 66
Implementer - security considerations 25
Informative references 9
Initial state 21
Introduction 6

M

Microsoft implementations 66

O

Overview
  scale-out technologies 12
  storage architecture 12
  summary of protocols 19
  synopsis 11

P

Preconditions 21

R

References 9
Requirements
  coherency 25
  error handling 25
  overview 11
  preconditions 21

S

Scale-out technologies overview 12
Security considerations 25
  authentication 30
  authorization for user and group administration 25
Storage architecture overview 12
System architecture 11
System dependencies 20
  with other systems 21
  within the system 21
System errors 25
System protocols 19
System requirements - overview 11

System use cases
creating a sharepoint document library file from the client console 22
overview 22

T

Table of protocols 19
Tracking changes 67

U

Use cases 22
creating a sharepoint document library file from the client console 22

V

Versioning
Microsoft implementations 66
overview 24