



OpenText™ Documentum™ Content Management

Server and Server Extensions Installation Guide

Install the OpenText™ Documentum™ Content Management Server and its extensions.

EDCSY250400-IGD-EN-01

**OpenText™ Documentum™ Content Management
Server and Server Extensions Installation Guide**
EDCSY250400-IGD-EN-01
Rev.: 2025-Dec-31

This documentation has been created for OpenText™ Documentum™ Content Management CE 25.4.
It is also valid for subsequent software releases unless OpenText has made newer documentation available with the product,
on an OpenText website, or by any other means.

Open Text Corporation

275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1

Tel: +1-519-888-7111
Toll Free Canada/USA: 1-800-499-6544 International: +800-4996-5440
Fax: +1-519-888-0677
Support: <https://support.opentext.com>
For more information, visit <https://www.opentext.com>

© 2025 Open Text

Patents may cover this product, see <https://www.opentext.com/patents>.

Disclaimer

No Warranties and Limitation of Liability

Every effort has been made to ensure the accuracy of the features and techniques presented in this publication. However,
Open Text Corporation and its affiliates accept no responsibility and offer no warranty whether expressed or implied, for the
accuracy of this publication.

Table of Contents

1	Overview	7
2	Documentum CM Server	9
2.1	Installation overview	9
2.2	Prerequisites	18
2.3	Configuring for certificate-based SSL communication	40
2.4	Configuring Documentum Secret Integration Service	49
2.5	Installing and configuring Documentum CM Server	52
2.6	Completing the installation	84
2.7	Post-installation requirements and tasks	86
2.8	Configuring operation mode	113
2.9	Configuring Always On availability group of Microsoft SQL Server	116
2.10	Java Method Server for High-Availability	116
2.11	Installing Documentum CM Server with Microsoft Cluster Services ...	124
2.12	Uninstalling Documentum CM Server	134
2.13	Troubleshooting	136
2.14	Documentum CM Server installation directories and repository configuration scripts	150
2.15	Object type categories for Oracle database storage	159
2.16	Defining Oracle database parameters for repository tables	162
2.17	Independent Java Method Server	167
3	Distributed Content	177
3.1	Distributed Configuration components	177
3.2	Distributed Configuration models	194
3.3	Installing Branch Office Caching Services servers	234
3.4	Post-installation task	238
3.5	Configuring Branch Office Caching Services servers	239
3.6	Uninstalling Branch Office Caching Services server	254
3.7	Upgrading Branch Office Caching Services server	255
3.8	Installing and configuring Messaging Service servers	255
3.9	Configuring Accelerated Content Services, Branch Office Caching Services, Messaging Service, and Tomcat for SSL connections	263
3.10	Installing and configuring remote Content Server in distributed or load-balanced configurations	271
3.11	Implementing single-repository models	276
3.12	Implementing multi-repository models	298
3.13	Managing single-repository models	319
3.14	Managing multi-repository models	328
3.15	Federation infrastructure	342
4	Foundation CMIS API	345

4.1	Introduction	345
4.2	Configuration settings	346
4.3	Configuring token-based authentication for Browser binding	354
4.4	Deploying on web application servers	355
4.5	Post-deployment tasks	364
4.6	Enabling OTDS authentication in Foundation CMIS API	366
5	Foundation Java API	367
5.1	Prerequisites	367
5.2	Installing Foundation Java API	376
5.3	Post-installation task	380
5.4	Uninstalling Foundation Java API	380
5.5	IPv6 support	381
5.6	Troubleshooting	382
6	Foundation SOAP API	383
6.1	Introduction	383
6.2	About Foundation SOAP API deployment	383
6.3	Configuration	385
6.4	Deploying on web application servers	393
6.5	Post-deployment task	400
6.6	Enabling OTDS authentication in Foundation SOAP API services	400
7	Foundation REST API	401
7.1	Introduction	401
7.2	Configuring Foundation Java API	401
7.3	Deploying on web application servers	402
7.4	Post-deployment task	407
7.5	Configuring Foundation REST API to connect to HashiCorp Vault	407
7.6	Accessing Foundation REST API URL in an IPv6 environment	407
7.7	Validating Foundation REST API deployment	408
8	Documentum Administrator	409
8.1	Planning for deployment	409
8.2	Preparing the client hosts	411
8.3	Preparing the application server host	412
8.4	Deploying Documentum Administrator	421
8.5	Post-deployment tasks	439
8.6	Troubleshooting	451
9	Thumbnail Server	455
9.1	Introduction	455
9.2	Installation overview	459
9.3	Verifying Thumbnail Server installation	474

9.4	Configuring Thumbnail Server in a trusted content store	474
9.5	Administration and configuration	475
10	XML Store	479
10.1	Introduction	479
10.2	Deploying XML Store	482
10.3	Enabling XML Store for a repository	482
10.4	Enabling XML Store to work with multiple Documentum CM Servers	483

Chapter 1

Overview

This guide is intended for system and repository administrators for installing and configuring OpenText Documentum Content Management (CM) Server and its extensions on supported operating systems, web browsers, and web application servers. Users of this guide must have working knowledge of Documentum CM Server and its extensions and have an understanding of relational database management systems (RDBMS).

Highlights of the components:

- OpenText™ Documentum™ Content Management Server: The foundation of OpenText Documentum Content Management (CM)'s content management system for creating, managing, delivering, and archiving enterprise content.
- OpenText™ Documentum™ Content Management Distributed Content: A set of components such as OpenText™ Documentum™ Content Management Accelerated Content Services, OpenText™ Documentum™ Content Management Branch Office Caching Services, and OpenText™ Documentum™ Content Management Messaging Service that can be configured to provide distributed content for web clients.
- OpenText™ Documentum™ Content Management Foundation CMIS API: A web application as an archive file that is deployed on web application servers.
- OpenText™ Documentum™ Content Management Foundation Java API: An application programming interface (API) that consists of a set of Java classes for writing applications that work with Documentum CM Server or need access to the repository.
- OpenText™ Documentum™ Content Management Foundation SOAP API: A service-oriented architecture that includes a set of web services and tools for creating custom services that can be deployed on web application servers.
- OpenText™ Documentum™ Content Management Foundation REST API: A set of RESTful web service interfaces interacting with OpenText Documentum CM that can be deployed on web application servers.
- OpenText™ Documentum™ Content Management Documentum Administrator: A tool for administering, configuring, and maintaining Documentum CM Servers and repositories.
- OpenText™ Documentum™ Content Management Thumbnail Server: A server to retrieve the low-resolution image files from a designated thumbnail file store on Documentum CM Server.
- OpenText™ Documentum™ Content Management XML Store: A component that provides extended capabilities to Documentum CM Server to store and process XML data in the repository.

Chapter 2

Documentum CM Server

This documentation is intended for system administrators responsible for installing and configuring Documentum CM Server. The *OpenText Documentum Content Management - On-Premises Upgrade and Migration Guide (EDCCS250400-UMD)* contains the instructions on upgrading Documentum CM Server.

2.1 Installation overview

Use the information provided in this section to understand the Documentum CM Server architecture and the components you must install while installing the Documentum CM Server. Make sure that you plan the installation based on your business needs.

2.1.1 Documentum CM Server architecture

This section provides basic information about the Documentum CM Server architecture and the components that are installed during the installation process.

2.1.1.1 Documentum CM Server and repository

Documentum CM Server is a powerful, robust, and scalable enterprise content management system that provides advanced content management and process management functions to organize, control, and access all your information assets in your organization.

Documentum CM Server manages content stored in object-based repositories. A repository comprises of a storage that stores native content files, and a relational database management system (RDBMS) that stores properties of these content files called metadata, such as document owner, version, and creation date. Metadata describes the content object and its relationship between other objects in repository, and is used to manage and search for content.

Documentum CM Server itself consists of several distinct processes and components that are mostly transparent to the user during installation:

- Application server

Documentum CM Server uses a private embedded application server as a container for Java Method Server (JMS), OpenText Documentum Content Management (CM) Accelerated Content Services, and other components.

- Java Method Server

Java Method Server is a customized version of Tomcat that executes Documentum CM Server Java methods. One Java Method Server is installed with each Documentum CM Server installation.

- Accelerated Content Services server

Accelerated Content Services server is a lightweight server that is automatically created during Documentum CM Server installation. The Accelerated Content Services server reads and writes content for web-based client applications using HTTP and HTTPS protocols. Accelerated Content Services servers do not modify object metadata but write content to storage areas.

- OpenText Documentum Content Management (CM) Foundation Java API

Foundation Java API provides the programming interface that client applications use to communicate with Documentum CM Server.

2.1.1.2 Connection broker

Documentum CM Server clients connect to Documentum CM Server through connection brokers. A connection broker is a process that provides client sessions with Documentum CM Server connection information, such as their IP addresses and port numbers, as well as proximity values of their network locations.

The connection brokers that handle a client connection request are defined in the `dfc.properties` file of the client. When a user or application requests a repository connection, the request goes to a connection broker identified in the client `dfc.properties` file. The connection broker returns the connection information for the repository or a particular server identified in the request.

Connection brokers do not request information from Documentum CM Servers, but rely on the servers to regularly broadcast their connection information to them. When Documentum CM Server starts, it automatically broadcasts information about itself to one or more connection brokers. Each connection broker that receives the broadcast adds the Documentum CM Server to its list of available servers. The information on connection broker is configured in the server config object (`dm_server_config`) of the server.

Each Documentum CM Server installation must have at least one connection broker. The first connection broker is started as part of the installation process.

When a client application wants to connect to a repository, the following occurs:

1. The client contacts the connection broker and requests the information it needs to connect with a Documentum CM Server for the requested repository.
2. The connection broker sends back the IP address for the host on which the Documentum CM Server resides and the port number that the Documentum CM Server is using.
3. The client application uses that information to open a connection to Documentum CM Server.

2.1.1.3 Global registry

When a Documentum CM Server installation includes multiple repositories, certain installation-wide elements are shared among all repositories.

To manage these installation-wide elements, each Documentum CM Server installation has a central repository called the global registry. You must designate a repository as a global registry, even if you plan to install one repository. The global registry is a repository like any other repository, except that all other repositories connect to it when they need an installation-wide element.

If you have a one-repository implementation, that repository is both a content repository and a global registry. If you have a Documentum CM Server implementation larger than a departmental one, consider creating a separate repository and designate that repository to be the global registry only.

2.1.1.3.1 Global registry user

A global registry user is created in all repositories, regardless of whether the repository is configured as a global registry.

- If you configure a repository as a global registry, provide the user name and password for the global registry user and the user state is set to Active. Make sure that you follow the password complexity rules. “[Password complexity rules](#)” on page 52 contains detailed information.
- If you do not configure a repository as a global registry, a global registry user is created with the default user name dm_bof_registry and the user state is set to Inactive. This user has read access to objects in a few folders in the System cabinet of the repository only.

2.1.2 Documentum CM Server installation models

Documentum CM Server and repositories can be installed and configured in many different ways to meet various content management requirements. Documentum CM Server installation supports the following types of configurations:

- Basic

In this model, Documentum CM Server, repository (including an RDBMS and a content storage), and connection broker are all installed on a single host. This is typically used in development and test environments.

- High-availability (HA)

In this model, multiple redundant Documentum CM Server instances and components are installed on a single host or multiple hosts and configured to eliminate single-point-of failure and achieve high-availability.

- Distributed

In this model, one or more repositories span multiple hosts and are configured to be accessed from multiple sites.

This document focuses on the basic installation model to describe the Documentum CM Server installation process.

2.1.2.1 Basic installation model

Components that are installed as a part of the Documentum CM Server installation process covered in this document are highlighted in yellow. Dotted lines indicate connections that are not persistent.

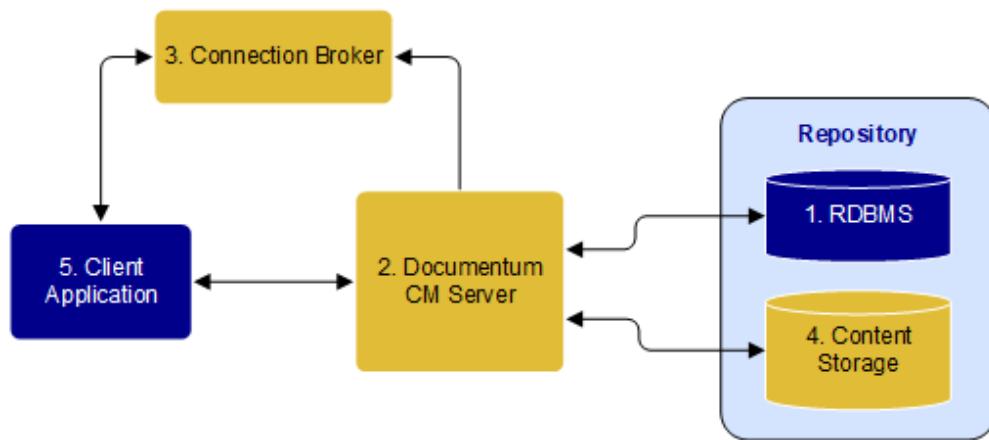


Figure 2-1: Basic installation model

1. Relational database management system (RDBMS), which stores content metadata.

This is a part of the repository and a prerequisite software component that must be installed prior to Documentum CM Server installation.

2. Documentum CM Server, which manages content stored in the repository.

3. Connection broker, which provides connection information to client applications.

4. Content storage, which stores native content files.

5. Documentum CM Server client application, which provides a user interface for accessing Documentum CM Server functionalities and managing repositories.

Documentum Administrator is the primary web-based administrative client tool for configuring and administrating Documentum CM Server and repositories.

Install the client application as a post-installation task.

2.1.3 Planning the installation

This section describes the installation decisions you should make before installing Documentum CM Server. These decisions will determine what prerequisite tasks you should perform, what choices to make and what information to provide during the installation process.

Use the Documentum CM Server installation checklist to plan your installation and record system information during installation and configuration. You can print out the checklist and keep it handy for reference.



Note: Information marked with an asterisk (*) next to it is prerequisite information you should obtain or plan for before installing Documentum CM Server.

Information	Note	Default value
Documentum CM Server:		
Documentum CM Server host FQDN *	for example, myhost.example.com	
Installation directory	<ul style="list-style-type: none"> Must not contain spaces On Linux: no default; for example, /usr/local/bin/Documentum 	C:\Documentum on Windows
Installation owner user name/password		
Application server administrator (admin) password	Make sure that you follow the password complexity rules. “Password complexity rules” on page 52 contains detailed information.	
Application server listening port *	A total of 20 ports, starting from the one you specify, will be used by the application server, and all of them must be available.	9080
Connection broker:		
Connection broker connection mode	Native, Secure, or Native and Secure	
Connection broker name		
Connection broker port	The port you specify and its subsequent port will be used by the connection broker and both must be available.	1489

Information	Note	Default value
Connection broker host name *		
Repository:		
Data directory	<ul style="list-style-type: none">• Local, SAN, or NAS• For SAN or NAS, enter the complete path including a shared device. For example: \ \\x.x.x.x\<folder1>\<folder2> where <x.x.x.x> is the IP address.	<\$DOCUMENTUM>/data
Share directory		<\$DOCUMENTUM>/share
Repository name	<ul style="list-style-type: none">• Up to 32 alphanumeric characters beginning with a letter.• The name docu reserved by the system.	

Information	Note	Default value
Repository ID	<ul style="list-style-type: none"> Any number from 1 to 16777215 and must not start with a zero (0). Must be unique on the network. The Repository ID is crucial for identifying all system related data, including configurations, objects, content, and so on and should be unique per repository. Each repository should have a unique Repository ID. These IDs are used to create file system paths in storage file paths to make sure that they do not conflict with other repositories. However, if you create two repositories to have same Repository ID and point to the same file path, they might overwrite each others' data. This is because, for a particular storage path, the system assumes that it has full access to that path (which includes the Repository ID). 	
Authentication domain (Windows) *	The default domain if the user does not specify a Windows domain when connecting to the repository	
Repository connection mode *	Native, Secure, or Native and Secure	
ODBC data source name (DSN)		
Database administrator name/password *	The account has privileges to create and delete databases and perform other database administrative tasks	

Information	Note	Default value
Repository owner name/ password	The database user account with the database owner role that has read and write access rights to the database	
Repository database name	The name of the database Documentum CM Server uses to store content metadata as well as system and repository information	
Repository data device file path	The configuration program automatically fills in this information	
Repository log device file path	The configuration program automatically fills in this information	
SMTP details *	SMTP server name or IP address, port, credentials, and SSL that Documentum CM Server uses for email notifications	
Installation owner email address *	Installation owner's email address that Documentum CM Server will use for email notifications	
Global registry name	The central repository to manage installation-wide elements	
Global registry user login name		dm_bof_registry
Global registry user password	Make sure that you follow the password complexity rules. "Password complexity rules" on page 52 contains detailed information.	

2.1.3.1 Location of the content storage area (data directory)

Documentum CM Server installation creates the default content storage area called the data directory for storing content files. After the initial installation, you can add additional storage areas and the business rules for distributing content among them by using Documentum Administrator.

The storage areas can be one of the following:

- A local file path on the Documentum CM Server host
- A file path on a remote host accessible by the Documentum CM Server
- A SAN or NAS file path
- A file path on other storage devices including retention type stores, such as Centera and Network Appliance SnapLock

Prepare a file path for the initial content storage area or the data directory that best suits your business needs. Make sure that the file path you select for the storage area has sufficient free space for the content files that will be added to it.

2.1.3.2 Ports to reserve for Documentum CM Server

Documentum CM Server and other components use a number of ports on the host:

- The application server listens on a port (9080 by default) for standard administration connections, and starting at this port, a total of 20 consecutive ports must be reserved for use by the application server.
- The connection broker requires two consecutive ports (1489 and 1490 by default) on which to listen: one for native connections and the other for secure connections (SSL).

On Linux, the port numbers can be any unused port numbers greater than 1024. Linux reserves port numbers up to 1024 for system use.

Identify available ports to be used by Documentum CM Server and its components. Make sure none of the reserved ports are being used for other purposes.

2.1.3.3 Connection modes to connect to connection broker and repository

There are three options you can select from for the Documentum CM Server client to connect to the connection broker and the repository:

- **Native**: The client connects through a non-TLS/SSL port.
- **Secure**: The client connects through a secure TLS/SSL port. The client and the connection broker or repository do not use TLS/SSL authentication to authenticate each other. However, the information exchanged between them is encrypted.
- **Native and Secure**: The connection broker or repository accepts both native and secure connection requests.



Note: (Only for **Secure** or **Native and Secure**) If client connects through TLS/SSL port in anonymous mode (without certificates), the TLS version used is 1.2. However, if client connects through TLS/SSL port in non-anonymous mode (with certificates), the TLS version used is 1.3.

Determine which connection mode you want to use for the connection broker and the repository respectively.

2.2 Prerequisites

Make sure that your system meets the system requirements and perform prerequisite tasks based on your installation plan before installing Documentum CM Server.

- Administrative privileges on the computer where you are installing Documentum CM Server.
- Working knowledge of Microsoft Windows and Linux, depending on which platform you select for your Documentum CM Server installation.



Notes

- OpenText recommends that you apply the latest operating system, .NET Framework, and related component patches. For example, operating system library updates, .NET Framework updates and so on.
- Make sure that you do not install Documentum CM Server on a day where the daylight savings clock shifts forward or backward especially in an European time zone.

2.2.1 Setting up system

The system requirements information is available in the product *Release Notes*. In addition to the system requirements, The `xterm` program may be installed in various file paths depending on the operating system and software packages installed.

2.2.2 Setting up database

A relational database management system is a prerequisite to installing Documentum CM Server. The database stores metadata of content objects in the Documentum CM Server repository.

The database documentation contains information on installing and configuring the RDBMS.

Make sure that the RDBMS meets the following general requirements:

- Database location

You can install the relational database either locally on the Documentum CM Server host or remotely on a separate host running any operating system supported by the database vendor. For example, you can install Documentum CM Server on a Windows host and use a database installed on a Linux host.

In a production environment, the database is almost always installed on a different host than the Documentum CM Server host for better performance.

- Database client

If you install the database on a separate host, also install the database client software on the Documentum CM Server host.

- For remote database installations, verify that you can connect to the database by using a database client from the system where you want to install Documentum CM Server.
- For local database installations on a Linux host, verify that the system path includes the installation directory for the database.

On a Windows host, the installer updates the system path automatically.

- Database code page

For new repositories, install the database by using the Unicode code page, which can accurately store characters from all supported languages.

- For new SQL Server repositories, Documentum CM Server uses only `nvarchar` and `nchar` types, which automatically use Unicode. If you need to determine the settings of an existing SQL Server database, use the system stored procedure `sp_helpsort` or view the properties of the particular database in Enterprise Manager.
- On Oracle, use UTF8 (this includes AL32UTF8 and AL16UTF8).



Note: Oracle has deprecated AL16UTF16. AL16UTF16 is supported for now, but it is recommended to migrate to AL32UTF8 or AL16UTF8 code page.

- Database locales

Typically, Documentum CM Server is installed on the English version of database. However, Documentum CM Server installation is also supported on localized databases if the database fulfills the following criteria:

- Database supports internationalization of locales (I18N)
- Database adheres to I18N standards
- Documentum CM Server installation is performed with UTF8 and case sensitive (SQL)

- Database administrator account

Regardless of the database you use with Documentum CM Server, obtain the database administrator's user name and password. You might need this information while configuring the repository.

- Database service (Windows)

If Documentum CM Server and the database are located on the same Windows host, make sure that the database service is set to start automatically.

Documentum CM Server installation sometimes requires a restart of the computer. After the restart, installation does not proceed correctly unless the database starts automatically.



Note: As an additional requirement, create and set the value of the `DM_DB_HA_ENABLE` environment variable to 1 for commit operations and HA to work properly.

2.2.2.1 Microsoft SQL Server database requirements

If you use the Microsoft SQL Server database with Documentum CM Server, make sure that it meets the following requirements:

- Use a full SQL Server installation on the host where SQL Server is installed. Install the SQL Server Management Studio and SQL Server client on the Documentum CM Server host, regardless of whether the database is local or remote. You need to install the required client packages from SQL Server installation for Documentum CM Server to work remotely with the SQL Server database.
- Use the Custom installation option so that you can set the database code page, case-sensitivity, and other options.
- Install the SQL Server instance in SQL Server.
- Install SQL Server for internationalization. The SQL Server documentation and Microsoft MSDN website contains more details.

- Make sure that the SQL Server sort order is set to **Dictionary**.
- Make sure that you configure the SQL Server with the following settings:
 - PARAMETERIZATION=FORCED
 - Allow Snapshot Isolation = ON
 - READ_COMMITTED_SNAPSHOT = ON

Make sure that the database statistics are up to date and optimal. Configure the database based on the results after profiling the database usage.

- On the Documentum CM Server host, create an ODBC data source for SQL Server.
 - Make sure that you select a 64-bit ODBC data source using the SQL Server driver only.



Note: SQL Server Native Client driver is not supported.

- You can also use the **User DSN** tab.
- Use the **With SQL Server authentication using a login ID and password entered by the user** option for SQL authentication method. If you select this option, enter the SQL Server administrator login ID and password. The repository owner (database user) does not need to have a Windows account.
- (Only for SQL Server 2019) Install SQL Server 2019 for Microsoft Windows latest cumulative update.
- Install the `sqlcmd` utility.



Caution

Install the database in case-sensitive mode with row-level locking enabled. If you installed SQL Server in case-insensitive mode, you need to reconfigure the database before you install or upgrade Documentum CM Server.

2.2.2.1.1 Supporting ODBC driver for SQL Server

1. Download and install the supported version of ODBC driver.
The product *Release Notes* contains the information about the supported versions.
2. Create new system DSN using the supported version of ODBC driver. For example, `sqldsn`.
3. When you create new repository, select the newly created DSN as the data source.
4. To upgrade to the supported version of ODBC driver for the existing repository, perform the following tasks:

- a. Stop the repository.
- b. Update the `database_conn` attribute in `server.ini` with the new DSN to indicate the server to point to the latest DSN.
- c. Restart the repository.



Note: When using ODBC driver 18, you must not uninstall ODBC driver 17.

2.2.2.2 Oracle database requirements

If you use the Oracle database with Documentum CM Server, make sure that it meets the following requirements:

- If Documentum CM Server resides on a host different from the Oracle database, you need to install the Oracle Database Client package on the Documentum CM Server host.
- On Linux, make sure that these environment variables are set in the environment of the installation owner:
 - `ORACLE_HOME`
 - `TNS_ADMIN`
 - `ORACLE_SID`

This environment variable points to the file path of the `tnsnames.ora` file. Documentum CM Server installer looks first for `TNS_ADMIN`, then for `ORACLE_HOME`, in order to locate the `tnsnames.ora` file.

- Configure the `tnsnames.ora` file.

Oracle database aliases (TNS aliases) are defined by entries in the `tnsnames.ora` file. Configure the `tnsnames.ora` file on the Documentum CM Server host. Use the Oracle SQL*Net configuration tool to create a database alias referring to the database instance you plan to use for Documentum CM Server. After you create the alias, test the alias by connecting to the Oracle database.

Entries in the `tnsnames.ora` file for the Oracle HTTP service and data expo service do not contain parameters for HOST, SID, and SERVICE. If the first entry in the `tnsnames.ora` file is for one of these services, Documentum CM Server installer is unable to parse the `tnsnames.ora` file and cannot connect to the database. Make sure that the first entry in the `tnsnames.ora` file is not for the Oracle HTTP service or data expo service.

To install Documentum CM Server with Oracle RAC, update the Oracle machine name or IP address in `tnsnames.ora` with SCAN name or SCAN IP address.

Make sure that you maintain the same Oracle database Service_Name when you migrate non-RAC Oracle database to Oracle RAC database.



Notes

- Documentum CM Server supports Oracle RAC only as a standalone support. Documentum CM Server does not support the failover or load balancing features of Oracle RAC.

- Documentum CM Server provides limited support for Oracle RAC. Documentum CM Server supports only those features that are natively supported per the Documentum CM Server design.

The `database_conn` key in the `server.ini` file must match the database entry in the `tnsnames.ora` file. If it does not, an error occurs. To fix, modify the `database_conn` key in the `server.ini` file and continue with the installation or upgrade.

- For Windows and Linux hosts, install the 64-bit Oracle Client package on the Documentum CM Server host and update the value of the shared library path environment variable (`LD_LIBRARY_PATH` for Linux) to include the library directory.
- Set the value of `Cursor_sharing` to Force.
- Enable the automatic memory management.
- Increase the size of the REDO log files from the default value of 50 MB and/or increase the number of REDO log files if the database runs into heavy updates.
- Make sure that the database statistics are up to date and optimal. Configure the database based on the results after profiling the database usage and by monitoring database reports such as AWR report, Oracle Alert log, Query execution plans, and so on. The following parameters can also impact the performance:
 - `Session_cached_cursors`
 - `Processes`
 - `MEMORY_TARGET`
 - `MEMORY_MAX_TARGET`
 - `Pga_aggregate_target`
 - `SGA_TARGET`
 - `SGA_MAX_SIZE`
- The `DM_CHECK_EMPTY_STRING_VALUE_IN_ORACLE` environment variable handles the '' (empty strings) in queries.
 - If `DM_CHECK_EMPTY_STRING_VALUE_IN_ORACLE = 1`, '' is treated as '' (a space) so that query results is same for all databases.
 - If a value for `DM_CHECK_EMPTY_STRING_VALUE_IN_ORACLE` is not set (default) or if `DM_CHECK_EMPTY_STRING_VALUE_IN_ORACLE = 0`, '' is treated as ''.
- The `DM_DEGREE_OF_PARALLELISM_ORACLE` environment variable is used to enable the parallel indexing on the Oracle database. The value of the variable must be set to an integer (for example, 4) which is used as Degree of Parallelism (DOP) while creating indexes in Oracle database.
- Set up networking parameters.
- Make sure that the Oracle listener is running on the Oracle host.

- Optionally, migrate an existing Oracle instance to AL32UTF8 or AL16UTF8 code page.
- Perform the following tasks to support Oracle multi-tenant container database:
 - Create a net service for the PDB service in `tnsnames.ora`.
 - Set `USE_SID_AS_SERVICE_LISTENER = ON` in `listener.ora` to allow using service name as SID.
- When PDB is configured, it will be in the mounted state. You must provide the read write permission.
- Perform the following task, if required, for the Oracle Instant client support: If you do not find any `bin` folder at the database home path where Oracle Instant client is installed, you must manually create a `bin` folder and then copy the `sqlplus` file inside that folder before starting the repository configuration with the Oracle Instant client driver.
- If you want to use Kerberos authentication (without using password) between Documentum CM Server and Oracle database, you must configure for Kerberos authentication on database. *Oracle* documentation contains detailed information.

For example:

1. Add the AD machine details to the database machine.
2. Create the `krb5.realms` file at `/etc/` with the following content:

```
. otxlab.net = CSDEV.LOCAL.OTXLAB.NET
```
3. Create the `krb5.conf` file at `/krb5/` with the following content:

```
[libdefaults]
default_realm = CSDEV.LOCAL
dns_lookup_kdc = true
dns_lookup_realm = true
dns_fallback=yes
allow_weak_crypto=true
[realms]
CSDEV.LOCAL = {
    kdc = AD.otxlab.net:88
}
[domain_realm]
. otxlab.net = CSDEV.LOCAL
otxlab.net = CSDEV.LOCAL
. OTXLAB.NET = CSDEV.LOCAL
OTXLAB.NET = CSDEV.LOCAL
```
4. Create a user in AD with name as fqdn of Oracle database host name and logon name as follows:

```
Oracle/cshostname.otxlab.net@CSDEV.LOCAL
```
5. Create a new AD user that can be used as a restricted user.
6. Create the keytab for host name principal (for example, `cshostname.otxlab.net`)
7. Copy the keytab to the shared file path (for example, `/krb5/`).

8. Create an empty file for storing the cache at the shared file path.
9. Update the `sqlnet.ora` with the attributes as specified in [step 5 in “Setting up restricted user for operation mode” on page 113](#).
10. Run the following command for Kerberos ticket generation and then provide the password:

```
okinit <restricted database user name created in AD>
```
11. Run the following command to view the generated ticket:

```
oklist -f
```
12. To verify if the session is created using Kerberos, use the following command:

```
sqlplus /@tnsalias
```

2.2.2.3 PostgreSQL database requirements

If you use the PostgreSQL database with Documentum CM Server, make sure that it meets the following requirements:

- Install the PostgreSQL server and client, and configure the `iptables`.
- Install the PostgreSQL server (as installer needs to run SQL scripts through `psql`) and ODBC driver in the Documentum CM Server host.
- On Windows:
 - Perform the following while creating an ODBC data source:
 - Create a driver with the **UNICODE** option.
 - Navigate to **System DSN > Datasource** and perform the following:
 - Remove **dd_** from **SysTable Prefixes**.
 - Disable the **LF<->CR/LF** conversion option.
 - Enable the **Updatable Cursors** option.
 - On Red Hat Enterprise Linux, Oracle Linux, and Ubuntu:
 - Set the following environment variables in the environment of the installation owner:
 - `POSTGRES_HOME`
 - `DATABASE_CONNECTION`
 - `POSTGRES_SID`
 - To allow remote connection:
 - Change the value of the method parameter to `trust` instead of `ident` in `/var/lib/pgsql/<supported PostgreSQL version>/data/pg_hba.conf`. For example:

- Use local for Linux domain socket connections only: local all all trust
 - IPv4 local connections: host all all 127.0.0.1/32 trust
 - IPv6 local connections: host all all ::1/128 trust
 - Change the value of listen_address to '*' in /var/lib/pgsql/<supported PostgreSQL version>/data/postgresql.conf.
- Configure ODBC.INI as follows:

```
[MyPostgres]
Description = PostgreSQL connection to MyPostgres
Driver = PostgreSQL
Database = postgres
Servername = localhost
UserName = postgres
Password = user password
Port = 5432
Protocol = <supported PostgreSQL version>
ReadOnly = No
RowVersioning = No
ShowSystemTables = No
ShowOIDColumn = No
FakeOIDIndex = No
UpdateableCursors = Yes
DEBUG = Yes
```

- Configure ODBCINST.INI as follows:

```
[PostgreSQL]
Description = ODBC for PostgreSQL
Driver = /usr/pgsql-<supported PostgreSQL version>/lib/psqlodbw.so
Driver64 = /usr/pgsql-<supported PostgreSQL version>/lib/psqlodbw.so
Setup64 = /usr/lib64/libodbcpsqls.so
FileUsage = 1
```

- To update the i_partition value of any object to another partitioned table range on a data partition enabled Oracle Linux/PostgreSQL environment, enable the constraint_exclusion parameter. Since PostgreSQL uses the concept of constraint exclusion to enable partition boundary checking, you must set the constraint_exclusion parameter in the postgresql.conf file. Open the postgresql.conf file, navigate to the QUERY TUNING section and remove the pound sign (#) for the constraint_exclusion entry. Set the constraint_exclusion parameter to <on>, and save the postgresql.conf file before exiting.



Notes

- Batching, specifically the delayed commit on batches, is not supported in PostgreSQL.
- For sorting issues with the ORDER BY clause, make sure that you add r_object_id in the select statement so that the Documentum CM Server explicitly adds ORDER BY r_object_id in the select statement.
- Japanese locale in Documentum CM Server for the PostgreSQL database is not supported.

2.2.3 Setting up required user accounts

Before installing Documentum CM Server, set up required user accounts in the host operating system and RDBMS.

2.2.3.1 Installation owner account

The installation owner account is an operating system account with appropriate permissions to install Documentum CM Server and create repositories. The installation owner account may be a local account on the Documentum CM Server host or a domain account in the domain where Documentum CM Server is installed. The account must be a member of the Administrators group on the local host.

Documentum CM Server runs under the installation owner account. The installation owner can perform all administrative and maintenance tasks associated with Documentum CM Server and repositories.

For security reasons:

- On Windows, it is recommended that the installation owner account is not the same account as the Windows Administrator.
- On Linux, it is recommended not to use the root account as the installation owner account.

You can create an operating system account to use exclusively for Documentum CM Server installation and repository configuration. You can use a single operating system account as installation owner for multiple Documentum CM Server installations on the network.



Notes

- The database user specified in the `server.ini` file as `database_owner` should have an account with the database.
- The installation owner account should have full access to `AEK.KEY` and `DBPASSWD.TXT`.

2.2.3.1.1 Installation owner account naming requirements

The installation owner user name must consist of letters, numbers, dashes (-) or underscores (_). The first character must be a letter. All characters must be ASCII characters.

The installation owner password consist of letters, numbers, dashes, underscores, or periods.

2.2.3.1.2 Required rights for the installation owner account

The installation owner account must have the following user rights on the host operating system:

- Act as part of the operating system
- Create a token object
- Increase quotas
- Log in as a service
- Log in locally
- Replace a process-level token

On Windows, these rights are automatically inherited with membership in the local Administrators group. Installer checks for these rights and grants them if necessary.

The installation owner must have Full Control permission on the directories into which Documentum CM Server is being installed, including data and share directories. The installation owner must also have write permission on the directory from which installer is run.

On Linux, the installation owner must have read, write, and execute permission on the </tmp> directory.

To support external password validation, set up a group account whose members are the installation owner, any other Documentum CM Server administrators, and repository owners. This will be the group that owns the external password validation program.

2.2.3.1.3 Installation owner email account and SMTP server information

Documentum CM Server requires the installation owner's email address and SMTP server information to send email notifications to the installation owner. During installation or upgrade, you need to provide this information and installer attempts to connect to the SMTP server. The SMTP server can be located on the Documentum CM Server host or another computer on the network.

Set up an email account for the installation owner and obtain the host name or IP address, port, credentials, and SSL communication requirement of the computer hosting the SMTP server Documentum CM Server can connect to.



Note: If you enable SSL communication, you must manually import the SSL server certificate to the default Java trust store (`cacerts`) or Foundation Java API trust store (`dgc.keystore`).

2.2.3.2 Repository owner account

The repository owner account is a database user account that Documentum CM Server uses to connect to the database. This account owns all objects in the database and gives Documentum CM Server access to the database tables underlying the repository. Each repository must have a unique repository owner.

You can create the repository owner (database user) account in one of these two ways:

- Allow Documentum CM Server to automatically create the repository owner account in the database when you create the repository during the installation process.

The configuration program automatically grants the account proper privileges.

- Manually create the repository owner account in the database prior to installing Documentum CM Server.

Make sure that the account has appropriate privileges to perform the following tasks:

- Connect to the database
- Create tables, views, and indexes in the database
- Insert records (rows) into the tables
- Drop tables, views, and indexes

Note the following requirements for different RDBMS:

- Microsoft SQL Server

The repository owner must be able to access `tempdb`, and if the account is created before installer is run, the user must own all tables and views. Make sure that the repository owner has the Create Any Database privilege.

- Oracle database

- If you create the account before Documentum CM Server installation, provide a value for the `select_catalog_role` parameter.
- The repository owner must have Connect and Resource privileges. The Resource privilege encompasses creating and maintaining database objects. The repository owner also must have permission to create any view, resource, and unlimited tablespace. The tablespace created by the repository owner for tables or indexes must be designated the default, while the standard Oracle temporary tablespace must be the default for any temporary tables that the repository owner creates. The name of the temporary tablespace must be valid for the Oracle configuration used. The default name is either `temporary_data` or `temp`, depending on the version of Oracle.

The repository must also have the Select Catalog Role privilege.



Note: If you have any concerns granting the Select Catalog Role privilege to the repository owner, create a new table in the current schema and retrieve all the information from the V\$SESSION table into the new table. Then, set the DM_ORACLE_SESSION_TABLE environment variable to the new table name.

To use Microsoft Cluster Services, the repository owner must have an account in the domain in which you install the repository.

2.2.3.3 Repository user accounts

Repository users are the end users in the repository.

- On Windows, if the default user authentication is used, each user must have a Windows account in the domain where Documentum CM Server is installed. If inline password authentication is used, this is not a requirement.
- On Linux, if the default user authentication is used, each user must have an operating system account in the domain where Documentum CM Server is installed. If inline password authentication is used, this is not a requirement.

2.2.4 Configuring heap memory for Tomcat

- On Windows, add one of the following information in the startup script:

```
set CATALINA_OPTS=-Xms1024m -Xmx2048m;
```

Or

```
set JAVA_OPTS=-Xms2048m -Xmx2048m;
```

This increases the heap size of Tomcat to 2 GB for both the startup and maximum memory.

- On Linux, add one of the following information in the startup script:

```
export CATALINA_OPTS=-Xms1023m -Xmx2048m;
```

where *-Xms* is the minimum or initial size of your heap and *-Xmx* is the maximum size.

Or

```
export JAVA_OPTS="-Xms1024m -Xmx1024m"
```

2.2.5 Pre-installation tasks on Windows

- Set the date and time formats to a four-digit year (yyyy) date in the Windows regional settings.
- Disable the user access control (UAC).



Note: If you want to enable UAC for using the dm_assume_user utility, make sure the following: System administrator must provide the Replace a process level token privilege to the OpenText Documentum CM service user (mostly the installation owner). Also, any OpenText Documentum CM user must have the *Allow logon locally* privilege to change their password.

- Disable the IP Helper service from the Windows Services console and restart the machine. This method disables the Teredo Tunneling Pseudo-Interface.
- Enable the Computer Browser service (optional).
- In all the supported operating systems, install the latest version of Microsoft Visual C++ 2013, 2019, and 2022 Redistributable (64-bit) packages before creating a repository. This provides the correct operating system runtime libraries for the Documentum CM Server and other utilities.
- Install the Microsoft security updates released in June, 2014 onwards to avoid vulnerabilities on the Windows hosts.

2.2.6 Pre-installation tasks on Linux

In addition to system requirements, in Linux hosts, you need XWindows. XWindows must be installed on the Linux hosts to run the graphical installation program.

To enable Federal Information Processing Standard (FIPS) on Linux, you must enable the Federal Information Processing Standard settings on the operating system. Refer to the operating system vendor documentation for instructions.

2.2.6.1 Red Hat Enterprise Linux

The pre-installation tasks in this section is applicable for all Red Hat Enterprise Linux based environments.

Before installing the Documentum CM Server, make sure that you have done the following configuration:

- To make sure that the repository configuration does not fail with `libsasl2.so.2: cannot open shared object file` error on Red Hat Enterprise Linux 7.x/8.x, log into the root account and run the following command:

```
ln -s /usr/lib64/libsasl2.so.3.0.0 /usr/lib64/libsasl2.so.2
```

- To make sure that the repository, job, IAPI, and IDQL logs do not display the `misc.c si_init() setrlimit failed errno=22 Invalid argument` error on Red Hat Enterprise Linux caused by the operating system resource constraint, perform the following tasks in the `limits.conf` file in `/etc/security/`:

- Remove the soft core 0 and hard core 0 lines.
- Add the <Documentum CM Server installation owner> - core -1 line.
- Restart the OpenText Documentum CM processes and reboot the system.
- To make sure that the dm_assume_user utility works properly in Red Hat Enterprise Linux 8.x, add the following line in the \$DOCUMENTUM/dba/dm_RunExternalCommand.sh file:

```
#!/bin/sh
umask 0022
touch $4
...
```

- Log in to your machine as a root user and add the following line in the end of the /etc/sysctl.conf file:

```
kernel.shmmni = 6000
```

Then, run the following command as a root user to reflect the changes:

```
sysctl -p
```

2.2.6.2 SUSE Enterprise Linux

Before installing the Documentum CM Server, make sure that you have done the following configuration:

To make sure that the dmdb test does not fail during the loading of the shared libraries of libssl.so.10, while configuring the repository, perform the following steps:

```
root:~ # cd /lib64
root:/lib64 # ln -s libcrypto.so.1.0.0 libcrypto.so.10
root:/ # cd /usr/lib64
root:/usr/lib64 # ln -s libsasl2.so.3 libsasl2.so.2
root:/usr/lib64 # ln -s libssl3.so libssl.so.10
```

2.2.6.3 Setting the required environment variables

In the installation owner's environment, set environment variables that identify the directories into which Documentum CM Server will be installed. The variables must be set in the installation owner's environment.

You can create the Documentum CM Server installation directory before installing the server, or you can allow the installer to create the directory for you during installation. If you allow the installer to create the directory, make sure that the directory name you provide during the installation matches the one specified in the environment variable.

Symbolic links (symlinks) are not supported. Do not use them in the installation directory or any other environment variable used by Documentum CM Server. This restriction also applies to environment variables used to specify the database location. For example, if you use Oracle, the ORACLE_HOME environment variable cannot use a symbolic link. However, the symbolic links for Java upgrade is supported.

Environment variables and the installation directory must contain only ASCII characters. The name of the installation directory must not contain spaces.

On Linux, you need to set certain environment variables in the installation owner's environment.

You must set the \$DOCUMENTUM variable first for installer to run successfully.

- \$DOCUMENTUM

The Documentum CM Server installation directory. The installation owner must have read, write, and execute permission on the \$DOCUMENTUM directory and its subdirectories. There is no default installation directory on Linux, and \$DOCUMENTUM cannot be mounted with the nosuid option.

The Documentum CM Server configuration program (`dm_launch_server_config_program.sh`) automatically sets all required environment variables except for those required by each database. If you do not use the Documentum CM Server configuration program, you need to manually set all environment variables.

You can set all of the following variables, except LC_ALL and DISPLAY, by sourcing \$DOCUMENTUM/product/<version_number>/bin/dm_set_server_env.sh or \$DOCUMENTUM/product/<version_number>/bin/dm_set_server_env.csh. Set the variables LC_ALL and DISPLAY in the installation owner's .cshrc file (C shell) or .profile file (Bourne or KornShell). Alternatively, set the variables in a file called by the .cshrc file or the .profile file or in other ways permitted by Linux.

- \$DM_HOME

Must be \$DOCUMENTUM/product/<version_number>, where <version_number> is the version of Documentum CM Server.

- Set the DM_JMS_HOME environment variable manually in Red Hat Enterprise Linux operating system. For example, \$DOCUMENTUM/tomcat.

- LD_LIBRARY_PATH on Linux

Add the directory containing the client library to the appropriate environment variable of the installation owner. You must set this environment variable before configuring or starting Documentum CM Server, as well as running IAPI or IDQL.

- DISPLAY

Controls the display. Set the value to <host/ip>:0.0.

- LC_ALL

Set the value to C.



Caution

If this value is not set correctly, the Java Method Server will fail.

- JAVA_TOOL_OPTIONS

For the JDK 17 support, set the value of the JAVA_TOOL_OPTIONS environment variable in the .cshrc file in Linux as follows:

```
setenv JAVA_TOOL_OPTIONS "-Djdk.util.zip.disableZip64ExtraFieldValidation=true"
```

2.2.6.4 Setting up the services file

The services file must contain two entries for each repository running on a host. Manually create the service name entries in the services file before you install the Documentum CM Server.

The repository does not have default service names or default port numbers. The service name you put in the services file must be the same name you will provide during repository configuration, which is then used to create the server.ini file. The service name for the repository can be the same as the repository name, but this is not required.

The services file must include entries that designate two consecutive port numbers: one for native connections, and the other for secure (SSL) connections. Append _s to the name of the repository service for the secure connections.

Create two network service entries in the system's service table using the following format:

```
<service_name> <port_number>/tcp #comments here  
<service_name_s> <port_number>/tcp #comments here
```

In the following repository service entries example, repository service is named repo1:

```
dm_repo1  
47625/tcp # <version_number> Repository native connection  
dm_repo1_s  
47626/tcp # <version_number> Repository secure connection
```



Notes

- Even if you are not using secure (SSL) connections, two consecutive port numbers are still required by Documentum CM Server.
- If correct services file entries are not present during installation, the installation will fail.

If you have multiple repositories on a single host, create a services file entry for each repository. Make sure that the repositories use different names and port numbers.

2.2.6.5 Enabling random generator on Linux

A connection request through SSL waits for a random number to be created and a delay is noticed. To avoid the delay, you can enable or disable the random generator on the Linux machine. Perform the following boot safe (inittab) on all Documentum CM Server machines:

Start the random generator as root: /sbin/rngd -b -r /dev/urandom -o /dev/random

2.2.6.6 Installing the script utilities

Before installing the Documentum CM Server, make sure that you have completed the following tasks:

- Install the Expect (version 5.45.4 or earlier) script utility.
- Install the TCL (version 8.6.8 or earlier) script utility.

2.2.7 Installing JDK

Download and install the supported version of Oracle JDK/OpenJDK from the Oracle JDK/OpenJDK website. The *Oracle JDK* and *OpenJDK* documentation contains detailed information.

Set the `JAVA_HOME` environment variable to point to the installed Oracle JDK/OpenJDK version. In addition, set the `PATH` environment variable to `JAVA_HOME\bin`.

To run Java methods with the supported version of Oracle JDK/OpenJDK, do the following:

- On Windows, set the `JDK_JAVA_OPTIONS` environment variable to the following value:

```
-Djava.locale.providers=COMPAT,SPI
--add-opens=java.base/java.lang=ALL-UNNAMED
--add-opens=java.base/java.lang.invoke=ALL-UNNAMED
--add-exports=java.base/sun.security.provider=ALL-UNNAMED
--add-exports=java.base/sun.security.pkcs=ALL-UNNAMED
--add-exports=java.base/sun.security.x509=ALL-UNNAMED
--add-exports=java.base/sun.security.util=ALL-UNNAMED
--add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED
```

- On Linux, set the `JAVA_TOOL_OPTIONS` environment variable to the following value:

```
-Djdk.util.zip.disableZip64ExtraFieldValidation=true
-Djava.locale.providers=COMPAT,SPI
--add-opens=java.base/java.lang=ALL-UNNAMED
--add-opens=java.base/java.lang.invoke=ALL-UNNAMED
--add-exports=java.base/sun.security.provider=ALL-UNNAMED
--add-exports=java.base/sun.security.pkcs=ALL-UNNAMED
--add-exports=java.base/sun.security.x509=ALL-UNNAMED
--add-exports=java.base/sun.security.util=ALL-UNNAMED
--add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED
```

2.2.7.1 Using SSL communication

OpenText strongly recommends to use certificated-based SSL communication. “[Configuring for certificate-based SSL communication](#)” on page 40 contains the information.

For SSL communication, by default, Foundation Java API searches the value of `dfc.security.ssl.truststore` (property for certificate) in `dfc.properties`. If found, Foundation Java API uses the certificated-based SSL communication. Otherwise, Foundation Java API defaults to the cipher-based anonymous SSL supported by Java.

Oracle JDK/OpenJDK 11.0.2 and later do not support anonymous ciphers for SSL or TLS communication and results in the following exception:

```
DfDocbrokerException:: THREAD: main; MSG: [DFC_DOCBROKER_REQUEST_FAILED]
Request to Docbroker "XX.XXX.XX.XX:1489" failed; ERRORCODE: ff; NEXT: null at
com.documentum.fc.client.DfDocbrokerException.
newRequestFailedException(DfDocbrokerException.java:13) at com.documentum.
fc.client.impl.docbroker.ServerMapBuilder.getMap(ServerMapBuilder.
java:72)...
Caused by: DfIOException:: THREAD: main; MSG: [DM_SESSION_E_RPC_ERROR]error:
"Server communication failure"; ERRORCODE: 100; NEXT: null at com.
documentum.fc.client.DfIOException.
newCommunicationFailureException(DfIOException.java:16) at com.documentum.
fc.client.impl.connection.netwise.AbstractNetwiseRpcClient.
sendMessage(AbstractNetwiseRpcClient.java:216)
...
Caused by: javax.net.ssl.SSLHandshakeException: Received fatal alert:
handshake_failure at java.base/sun.security.ssl.Alert.
createSSLEException(Alert.java:128) at java.base/sun.security.ssl.Alert.
createSSLEException(Alert.java:117)
```

However, if you want to use anonymous SSL communication, perform the following tasks:

1. In the supported version of Oracle JDK/OpenJDK, open the `java.security` file from `<JAVA_HOME>\conf\security`.
2. Find the line starting with `jdk.tls.disabledAlgorithms`.
3. Remove `anon` from the list of disabled algorithms.
4. (Only for Red Hat OpenJDK) In the supported version of Red Hat OpenJDK, perform the following tasks:
 - a. Open the `java.config` file from `/etc/crypto-policies/back-ends`.
 - b. Find the line starting with `jdk.tls.disabledAlgorithms`.
 - c. Remove `DH_anon` from the list of disabled algorithms.
5. Save your changes.



Note: For anonymous Java Method Server SSL communication, set `TLS_DH_anon_WITH_AES_128_CBC_SHA` as value to the `enabled-cipher-suites` parameter in `server.xml` located in `%DM_JMS_HOME%\conf`.

2.2.8 Configuring internationalization settings

Documentum CM Server runs in the UTF-8 code page. Perform the following tasks before Documentum CM Server installation:

- Install the server host code page.
- Set the code page in the database.
- Set the server host locale.

The server host locale and the server code page need not be same. For example, if the host code page is set to `ISO-8859_1`, the host locale would typically be set to a European language (English, French, German, Italian, Portuguese, or Spanish). If the host locale is set to French, a client that connects to the Documentum CM Server without specifying a client locale is served French data dictionary labels.

If the host locale is one of the languages supported by OpenText Documentum CM, the data dictionary information for that locale is loaded. Otherwise, the server defaults to loading the English data dictionary information. You can load additional sets of data dictionary information by modifying the `data_dictionary.ini` file. OpenText Documentum CM only supports the languages that are shipped with Documentum CM Server.

- On Windows hosts, the host locale is set in the Regional Settings dialog box.
- On Linux hosts, the host locale is set with the `LANG` environment variable.

Documentum CM Server can be installed on computers that run the following operating system code pages:

- For U.S. and Western European sites, `ISO-8859_1` (Latin-1)
- For Korean sites, `EUC-KR`
- For Japanese sites that use Linux, `EUC-JP`
- For Japanese sites that use Windows, `Shift_JIS`
- For Chinese sites with locale `zh_ms936`
- For Russian with locale `ru_Windows-1251`

For more information about the locale-based configuration settings for the data dictionary files installed with Documentum CM Server, see *OpenText Documentum Content Management - Server Administration and Configuration Guide (EDCCS250400-AGD)*.

2.2.9 Installing MailApp DAR

The MailApp DAR file on the Documentum CM Server is required for any clients to use the changed email management features introduced in Webtop 6.8. By default, this DAR is installed by the Documentum CM Server configuration program.



Note: If you import a file (object) with the .msg format to the repository and then try to parse the .msg file with `shouldParseMsgFile` set to true, then aspects are attached to the object. After the aspects are attached, the checkout, checkin, and cancel checkout content transfer operations are not allowed on the object. This is the default behavior.

2.2.10 Configuring Documentum CM Server to use HashiCorp Vault

To configure Documentum CM Server to use HashiCorp Vault, store all secrets in the HashiCorp Vault server. For more information about storing secrets in the HashiCorp Vault server, see *HashiCorp* documentation.

After storing the secrets in the HashiCorp Vault server, the Documentum CM Server installer updates the secret and key name information in the `<secret_name>/<key_name>` format.

The following table lists the secret and key names:

Table 2-1: Secret and key name

Secret name	Key name	Description
INSTALL_OWNER_PASSWORD	<code><hostname></code>	Installation owner password. For example, <code>INSTALL_OWNER_PASSWORD/winora.net</code> .
DCTM_SERVER_JMX_PASSWORD	<code><hostname></code>	Tomcat administrator user password. For example, <code>DCTM_SERVER_JMX_PASSWORD/winora.net</code> .
DOCBROKER_CERT_PASSWORD	<code><hostname></code>	Connection broker password. For example, <code>DOCBROKER_CERT_PASSWORD/winora.net</code> .
DOCBASE_CERT_PASSWORD	<code><repository name></code>	Repository certificate password. For example, <code>DOCBASE_CERT_PASSWORD/testrepo</code> .

Secret name	Key name	Description
DFC_SECURITY_SSL_TRUSTSTORE_PASSWORD	<hostname>	<p>Foundation Java API trust store password.</p> <p>For example, DFC_SECURITY_SSL_TRUSTSTORE_PASSWORD/winora.net.</p>
AEK_PASSWORD	<AEK key name>	<p>AEK key password.</p> <p>For example, AEK_PASSWORD/CSaek.</p>
DATABASE_PASSWORD	<database service name>_<username>	<p>Database administrator user password.</p> <p>For example, DATABASE_PASSWORD/pgsql_testuser.</p>
DOCBASE_PASSWORD	<repository name>	<p>Repository password.</p> <p>For example, DOCBASE_PASSWORD/testrepo.</p>
DOCBASE_SMTP_PASSWORD	<repository name>	<p>SMTP user password.</p> <p>For example, DOCBASE_SMTP_PASSWORD/testrepo.</p>
XML_STORE_PASSWORD	<repository name>	<p>OpenText Documentum Content Management (CM) XML Store administrator password.</p> <p>For example, XML_STORE_PASSWORD/testrepo.</p>
DFC_GLOBALREGISTRY_PASSWORD	<global repository name>	<p>Global registry password.</p> <p>For example, DFC_GLOBALREGISTRY_PASSWORD/globalrepo.</p>
PRIMARY_DOCBASE_INSTALL_OWNER_PASSWORD	<primary repository name>	<p>Primary repository installation owner password.</p> <p>For example, PRIMARY_DOCBASE_INSTALL_OWNER_PASSWORD/primaryrepo.</p>
GCP_CREDENTIALS	<repository name>_<filestore name>	<p>Google Cloud storage type service account credentials.</p> <p>For example, GCP_CREDENTIALS/gcprepo_gcpstorename.</p>

Secret name	Key name	Description
MEMBER_REPO_SUPER_USER_PASSWORD	<repository name>	Member repository password. For example, MEMBER_REPO_SUPER_USER_PASSWORD/testrepo.
CLIENTS_PREFERENCE_PASSWORD	<repository name>	Preference password. For example, CLIENTS_PREFERENCE_PASSWORD/clientrepo.
CLIENTS_PRESET_PASSWORD	<repository name>	Preset password. For example, CLIENTS_PRESET_PASSWORD/clientrepo.

2.2.11 Preparing installation package

Download the Documentum CM Server software for your operating system and database.

- On Windows, expand the compressed archive by double-clicking the file.
- On Linux, expand the compressed archive by typing:

```
% tar -xvf <filename>
```

Make sure that you have execute permission on the serverSetup.bin file. If not, add execute permission to the file by running this command:

```
chmod +x serverSetup.bin
```

2.3 Configuring for certificate-based SSL communication

This section describes the information on certificate-based SSL communication.

2.3.1 Connection modes

Documentum CM Server and connection broker support the following connectivity modes:

Mode	Description
Native	Connections are in the raw RPC format without any encryption.

Mode	Description
Secure	SSL mode is: <ul style="list-style-type: none"> • used with anonymous ciphers (ADH:DEFAULT) • used with certificates
Dual (Native & Secure)	Connections are both in the raw RPC format without any encryption and SSL mode is used with anonymous ciphers (ADH:DEFAULT) and certificates.

OpenText strongly recommends that you configure Documentum CM Server, connection broker, and Foundation Java API in the Secure mode with certificated-based SSL.

In addition to the existing connectivity options, a new connectivity mode is available. In this mode, certifications are used to verify SSL servers.

 **Note:** Certificated-based SSL and non-anonymous mode SSL are synonymous.

Foundation Java API supports the following connectivity modes:

Mode	Description
Native	Connections are in the raw RPC format without any encryption
Secure	SSL mode is used with anonymous ciphers (ADH:DEFAULT)
Try_native_first	Native mode is attempted first, falling back to secure
Try_secure_first	Secure mode is attempted first, falling back to native

In the OOTB setup, the secure mode will continue to use anonymous mode by default. In the anonymous SSL mode, certificates are not used. In addition, both end parties are trusted and not verified. You must configure Documentum CM Server, connection broker, and Foundation Java API to use certificated-based SSL.

Note the following details about Certificated-based SSL:

- Clients verify the servers. For example, Foundation Java API verifies the Documentum CM Server during connection handshake.
- Clients will connect only to servers that are in the trust store of the SSL client.

 **Note:** A trust store is a file containing certificates from trusted servers.

- Only SSL server verification is supported. No verification is provided to the SSL clients.

2.3.2 Pre-installation requirements and tasks

Certificates are not provided. You must procure or generate your certificates and manually configure the components to use the non-anonymous/certificated-based SSL mode.

You can select to maintain your existing key. If you need to use the AEK key before creating the repository, create a key using the dm_crypto_create tool.

To create password-based AEK, use the following command format:

```
dm_crypto_create -location <name of AEK key> -passphrase <password>
```

To create HashiCorp Vault-based AEK, use the following command format:

```
dm_crypto_create -location <$DM_HOME/dba/secure/name of AEK key> -secret_id  
<secret_name>/<key_name> -dsis_url <DSIS URL> -dsis_daemon_token <DSIS authentication  
token> -vault
```

Example:

```
dm_crypto_create -location /home/test/dctm/dba/secure/CSaek -secret_id AEK_PASSWORD/  
CSaek -dsis_url http://localhost:8200/dsis -dsis_daemon_token 123456 -vault
```

Make sure that you follow the password complexity rules. “[Password complexity rules](#)” on page 52 contains detailed information.

2.3.2.1 Certificates

You must purchase certificates from a Certificate Authority (CA).

Perform the following steps to procure a certificate:

1. Provide details of the certificate to the CA in the Certificate signing request (CSR) format. The CA obtains CSR from user and signs it and provides the signed certificate to the user.
2. You will receive certificate signed by a CA and a private key with CA certificate chain.
3. Depending on the CA, the certificate and chain can be in the PKCS#12 or plain DER or PEM format. PKCS#12 is widely-used specialized keystore format.
 - PKCS#12 format is provided with a password
 - The private key is provided separately for PEM/DER certificates

You can also generate certificates using tools. Such certificates are known as self-signed certificates and are generally used for testing.

There is no restriction or limitation on the certificate type, private key type/length, or certificate chain length.

2.3.2.2 Building a keystore

The keystore must be in the PKCS#12 format.

If certificates are in the PKCS#12 format, no additional configuration is required. Proceed to configure the connection broker, server, and Foundation Java API.

If the certificate and private key are provided in PEM or DER format, you must use tools to import to the PKCS#12 format and then perform the configuration.

You can use OpenSSL to generate certificates and keys, convert among the different representations, convert among the key/certificate/store formats, dump key/certificate/store details/contents, debug connections, and so on. The *OpenSSL* documentation contains the information.

You can use Keytool to generate certificates and keys per Java standards for Java-based applications. The *Keytool* documentation contains the information.

2.3.2.3 Preparing keystore password

The PKCS#12 keystore is provided with a password for security. If the CA has provided the certificate in this format, the password will also be available.

Password formats:

- The password can be provided in a plain text format.
- The password can also be provided in an encrypted format using the `dm_encrypt_password` utility that is packaged along with Documentum CM Server installation. The `dm_encrypt_password` utility uses `aekey.key` in `$DOCUMENTUM/dba/secure` to encrypt the password.
- For password-based AEK keys, use the following command format:

```
dm_encrypt_password -encrypt <password> -file cert.pwd -keyname <name of AEK key> -passphrase <passphrase>
```

where `cert.pwd` is the file that contains the encrypted password for the connection broker/Documentum CM Server certificate.



Note: This is similar to how the Database password is stored on the Documentum CM Server machine.

If the password is encrypted, it will be specific to the Documentum CM Server where it was encrypted. It cannot be copied to any other Documentum CM Server that has a different `aekey.key`.



Important

If the installation of Documentum CM Server is configured with HashiCorp Vault, update the contents of the `broker.pwd` and `server.pwd` in the `<secret_name>/<key_name>` format instead of encrypted passwords. For example:

In the `broker.pwd` file:

```
DOCBROKER_CERT_PASSWORD/<host name of connection broker>
```

In the `server.pwd` file:

```
DOCBASE_CERT_PASSWORD/<repository name>
```

2.3.2.4 Building a trust store

Considerations for building a trust store:

- For Documentum CM Server, the trust store must be in the PKCS#7 binary format. PKCS#7 is a widely-used specialized trust store format.
- For Foundation Java API, the trust store must be in the JKS format. JKS is a widely-used specialized keystore format standardized by Java.



Note: The keystore is a file containing the certificate and private key.

The trust store contains all certificates that the SSL client can trust. The certificate list includes the list of all root and intermediate CAs. It is optional to store the leaf certificates (connection broker certificate) if you obtained it from CA. The certificate chain is adequate to validate the leaf certificate.

If the certificates are self-signed and do not include a certificate chain, then all the certificates must be stored in the trust store. However, if the certificates are self-signed and include a certificate chain, then all certificates, except the leaf certificate, must be stored in the trust store.

For password-based AEK key with SSL, you must manually import the connection broker and Documentum CM Server certificates to the default Java truststore (cacerts) using the following keytool commands:

```
keytool -importcert -trustcacerts -cacerts -file broker crt.pem -alias broker -storepass  
changeit -noprompt  
keytool -importcert -trustcacerts -cacerts -file server crt.pem -alias server -storepass  
changeit -noprompt{noformat}
```



Note: Before running the keytool command, make sure that the install owner has both the read and write permissions on the Java truststore (cacerts) file.

2.3.2.5 Location of the keystore file, keystore password file, and trust store file

The connection broker keystore, connection broker keystore password file, Documentum CM Server keystore, Documentum CM Server keystore password file, and the Documentum CM Server trust store must reside in the \$DOCUMENTUM/dba/secure directory.

The Foundation Java API trust store can be stored anywhere on the local machine which is accessible to Foundation Java API.

2.3.3 Configuration

Configuration is a one-time activity. Subsequent patches/upgrades will not affect the configuration.

2.3.3.1 Configuring connection broker

The connection broker works as a SSL server. It requires details of the keystore and the keystore password.

Connection broker settings are stored in the docbroker.ini file as follows:

Parameter name	Description
crypto_keyname = <CSaek>	It is the AEK key name that is used to encrypt the password in the broker.pwd file.
keystore_file = <filename>	Keystore containing the connection broker certificate and private key
keystore_pwd_file = <filename>	File containing the plain text or encrypted keystore password
Cipherlist = <list>	<p>List of ciphers separated by ":". OpenSSL library supports these ciphers.</p> <ul style="list-style-type: none"> When Foundation Java API (client) and Documentum CM Server uses bundled Java: AES128-SHA:AES256-GCM-SHA384:AES256-SHA:AES128-GCM-SHA256:DES-CBC3-SHA:TLS_AES_256_GCM_SHA384:TLS_AES_128_GCM_SHA256 When Foundation Java API (client) and Documentum CM Server uses different Java: AES128-SHA:AES256-SHA:DES-CBC3-SHA
isDSISEnabled	Indicates if the passwords and secrets are stored in the vault type server. Valid values are T and F.

Parameter name	Description
dsis_daemon_token	DSIS authentication token. For more information about DSIS authentication token, see “Configuring Documentum Secret Integration Service” on page 49.
dsis_url	DSIS URL. For more information about DSIS URL, see “Configuring Documentum Secret Integration Service” on page 49.



Notes

- The ciphers that work depends on the Java version and it's capabilities.
- The `isDSISEnabled`, `dsis_daemon_token`, and `dsis_url` parameters are applicable only if you want to use a vault type.
- For Documentum CM Server HA, you should use the same certificates for Documentum CM Server and connection broker from Documentum CM Server 1. Copy the certificates from primary Documentum CM Server to secondary Documentum CM Server.

2.3.3.2 Configuring Server

Documentum CM Server works both as an SSL server and an SSL client. It requires details about the keystore, keystore password, and trust store.

Documentum CM Server settings are stored in the `server.ini` file as follows:

Parameter name	Description
<code>keystore_file = <filename></code>	Keystore containing the connection broker certificate and private key
<code>keystore_pwd_file = <filename></code>	File containing the plain text or encrypted keystore password
<code>truststore_file = <filename></code>	File containing the trusted connection broker certificates
<code>Cipherlist = <list></code>	List of ciphers
<code>crypto_keystore = <Local or Remote_Vault></code>	Local is used for password-based AEK key and Remote_Vault is used for HashiCorp Vault-based AEK key.
<code>isDSISEnabled</code>	Indicates if the passwords and secrets are stored in the vault type server. Valid values are T and F.
<code>dsis_daemon_token</code>	DSIS authentication token. For more information about DSIS authentication token, see “Configuring Documentum Secret Integration Service” on page 49.

Parameter name	Description
dsis_url	DSIS URL. For more information about DSIS URL, see “Configuring Documentum Secret Integration Service” on page 49.

 **Note:** The `isDSISEnabled`, `dsis_daemon_token`, and `dsis_url` parameters are applicable only if you want to use a vault type.

For the complete list of the `server.ini` keys, see *OpenText Documentum Content Management - Server Administration and Configuration Guide (EDCCS250400-AGD)*.

 **Note:** Ensure that the PKCS#12 certificate file is created using the Federal Information Processing Standard (FIPS) compliant `-descert:` and `-nomac` options.

2.3.3.3 Configuring Foundation Java API

Foundation Java API works as an SSL client. It requires details about the trust store.

Foundation Java API settings are stored in the `dfc.properties` file as follows:

Parameter name	Description
<code>dfc.security.ssl.use_existing_truststore = <boolean:on off></code>	To specify whether Foundation Java API uses the default Java trust store or a specified trust store. <ul style="list-style-type: none"> • On/True: Use Java’s trust store. • Off/False: Use the provided trust store.
<code>dfc.security.ssl.truststore = <filename path></code>	File containing trusted connection broker and Documentum CM Server certificates.
<code>dfc.security.ssl.truststore_password = <password></code>	To specify the plain text or encrypted trust store password.
<code>dfc.dsis.enabled=true</code>	Indicates if a vault type is enabled.
<code>dfc.dsis.daemon.url=http://localhost:8200/dsis</code>	DSIS URL.
<code>dfc.dsis.daemon.token=<token generated for the DSIS daemon agent></code>	Token generated for the DSIS daemon agent.

 **Note:** The `dfc.dsis.enabled`, `dfc.dsis.daemon.url`, and `dfc.dsis.daemon.token` parameters are applicable only if you want to use a vault type.

Sample:

```
dfc.security.ssl.use_existing_truststore = FALSE
dfc.security.ssl.truststore = <full path of Foundation Java API trust store including
the name of Foundation Java API trust store (dfc.keystore)>
dfc.security.ssl.truststore_password = password
dfc.dsis.enabled=true
```

```
dfc.dsds.daemon.url=http://localhost:8200/dsis  
dfc.dsds.daemon.token=8045879651587
```

When the Documentum CM Server and connection broker security certificates are provided through `dfc.security.ssl.truststore`, it does not work and results in SSL communication failure in the following two scenarios:

- When the Java socket factory is created before Foundation Java API initialization
- When Foundation Java API trust store overrides the default Java cacerts (keystore)

To resolve the SSL communication failure, you must import the certificates directly in Java using the following steps:

1. Copy the security certificates (.DER files) of both Documentum CM Server and connection broker to the Foundation Java API machine. For example, copy `servercrt.der` and `brokercrt.der` from <location where .DER files of Documentum CM Server and connection broker are available> to the Foundation Java API machine.
2. Import the security certificates of both Documentum CM Server and connection broker in Java used by Foundation Java API based clients.

```
keytool -import -trustcacerts -file <dctmserver_securityfile(.der)_location>  
-keystore <JAVA_HOME>\lib\security\cacerts -alias server  
  
keytool -import -trustcacerts -file <connectionbroker_securityfile(.der)_location>  
-keystore <JAVA_HOME>\lib\security\cacerts -alias broker
```



Notes

- For importing other formats of security certificates, see *Keytool* documentation.
- If the Foundation Java API based client application wants to connect to multiple Documentum CM Servers which are running in secure mode using certificates, then you must import all those Documentum CM Server and connection broker security certificates explicitly in the Java trust store.

2.3.4 Installer support

During the configuration of connection broker and Documentum CM Server, the installer will prompt you for Certificate details which is used for configuration.

2.3.5 Compatibility

All the components in a setup must use a consistent secure mode. For example, the Documentum CM Server, connection broker, and Foundation Java API must be in the anonymous (default) mode or all components must be in the certificate mode.

A mixed mode where some components are in anonymous and some in certificated-based is not supported.

When a new component, for example, another connection broker or Documentum CM Server, is installed, the new component will not have certificate settings. Therefore, you must bring all the other related components to the anonymous mode before performing the installation.

Since legacy, prior to 7.1, clients and components will not be able to communicate in the non-anonymous mode, you must use only the anonymous mode in the system that has legacy clients.

2.3.5.1 Constraints and limitations

All files must be available in the \$DOCUMENTUM/dba/secure directory. This is designed so that access to this directory can be monitored or restricted for enhanced security.

2.4 Configuring Documentum Secret Integration Service

Documentum Secret Integration Service (DSIS) is a separate program that connects OpenText Documentum CM products and components with different types of vault.

OpenText Documentum CM supports the following vault types:

- HashiCorp Vault: Stores and retrieves all secrets automatically from the HashiCorp Vault server.

The HashiCorp Vault type is supported in both the on-premises and cloud environments.

- Kubernetes native secrets: Retrieves all secrets from Kubernetes secrets.

OpenText Documentum CM retrieves the passwords automatically from Kubernetes API instead of storing them as environment variables.

The Kubernetes native secrets vault type is supported only in cloud environments.

The DSIS ZIP file is packaged with the Documentum CM Server installer. DSIS must be running on every machines that uses it.

To configure Documentum Secret Integration Service:

1. Download the Documentum CM Server installer from My Support.
2. Extract the installer to a temporary location.
3. Copy the `dsis.zip` file from the temporary location.
4. Extract the `dsis.zip` file to a folder named `dsis`.
5. Configure the DSIS daemon agent.

To update the parameters in the `dsis/application.properties` file to configure the DSIS daemon agent, provide the appropriate information as described in the following table:

Parameter name	Description
<code>dsis.dctm.host</code>	Host of the DSIS daemon agent. DSIS daemon agent always runs on the localhost only. Make sure that the value is set to <code>localhost</code> . Do not change this value.
<code>dsis.dctm.port</code>	Port used for the DSIS daemon agent connection. By default, the value is set to 8200. However, you can change to any available port. Make sure that the port you change is the same port used for the DSIS daemon agent connection.
<code>dsis.dctm.executorThreadCount</code>	Executor thread count used for the load requirement. By default, the value is set to 10. Do not change this value.
<code>dsis.dctm.token</code>	DSIS authentication token. To generate the DSIS authentication token, go to the <code>dsis</code> folder and run the following command: Windows: <pre>java -cp .;dsis.jar;lib/* com.dctm.vault.TokenGenerator</pre> Linux: <pre>java -cp .:dsis.jar:lib/* com.dctm.vault.TokenGenerator</pre> Then, copy the generated token and provide it as a value for this parameter.
<code>dsis.dctm.kvpath</code>	Obtain the Key-Value (KV) path from the HashiCorp Vault server and provide it as a value for this parameter. For more information about Key-Value path in HashiCorp Vault, see <i>HashiCorp</i> documentation.  Note: You can configure individual secrets to have its own Key-Value path.
<code>dsis.dctm.tokenNeeded</code>	Retain the default value.
<code>dsis.dctm.retryFailure</code>	Retain the default value.
<code>dsis.dctm.retrySleepInterval</code>	Retain the default value.

Parameter name	Description
dsis.dctm.enforceListDuringInit	Retain the default value.
dsis.dctm.vault_type	Type of vault. Valid values are K8API or HashiCorp.  Notes <ul style="list-style-type: none"> In an on-premises environment, this is a mandatory parameter and the only valid value is HashiCorp. In a cloud environment, if you want to use HashiCorp Vault, set the value to HashiCorp. However, if you want to use Kubernetes native secrets, set the value to K8API.
dsis.dctm.secretConfigName	Secret configuration name.  Note: Use this parameter only in a cloud environment.

6. Update the authentication method values in the `spring.cloud.vault` parameters. OpenText supports role-based, token-based, Kubernetes token, and TLS authentication methods.

For more information about the authentication method parameters, see *Spring Cloud Vault* documentation.
7. Install or start DSIS as follows:
 - Windows: Install DSIS as a Windows service using the steps provided in the `dsis\DSISWindowsServiceSetup\ReadMe.txt` file.
 - Linux: Start the DSIS daemon agent using the `dsis_start.sh` file in the bash mode.
8. Access the DSIS daemon agent using the following URLs including the HTTP header, `dsis-daemon-token`, with the value generated using the `java -cp .;dsis.jar;lib/* com.dctm.vault.TokenGenerator` command on Windows or the `java -cp .:dsis.jar:lib/* com.dctm.vault.TokenGenerator` command on Linux.
 - `http://localhost:8200/dsis/checkstatus`
The response code 200 for the preceding URL indicates that the DSIS daemon agent is initialized.
 - `http://localhost:8200/dsis/secret/<secret_name>/<key_name>`
The preceding URL provides the password, if the secrets information with the key is found. For example, if you use the `http://localhost:8200/dsis/secret/DOCBASE_PASSWORD/docbase1` URL, it returns the repository password. For more information about secret and key names that must be stored in the HashiCorp Vault server, see “[Configuring Documentum CM Server to use HashiCorp Vault](#)” on page 38.

- `http://localhost:8200/dsis/vaulttype`

The response code 200 for the preceding URL indicates that the vault type is initialized.

9. Tracing information of DSIS is captured in the `<dsis>/dsis.log` file. To enable debugging of the Spring library, add the following entries in the application.properties file:

```
logging.level.ROOT=${LOG_LEVEL_ROOT:warn}
logging.level.org.springframework=${LOG_LEVEL_SPRING:debug}
logging.level.org.springframework.web=${LOG_LEVEL_SPRING_WEB:debug}
logging.level.org.springframework.security=${LOG_LEVEL_SPRING_SECURITY:debug}
```

To upgrade DSIS, stop the older DSIS daemon agent. Then, repeat all the steps as described in [“Configuring Documentum Secret Integration Service” on page 49](#) to use the latest DSIS.

2.5 Installing and configuring Documentum CM Server

2.5.1 Installation workflow

The Documentum CM Server installation process can be broken down into two main processes: You first run Documentum CM Server installer to copy Documentum CM Server program files from the installation media into appropriate directories on the host. You then create connection brokers and repositories in the Documentum CM Server configuration program.

When the installer completes copying Documentum CM Server program files at the end of the first process, you can either proceed by having the installer launch the configuration automatically, or exit the installer and manually launch the configuration later to resume installation. If you select the first option, the configuration program automatically creates a connection broker using the default ports (1489 and 1490) while in the second option, you will need to set up a connection broker first using the configuration program before creating a repository.

2.5.2 Password complexity rules

From the 22.4 release, Documentum CM Server enforces password complexity rules for all the passwords managed by Documentum CM Server such as application server administrator password, AEK passphrase, database owner password, and inline users’ password. This excludes non-inline users’ password such as operating system, OTDS, and so on.

The following rules apply:

- Minimum password length must be 16 characters.

 **Notes**

- If you want to configure the minimum password length between 8 and 64 characters, you must set the `DM_CRYPTO_MIN_PASSWORD_LENGTH` environment variable to your desired value before installation.
- In addition, after configuring the repository, you can set the `DM_CRYPTO_MIN_PASSWORD_LENGTH` docbase module to your desired value for the inline users, so that the configuration value can be preserved for distributed Documentum CM Server setup and for Foundation Java API validation. You can set the docbase module using the following IAPI commands:

```
append,c,docbaseconfig,r_module_name
DM_CRYPTO_MIN_PASSWORD_LENGTH
append,c,docbaseconfig,r_module_mode
<desired value> //for example, 10
save,c,docbaseconfig
reinit,c
```

- Maximum password length must be 64 characters.
- Minimum of one uppercase character. Valid characters are A-Z.
- Minimum of one lowercase character. Valid characters are a-z.
- Minimum of one numeric value. Valid values are 0-9.
- Minimum of one special character. Valid special characters are: hyphen (-), underscore (_), and @

If you upgrade from earlier versions of Documentum CM Server to Documentum CM Server 22.4 and later and used passwords without the complexity rules, a warning appears in the console or the repository log or session log files. OpenText recommends that you to adhere to password complexity rules after the upgrade process.

2.5.3 Installing and configuring Documentum CM Server using GUI

You can install Documentum CM Server by running the `serverSetup.exe` installation program on Microsoft Windows or the `serverSetup.bin` installation program on Linux.

2.5.3.1 Installing Documentum CM Server program files

1. If you want to use HashiCorp Vault, perform all the tasks as described in “Configuring Documentum Secret Integration Service” on page 49 and make sure that DSIS is running.
2. Log on to the computer where you want to install Documentum CM Server as the installation owner.
3. Run `serverSetup.exe` on Windows or `serverSetup.bin` on Linux to launch the Documentum CM Server installer.

4. Accept the license agreement and click **Next**.
5. If you want to configure the HashiCorp Vault secrets, on the **Vault configuration** page, select the **Enable Vault** check box.
6. If you select the **Enable Vault** check box, provide the following information and click **Next**:
 - a. **DSIS URL:** Provide a value in the following format:
`http://localhost:<port mentioned in application.properties>/dsis`
 - b. **DSIS Token:** Provide the `dsis.dctm.token` token value as described in “Configuring Documentum Secret Integration Service” on page 49.

! **Important**

If you enabled the HashiCorp Vault configuration, the installer retrieves all the password information automatically from the HashiCorp Vault server. Make sure that you have stored all the required secrets as described in “Configuring Documentum CM Server to use HashiCorp Vault” on page 38.

7. On Windows, provide an installation directory for Documentum CM Server and click **Next**. The installation directory name must not contain spaces. For example, use `c:\DocumentumProducts` instead of `c:\Documentum Products` as the name of the installation directory. On Windows, the default installation directory is `C:\Documentum`.



Note: On Linux, there is no default installation directory. You can install Documentum CM Server in `/usr/local/bin/Documentum`.

8. On Windows, provide the installation owner password and click **Next**.
 - If HashiCorp Vault is enabled, then the installation owner password is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, then type the installation owner password.
9. Browse and specify the directory where the supported version of Oracle JDK/ OpenJDK is installed and click **Next**.
10. Set the administrator password and specify an available listening port for the embedded application server used by Documentum CM Server and click **Next**.
 - **Admin User Password:** Application server password. You must follow the password complexity rules. “Password complexity rules” on page 52 contains detailed information.
 - If HashiCorp Vault is enabled, then the application server password is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, then type the application server password.

- **Listen Port:** The port on which the application server listens for standard administration connections. A total of 20 ports, starting from the one you specify is used by the application server, and all of them must be available. The default port number is 9080. Irrespective of you accepting the default port or choosing another one, do not change this port after the initial configuration.

11. Review the installation summary and click **Install** to begin installation.
12. Installer prompts you if you want to configure Documentum CM Server now.

- **Configure now:** The installer launches the configuration program. If you select this option, it is not mandatory to select the connection modes (**Native**, **Secure**, and **Native and Secure**). The configuration program automatically creates a connection broker in the **Native and Secure** mode using the default ports (1489 and 1490) and you can proceed to create a repository.



Note: Make sure that the default ports are available.

- **Configure later:** Configure later and exit the installer. You can manually launch the configuration program later to resume the installation process. If you select this option, you must select the connection modes (**Native**, **Secure**, and **Native and Secure**). You must create a connection broker first in the configuration program before creating a repository.

13. After the installation is complete, if you want to enable Federal Information Processing Standard on Windows, do the following:

- a. Set the following environment variables:

```
OPENSSL_CONF=%DM_HOME%\bin\openssl.cnf
OPENSSL_CONF_INCLUDE=%DM_HOME%\bin
OPENSSL_MODULES=%DM_HOME%\bin
```

- b. Go to %DM_HOME%\bin\, open the openssl.cnf file, and update the entries as follows:

```
.include fipsmodule.cnf

[openssl_init]
providers = provider_sect
alg_section = algorithm_sect

[provider_sect]
fips = fips_sect
base = base_sect

[algorithm_sect]
default_properties = fips=yes

[base_sect]
activate = 1
```

14. After the installation is complete, on Linux, do the following:
 - a. Log in as the root user and navigate to the <\$DOCUMENTUM>/dba directory.
 - b. Make sure that the installation owner has the privilege to run the su command to use the dm_check_password_cs16.4 script and then run the dm_root_task_cs16.4 script.

On successful execution of the script, the permission on the dm_secure_writer_cs16.4 file in the \$DOCUMENTUM/dba directory and the dmliswap file in the \$DOCUMENTUM/product/<Documentum CM Server version>/install/admin directory are changed.

2.5.3.2 Creating a connection broker

Skip these steps if you chose **Configure now** in the installer, in which case the installer automatically launches the configuration program and creates a connection broker for you using the default ports (1489 and 1490).

Perform these steps if you chose **Configure later** in the installer:

1. If you want to use HashiCorp Vault, perform all the tasks as described in [“Configuring Documentum Secret Integration Service” on page 49](#) and make sure that DSIS is running.
2. From \$DM_HOME/install, run Server_Configuration_Program.exe on Windows or dm_launch_server_config_program.sh on Linux to launch the Documentum CM Server configuration program.
3. If you want to configure the HashiCorp Vault secrets, on the **Vault configuration** page, select the **Enable Vault** check box.
4. If you select the **Enable Vault** check box, provide the following information and click **Next**:
 - a. **DSIS URL:** Provide a value in the following format:

http://localhost:<port mentioned in application.properties>/dsis
 - b. **DSIS Token:** Provide the dsis.dctm.token token value as described in [“Configuring Documentum Secret Integration Service” on page 49](#).

! Important
If you enabled the HashiCorp Vault configuration, the installer retrieves all the password information automatically from the HashiCorp Vault server. Make sure that you have stored all the required secrets as described in [“Configuring Documentum CM Server to use HashiCorp Vault” on page 38](#).

 5. Click **Connection broker** and then click **Next**.
 6. On Windows, provide the installation owner password and click **Next**.

- If HashiCorp Vault is enabled, then the installation owner password is retrieved from the HashiCorp Vault server.
- If HashiCorp Vault is not enabled, then type the installation owner password.

On Linux, the installation owner password is not prompted.

7. Click **Add a connection broker** and click **Next**.
8. Select a connection mode in which the client connects to the connection broker and click **Next**.
 - **Native**: Documentum CM Server listens for client connection requests only on standard native ports. Documentum CM Server refuses requests for a secure connection.
 - **Secure**: Documentum CM Server listens for client connection requests only on a secure TLS/SSL (Transport Layer Security/Secure Sockets Layer) port for encryption. The client and Documentum CM Server do not use TLS/SSL authentication to authenticate each other. However, the information exchanged between the client and Documentum CM Server is encrypted. Documentum CM Server refuses connection requests other than TLS/SSL connections.
 - **Native and Secure**: Documentum CM Server accepts both native and secure connection requests.



Notes

- “[Connection modes to connect to connection broker and repository](#)” on page 18 contains more information about the TLS versions.
 - Make sure that you copy all the certificate files in the \$DOCUMENTUM/dba/secure directory.
9. For certificate-based SSL communication, click **Use certificates** if you want to enable SSL certificates, provide the required information, and then click **Next**.



Notes

- The **Use certificates** option is available only if you select the connection mode as **Secure** or **Native and Secure**.
- Documentum CM Server does not validate the expiration date of certificates for non-anonymous SSL communication. Use OpenSSL or keytool to validate the certificates.
- If you select **Use certificates**, provide the following information:
 - **Keystore file name**: The name of the keystore file in the PKCS#12 format.
 - **Keystore password file name**: The name of the keystore password file.

- If HashiCorp Vault is enabled, the password is retrieved from the HashiCorp Vault server. The password format is stored as an ID in the <secret_name>/<key_name> format. For more information about the secret ID, see “[Configuring Documentum CM Server to use HashiCorp Vault](#)” on page 38.
 - If HashiCorp Vault is not enabled, generate the keystore password file using the information as described in “[Preparing keystore password](#)” on page 43.
- **Cipher list:** The list of ciphers.
- Provide the information for the Foundation Java API trust store fields.
- **TrustStore:** The full path of the Foundation Java API trust store including the name of the Foundation Java API trust store (dfc.keystore) file.
 - **Password:** The password of the trust store file.
- If HashiCorp Vault is enabled, then the trust store password is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, then type the trust store password.
- Select **Use Default Java TrustStore** if you want to use the default Foundation Java API Java trust store.
10. For certificate-based SSL communication, provide the information, as relevant, and click **Next**.
 - If HashiCorp Vault is enabled, click **Vault AEK key**.
 - If HashiCorp Vault is not enabled, click **Local AEK key**.
 11. Provide the AEK setting information and click **Next**.
 - **Select AEK file:** select an AEK file from the list.
 - **AEK key name:** The AEK key name appears depending on the AEK key file you selected for **Select AEK file**.
 - **AEK passphrase:** The passphrase for the AEK key.
 - If HashiCorp Vault is enabled, then the passphrase for the AEK key is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled and if you do not specify AEK passphrase, it takes a default passphrase for the AEK key.



Note: Make sure that you follow the password complexity rules. “[Password complexity rules](#)” on page 52 contains detailed information.

For more information about creating the AEK key and the passphrase, see *OpenText Documentum Content Management - Server Administration and Configuration Guide (EDCCS250400-AGD)*.

12. Provide the connection broker information and click **Next**.
 - **Connection broker name:** A unique name for the connection broker.
 - **Connection broker port:** The available port reserved for the connection broker. The default port is 1489. If you provide a port (for example, 1480) or accept the default port (1489), make sure that either the next port (for example, 1481) or 1490 is also available.
 - **Service startup type:** On Windows, specify whether the connection broker service starts automatically at startup or is started manually. If you use the key with a passphrase, manual option only is available.
13. Review the summary information and click **Next**.
14. When the connection broker setup is complete, select to perform additional configurations or finish configuration.

2.5.3.3 Creating a repository

This section provides information about creating a repository enabled with and without HashiCorp Vault.

If a repository is created with HashiCorp Vault, all the password fields will not be available for editing as the installer retrieves all the password information stored in the HashiCorp Vault server.



Caution

Repositories using different AEK keys can only function independently. They cannot participate in a federation, replication, and so on where encrypted data or password are shared among them. If you are using multiple repositories with different AEK keys, each repository instance should use a different Java Method Server instance running on different port and with different `dmc.properties`, such that it points to the appropriate repository.



Notes

- Repositories using different AEK keys with different algorithms are not supported.
- The default settings of the IP version parameters have changed from JDK 7 and attempts to set up IP socket using IPv6, even when the machine has no IPv6 networking configured. Application must run with Java option `-Djava.net.preferIPv4Stack=true` to use IPv4 if the machine is not configured with IPv6. For Java Method Server, add `-Djava.net.preferIPv4Stack=true` in Java Method Server startup script if IPv6 is not configured on the machine. To access the Java Method Server URL in an IPv6 environment, you must provide the IPv6 address within the square brackets in the URL.

To create a repository with local- or HashiCorp Vault-based AEK key:

1. If you want to use HashiCorp Vault, perform all the tasks as described in [“Configuring Documentum Secret Integration Service” on page 49](#) and make sure that DSIS is running.
2. If you are not in the Documentum CM Server configuration program, from \$DM_HOME\install, run `Server_Configuration_Program.exe` on Windows or `dm_launch_server_config_program.sh` on Linux to launch the configuration program.
3. If you want to configure the HashiCorp Vault secrets, on the **Vault configuration** page, select the **Enable Vault** check box.
4. If you select the **Enable Vault** check box, provide the following information and click **Next**:
 - a. **DSIS URL:** Provide a value in the following format:
`http://localhost:<port mentioned in application.properties>/dsis`
 - b. **DSIS Token:** Provide the `dsis.dctm.token` token value as described in [“Configuring Documentum Secret Integration Service” on page 49](#).

! Important

If you enabled the HashiCorp Vault configuration, the installer retrieves all the password information automatically from the HashiCorp Vault server. Make sure that you have stored all the required secrets as described in [“Configuring Documentum CM Server to use HashiCorp Vault” on page 38](#).

5. Click **Repository** and click **Next**.
6. On Windows, provide the installation owner password and click **Next**.
 - If HashiCorp Vault is enabled, then the installation owner password is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, then type the installation owner password.
7. Click **Add a new repository** and click **Next**.
8. Provide the information about using existing AEK key or creating new AEK key.
 - **Use existing AEK key:** If you select this option and click **Next**, select **Local AEK key** (if HashiCorp Vault is not enabled) or **Vault AEK key** (if HashiCorp Vault is enabled), and click **Next**.
 - Provide the AEK settings information and click **Next**.
 - **AEK key name:** Provide a unique AEK key name. By default, the name is `CSaek`.



Note: If the AEK key name you provided is same as any existing key, a warning appears. Click **No**.

- **AEK passphrase:**

- If you select **Vault AEK key**, then the passphrase for the AEK key is retrieved from the HashiCorp Vault server.
- If you select **Local AEK key** and if you do not specify an AEK passphrase, it takes a default passphrase.

- **Create new AEK key:** If you select **Create new AEK key** and click **Next**, select **Create Local AEK key** (if HashiCorp Vault is not enabled) or **Create Vault AEK key** (if HashiCorp Vault is enabled), and click **Next**.

- Provide the AEK settings information and click **Next**.

- **Select AEK algorithm:** Select an algorithm from the list.

- **AEK key name:** Provide a unique AEK key name. By default, the name is CSaek.



Note: If the AEK key name you provided is same as any existing key, a warning appears. Click **No**.

- **AEK passphrase:**

- If you select **Create Vault AEK key**, then the passphrase for the AEK key is retrieved from the HashiCorp Vault server.
- If you select **Create Local AEK key** and if you do not specify an AEK passphrase, it takes a default passphrase.



Note: Make sure that you follow the password complexity rules. “[Password complexity rules](#)” on page 52 contains detailed information.

9. Specify a **Data directory path** for storing content files, indicate whether it resides on a **SAN or NAS device**, and click **Next**.

The data directory must not be a top-level directory on a SAN or NAS device such as \\x.x.x.x where <x.x.x.x> is the IP address. For SAN or NAS, type the complete path (use UNC path) including a shared device. An example of a valid data directory on a SAN or NAS device: \\x.x.x.x\<folder1>\<folder2>.

The default data directory is \$DOCUMENTUM/data.

10. Specify a share directory for storing client applications, code examples, and libraries, and click **Next**.

The share directory can be on the Documentum CM Server host or on another host that Documentum CM Server can access over the network.

If you specified a data directory that resides on a SAN or NAS device in [step 9](#), the share directory is automatically created as a sibling on the same level as the data directory. For example, if the data directory is \\x.x.x.x\Documentum\data where <x.x.x.x> is the IP address, the following share directory is automatically created:

\\x.x.x.x\Documentum\share

The default share directory is \$DOCUMENTUM/share.

11. Type the fully qualified domain name (FQDN) of the Documentum CM Server host computer and click **Next**.

An FQDN specifies its exact file path in the tree hierarchy of the Domain Name System (DNS). For example, given a device with a local host name `myhost` and a parent domain name `example.com`, the FQDN is `myhost.example.com`.

12. Provide the repository information and click **Next**.

- **Repository name:** The name for a repository can have up to 32 characters, and must consist of letter, numbers, or underscores (_). The first character must be a letter. All letters and numbers in the name must be ASCII characters. Do not include spaces or non-alphanumeric characters. The repository name `docu` is reserved by the system.
 - **Repository ID:** Valid repository IDs are provided with the Documentum CM Server software. You can also specify your own repository IDs. The repository ID can be any number from 1 to 16777215 and must not start with a zero (0). Each repository ID must be unique on the network. You can request for additional repository IDs to make sure that each of your repository IDs is unique.
 - **Description:** Optionally, type a brief description of the repository.
 - **Authentication domain:** The default domain if the user does not specify a Windows domain when connecting to the repository. Select the domain with the largest number of users. The configuration program automatically fills in this information.
 - **Service startup type:** On Windows, specify whether the repository service starts automatically at startup or is started manually. If you use the key with a passphrase, manual option only is available.
13. Provide the connection broker information, specify if you want to enable SSL certificate, and click **Next**.
- Provide the connection broker information.
 - **Connection Broker Port:** Type the connection broker port number.
 - **Connection Broker Host:** Type the host name.
 - **Use certificates:** Select if you want to enable SSL certificates.
 - If you select **Use certificates**, provide the Foundation Java API trust store information.

- **TrustStore:** The full path of the Foundation Java API trust store including the name of the Foundation Java API trust store (dfc_keystore) file.
 - **Password:** The password of the trust store file.
 - If HashiCorp Vault is enabled, then the trust store password is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, then type the trust store password.
 - Select **Use Default Java TrustStore** if you want to use the default Foundation Java API Java trust store.
14. Click a connection mode for the repository, provide the required information, and click **Next**.
- Click a connection mode for the repository.
 - **Native:** The client connects to the repository through a non-TLS/SSL port.
 - **Secure:** The client connects to the repository through a secure TLS/SSL port. The client and the repository do not use TLS/SSL authentication to authenticate each other. However, the information exchanged between the client and the repository is encrypted.
 - **Native and Secure:** The repository accepts both native and secure connection requests.
 - If the connection mode is **Secure** or **Native and Secure**, then **Use certificates** is enabled.
 - Select **Use certificates** if you want to enable SSL certificates. If you select **Use certificates**, provide the keystore and cipher list information.
 - **Keystore file name:** The name of the keystore file in the PKCS#12 format.
 - **Keystore password file name:** The password of the keystore file.
 - If HashiCorp Vault is enabled, the password is retrieved from the HashiCorp Vault server. The password format is stored as an ID in the <secret_name>/<key_name> format. For more information about the secret ID, see “[Configuring Documentum CM Server to use HashiCorp Vault](#)” on page 38.
 - If HashiCorp Vault is not enabled, generate the keystore password file using the information as described in “[Preparing keystore password](#)” on page 43.
 - **Cipher list:** The list of ciphers.



Notes

- Select **Use certificates** if you enable SSL certificates while creating the connection broker.

- “[Connection modes to connect to connection broker and repository](#)” on page 18 contains more information about TLS versions.
 - The repository must use non-anonymous SSL if the connection broker uses non-anonymous SSL.
- Provide the Foundation Java API trust store information.
 - **TrustStore:** The full path of the Foundation Java API trust store including the name of the Foundation Java API trust store (`dfc.keystore`) file.
 - **Password:** The password of the trust store file.
 - If HashiCorp Vault is enabled, the trust store password is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, type the trust store password.
 - Select **Use Default Java TrustStore** if you want to use the default Foundation Java API Java trust store.
15. Configure your relational database management system (RDBMS) and click **Next**.
- a. Select whether to create a new database and a repository owner (database user with the database owner role) or use existing ones.

If you select to use the existing database, you will not be prompted for the administrator access details. The database administrator can create a database before starting the Documentum CM Server configuration program and select to use the existing database to create a repository. However, the database administrator access details are mandatory if you select to create a new database.
 - b. By default, **Standalone Database Server** is selected. Click **Next** and then perform all the required steps described from [step 15.d](#) in this procedure.
 - c. (Only for Azure SQL Service) If you want to create a repository on Azure SQL service, select **Database Service (PaaS)** and perform all the relevant steps described in “[Using Azure SQL service](#)” on page 78.
 - d. Type or review the following information and click **Next**:
 - **Data source name** on Windows or **Net Service Name** on Linux: The data source used to connect to the database server.
 - **Administrator name** and **Administrator password**: The database administrator account has privileges to create and delete databases and perform other database administrative tasks.
 - If HashiCorp Vault is enabled, the administrator password is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, type the administrator password.

- **Database name:** The name of the database Documentum CM Server uses to store content metadata as well as system and repository information.
- **Repository Owner Account Name and Password:** The database user account with the database owner role that has read and write access rights to the database.
 - If HashiCorp Vault is enabled, the repository owner account password is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, type the repository owner account password.

! Important

For the supported databases, only the following special characters are supported in the repository owner and database passwords:
 ~ (tilde), @ (at), - (hyphen), _ (underscore), = (equal to), : (colon), , (comma), . (period), ? (question mark), and / (forward slash).

 **Notes**

- Make sure that you follow the password complexity rules for Oracle and PostgreSQL databases. “[Password complexity rules](#)” on page 52 contains detailed information.
- If you select to use the existing database, make sure that the user and database configuration returns the results successfully as follows:

Example query:

```
select a.name as userloginname, a.loginname as loginname, b.name as
databasename from sys.syslogins a left outer join sys.databases b on
a.sid=b.owner_sid where a.name = '<repository_owner>';
```

Example output:

userloginname	loginname	databasename
<repository_owner>	<repository_owner>	<database_name>

(1 row affected)

- e. (Only for PostgreSQL) If you want to proceed using **Use Particular Tablespace**, do the following, and click **Next**:

- On Windows: Log in as a postgres user and create a folder called db_<RepositoryName>.dat.dat in C:\Program Files\PostgreSQL\<supported PostgreSQL version number>\data\.
- On Linux: Log in as a postgres user and create a folder called db_<RepositoryName>.dat.dat in /var/lib/pgsql/<supported PostgreSQL version number>/data/.



Note: If you want to proceed using **Use Default Tablespace**, you need not manually create the db_<RepositoryName>.dat.dat folder.

- f. Configure the data file or data devices information and click **Next**. The configuration program automatically fills in the information for **Data device file path** and **Log device file path**.
16. Select the type of mail server.
- If you select **Office365 Configuration** for the OAuth support, register the application in the Microsoft Entra admin center.
For more information about registering an application, see *Microsoft* documentation.
After registering the application, click **Next**, and provide the information for the following parameters:
 - **Tenant Id:** Microsoft 365 Copilot tenant ID.
 - **Client Id:** Microsoft 365 Copilot client ID.
 - **Client Secret:** Password information.
 - If HashiCorp Vault is enabled, then the client secret information is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, then type the client secret information.
 - **Proxy Host:** The host name or IP address of the proxy server. This is an optional parameter for both Windows and Linux.
 - **Proxy Port:** The port through which the proxy communicates. This is an optional parameter for both Windows and Linux.
 - **Owner's email address:** The installation owner's email address that Documentum CM Server uses for email notifications.
- After providing the Microsoft 365 Copilot configuration information, click **Next** to proceed from [step 18](#).
- If you select **SMTP Configuration** and click **Next**, proceed from [step 17](#).
17. Provide the SMTP information and click **Next**.
- **SMTP server name:** The name of the SMTP server or the IP address.
If a valid SMTP server host name is not available, enter an invalid one. If the configuration program fails to connect to the SMTP server you provide, a warning is displayed, but you can still proceed with the installation.
 - **SMTP Port:** The port reserved for the SMTP server. This is an optional parameter for both Windows and Linux. The default port is 25 for native and 465 for secure connections.
 - **Authentication Required:** Specifies if authentication is required. The default value is **No**. If you select **Yes**, provide the information for **SMTP Username** and **SMTP Password**.

- If HashiCorp Vault is enabled, the SMTP password is retrieved from the HashiCorp Vault server.
- If HashiCorp Vault is not enabled, type the SMTP password.
- **SSL Enabled:** Specifies if you want to enable SSL communication. The default value is **No**.
- **Owner's email address:** The installation owner's email address that Documentum CM Server uses for email notifications.

You can modify the SMTP configuration after installation.

18. Specify if you want to use the repository you created as a global registry or to use another one and click **Next**.

- **Yes:** Designates the repository just created as the global registry.

Type the **Login name** and **Password** information for the global registry user in the current repository. Do not use the installation owner or the repository owner credentials. Make sure that you follow the password complexity rules. “[Password complexity rules](#)” on page 52 contains detailed information.

Client applications and other repositories uses this login name and password to connect to the global registry. Record the credentials so that you can provide this information when installing other OpenText Documentum CM products that require global registry access.

- If HashiCorp Vault is enabled, the global registry password is retrieved from the HashiCorp Vault server.
- If HashiCorp Vault is not enabled, type the global registry password.
- **No:** The **Specify whether to use another repository** page appears.

- **Yes:** Designates an existing repository as the global registry.

Type the information about the repository you want to use as the global registry: **Connection Broker Host** and **Port**

Select **Use certificates** if you want to enable SSL certificates.

The current repository is configured to access the remote global registry.



Note: You can change the global registry designation and edit connection information after installation through Documentum Administrator or the `dfc.properties` file. For more information about the instructions on enabling a repository as a global registry, see *OpenText Documentum Content Management - Server Administration and Configuration Guide (EDCCS250400-AGD)*.

- **No:** The global registry is not configured. You can manually add the global registry through Documentum Administrator or the `dfc.properties` file.

19. **Optional** Select **Enable External User Metrics Service** if you want to track external user transactions and click **Next**.
20. **Optional** Select the **XML Store** check box if you want XML Store, specify the following information, and click **Next**:
 - **XML Store Host**: Host name of the machine where BaseX server is installed and running.
 - **XML Store Port**: Port on which BaseX server is running.
 - **XML Store Administrator/Superuser password**: Password of the admin user in BaseX.
 - If HashiCorp Vault is enabled, the XML Store administrator/superuser password is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, type the XML Store administrator/superuser password.
21. Specify the data file path for BaseX and click **Next**.
22. Review the summary information and click **Next**.
23. Select to perform additional configuration or finish configuration.
Always click **Finish** to exit the Documentum CM Server configuration program. Otherwise, you must manually start the application server services.

2.5.3.4 Upgrading a repository

This section provides information about upgrading a repository enabled with and without HashiCorp Vault.

If a repository is upgraded with HashiCorp Vault, all the password fields will not be available for editing as the installer retrieves all the password information stored in the HashiCorp Vault server.



Caution

Repositories using different AEK keys can only function independently. They cannot participate in a federation, replication, and so on where encrypted data or password are shared among them. If you are using multiple repositories with different AEK keys, each repository instance should use a different Java Method Server instance running on different port and with different `dfc.properties`, such that it points to the appropriate repository.



Notes

- Repositories using different AEK keys with different algorithms are not supported.
- You must not change the existing AEK key (local or remote) if your pre-upgrade configuration supports the certificate-based SSL communication.

- The default settings of the IP version parameters have changed from JDK 7 and attempts to set up IP socket using IPv6, even when the machine has no IPv6 networking configured. Application need to run with Java option -Djava.net.preferIPv4Stack=true to use IPv4 if the machine is not configured with IPv6. For Java Method Server, add -Djava.net.preferIPv4Stack=true in the Java Method Server startup script if IPv6 is not configured on the machine.

To upgrade a repository enabled with HashiCorp Vault along with upgrading AEK key:

- If you want to use HashiCorp Vault, perform all the tasks as described in “Configuring Documentum Secret Integration Service” on page 49 and make sure that DSIS is running.
- If you are not in the Documentum CM Server configuration program, from \$DM_HOME\install, run Server_Configuration_Program.exe on Windows or dm_launch_server_config_program.sh on Linux to launch the configuration program.
- If you want to configure the HashiCorp Vault secrets, on the **Vault configuration** page, select the **Enable Vault** check box.
- Provide the following information and click **Next**:
 - DSIS URL:** Provide a value in the following format:

http://localhost:<port mentioned in application.properties>/dsis
 - DSIS Token:** Provide the dsis.dctm.token token value as described in “Configuring Documentum Secret Integration Service” on page 49.



Important

If you enabled the HashiCorp Vault configuration, the installer retrieves all the password information automatically from the HashiCorp Vault server. Make sure that you have stored all the required secrets as described in “Configuring Documentum CM Server to use HashiCorp Vault” on page 38.

- Click **Repository** and click **Next**.
- On Windows, type the installation owner password and click **Next**.
- Select **Upgrade an existing repository** and click **Next**.
- Select **Upgrade AEK key** and click **Next**.
 In the next page, the password for **AEK passphrase** is retrieved from the HashiCorp Vault server. Click **Next**.
- Select **Use existing AEK key** or **Create new AEK key** and click **Next**.
 If you select **Use existing AEK key**, proceed from **step 10**. If you select **Create new AEK key**, proceed from **step 12**.

10. Select **Vault AEK key** and click **Next**.
11. Select the AEK key from the **Select AEK file** list. Click **Next**.
12. Select **Create Vault AEK Key** and click **Next**.
13. Select an algorithm from the **Select AEK algorithm** list, type a name for **AEK key name** and click **Next**.
14. Provide the connection broker information and click **Next**.
15. Click a connection mode for the repository and click **Next**.
16. Select the type of mail server.

- If you select **Office365 Configuration** for the OAuth support, register the application in the Microsoft Entra admin center.

For more information about registering an application, see *Microsoft* documentation.

After registering the application, click **Next**, and provide the information for the following parameters:

- **Tenant Id**: Microsoft 365 Copilot tenant ID.
- **Client Id**: Microsoft 365 Copilot client ID.
- **Client Secret**: Password information.
- **Proxy Host**: The host name or IP address of the proxy server. This is an optional parameter for both Windows and Linux.
- **Proxy Port**: The port through which the proxy communicates. This is an optional parameter for both Windows and Linux.
- **Owner's email address**: The installation owner's email address that Documentum CM Server uses for email notifications.

After providing the Microsoft 365 Copilot configuration information, click **Next** to proceed from [step 18](#).

- If you select **SMTP Configuration** and click **Next**, proceed from [step 17](#).
17. Provide the SMTP information and click **Next**.
 - **SMTP server name**: The name of the SMTP server or the IP address.
If a valid SMTP server host name is not available, enter an invalid one. If the configuration program fails to connect to the SMTP server you provide, a warning is displayed, but you can still proceed with the installation.
 - **SMTP Port**: The port reserved for the SMTP server. This is an optional parameter for both Windows and Linux. The default port is 25 for native and 465 for secure connections.
 - **Authentication Required**: Specifies if authentication is required. The default value is **No**. If you select **Yes**, provide the information for **SMTP Username**.

- **SSL Enabled:** Specifies if you want to enable SSL communication. The default value is **No**.
- **Owner's email address:** The installation owner's email address that Documentum CM Server uses for email notifications.

You can modify the SMTP configuration after installation.

18. **Optional** Select **Enable External User Metrics Service** if you want to track external user transactions and click **Next**.
19. **Optional** Select the **XML Store** check box if you want XML Store, specify the following information, and click **Next**:
 - **XML Store Host:** Host name of the machine where BaseX server is installed and running.
 - **XML Store Port:** Port on which BaseX server is running.
 - **XML Store Administrator/Superuser password:** Password of the admin user in BaseX.
The XML Store administrator/superuser password is retrieved from the HashiCorp Vault server.
20. Specify the data file path for BaseX and click **Next**.
21. In the next page, the password for **Repository owner password** is retrieved from the HashiCorp Vault server. Click **Next**.
22. Review the summary information and click **Next**.
23. Select to perform additional configuration or finish configuration.
Always click **Finish** to exit the Documentum CM Server configuration program. Otherwise, you must manually start the application server services.

To upgrade a repository enabled with HashiCorp Vault with AEK key unchanged:

1. If you want to use HashiCorp Vault, perform all the tasks as described in “Configuring Documentum Secret Integration Service” on page 49 and make sure that DSIS is running.
2. If you are not in the Documentum CM Server configuration program, from \$DM_HOME\install, run **Server_Configuration_Program.exe** on Windows or **dm_launch_server_config_program.sh** on Linux to launch the configuration program.
3. If you want to configure the HashiCorp Vault secrets, on the **Vault configuration** page, select the **Enable Vault** check box.
4. Provide the following information and click **Next**:
 - a. **DSIS URL:** Provide a value in the following format:

```
http://localhost:<port mentioned in application.properties>/dsis
```

- b. **DSIS Token:** Provide the `dsis.dctm.token` token value as described in “Configuring Documentum Secret Integration Service” on page 49.

! **Important**

If you enabled the HashiCorp Vault configuration, the installer retrieves all the password information automatically from the HashiCorp Vault server. Make sure that you have stored all the required secrets as described in “Configuring Documentum CM Server to use HashiCorp Vault” on page 38.

5. Click **Repository** and click **Next**.
6. On Windows, type the installation owner password, and click **Next**.
7. Select **Upgrade an existing repository** and click **Next**.
8. Click **Keep AEK key unchanged** and click **Next**.

In the next page, the password for **AEK passphrase** is retrieved from the HashiCorp Vault server. Click **Next**.

9. Provide the connection broker information and click **Next**.
10. Click a connection mode for the repository and click **Next**.
11. Select the type of mail server.

- If you select **Office365 Configuration** for the OAuth support, register the application in the Microsoft Entra admin center.

For more information about registering an application, see *Microsoft* documentation.

After registering the application, click **Next**, and provide the information for the following parameters:

- **Tenant Id:** Microsoft 365 Copilot tenant ID.
- **Client Id:** Microsoft 365 Copilot client ID.
- **Client Secret:** Password information.
- **Proxy Host:** The host name or IP address of the proxy server. This is an optional parameter for both Windows and Linux.
- **Proxy Port:** The port through which the proxy communicates. This is an optional parameter for both Windows and Linux.
- **Owner's email address:** The installation owner's email address that Documentum CM Server uses for email notifications.

After providing the Microsoft 365 Copilot configuration information, click **Next** to proceed from [step 18](#).

- If you select **SMTP Configuration** and click **Next**, proceed from [step 17](#).

12. Provide the SMTP information and click **Next**.

- **SMTP server name:** The name of the SMTP server or the IP address. If a valid SMTP server host name is not available, enter an invalid one. If the configuration program fails to connect to the SMTP server you provide, a warning is displayed, but you can still proceed with the installation.
- **SMTP Port:** The port reserved for the SMTP server. This is an optional parameter for both Windows and Linux. The default port is 25 for native and 465 for secure connections.
- **Authentication Required:** Specifies if authentication is required. The default value is **No**. If you select **Yes**, provide the information for **SMTP Username**.
- **SSL Enabled:** Specifies if you want to enable SSL communication. The default value is **No**.
- **Owner's email address:** The installation owner's email address that Documentum CM Server uses for email notifications.

You can modify the SMTP configuration after installation.

13. **Optional** Select **Enable External User Metrics Service** if you want to track external user transactions and click **Next**.
14. **Optional** Select the **XML Store** check box if you want XML Store, specify the following information, and click **Next**:
 - **XML Store Host:** Host name of the machine where BaseX server is installed and running.
 - **XML Store Port:** Port on which BaseX server is running.
 - **XML Store Administrator/Superuser password:** Password of the admin user in BaseX.The XML Store administrator/superuser password is retrieved from the HashiCorp Vault server.
15. Specify the data file path for BaseX and click **Next**.
16. In the next page, the password for **Repository owner password** is retrieved from the HashiCorp Vault server. Click **Next**.
17. Review the summary information and click **Next**.
18. Select to perform additional configuration or finish configuration.

Always click **Finish** to exit the Documentum CM Server configuration program. Otherwise, you must manually start the application server services.

To upgrade a repository not enabled with HashiCorp Vault with AEK key unchanged:

1. If you are not in the Documentum CM Server configuration program, from \$DM_HOME\install, run `Server_Configuration_Program.exe` (Windows) or `dm_launch_server_config_program.sh` (Linux) to launch the configuration program.
2. On the **Vault configuration** page, do not select the **Enable Vault** check box.
3. Click **Repository** and click **Next**.
4. On Windows, type the installation owner password, and click **Next**.
5. Click **Upgrade an existing repository**, select the repository you want to upgrade from the list, and then click **Next**.
6. Click **Keep AEK key unchanged** and click **Next**.
7. Provide the local AEK setting information for **AEK passphrase** and click **Next**. If you do not specify AEK passphrase, it takes a default passphrase for the AEK.
8. Provide the connection broker information and click **Next**.
9. Click a connection mode for the repository and click **Next**.
10. Select the type of mail server.

- If you select **Office365 Configuration** for the OAuth support, register the application in the Microsoft Entra admin center.

For more information about registering an application, see *Microsoft* documentation.

After registering the application, click **Next**, and provide the information for the following parameters:

- **Tenant Id:** Microsoft 365 Copilot tenant ID.
- **Client Id:** Microsoft 365 Copilot client ID.
- **Client Secret:** Password information.
- **Proxy Host:** The host name or IP address of the proxy server. This is an optional parameter for both Windows and Linux.
- **Proxy Port:** The port through which the proxy communicates. This is an optional parameter for both Windows and Linux.
- **Owner's email address:** The installation owner's email address that Documentum CM Server uses for email notifications.

After providing the Microsoft 365 Copilot configuration information, click **Next** to proceed from [step 18](#).

- If you select **SMTP Configuration** and click **Next**, proceed from [step 17](#).

11. Provide the SMTP information and click **Next**.

- **SMTP server name:** The name of the SMTP server or the IP address. If a valid SMTP server host name is not available, enter an invalid one. If the configuration program fails to connect to the SMTP server you provide, a warning is displayed, but you can still proceed with the installation.
- **SMTP Port:** The port reserved for the SMTP server. This is an optional parameter for both Windows and Linux. The default port is 25 for native and 465 for secure connections.
- **Authentication Required:** Specifies if authentication is required. The default value is **No**. If you select **Yes**, provide the information for **SMTP Username** and **SMTP Password**.
- **SSL Enabled:** Specifies if you want to enable SSL communication. The default value is **No**.
- **Owner's email address:** The installation owner's email address that Documentum CM Server uses for email notifications.

You can modify the SMTP configuration after installation.

12. **Optional** Select **Enable External User Metrics Service** if you want to track external user transactions and click **Next**.
13. **Optional** Select the **XML Store** check box if you want XML Store, specify the following information, and click **Next**:
 - **XML Store Host:** Host name of the machine where BaseX server is installed and running.
 - **XML Store Port:** Port on which BaseX server is running.
 - **XML Store Administrator/Superuser password:** Password of the admin user in BaseX.
14. Specify the data file path for BaseX and click **Next**.
15. Type the password for **Repository owner password** and click **Next**.
16. Review the summary information and click **Next**.
17. Select to perform additional configuration or finish configuration.

Always click **Finish** to exit the Documentum CM Server configuration program. Otherwise, you must manually start the application server services.

To upgrade a repository not enabled with HashiCorp Vault along with upgrading AEK key:

1. If you are not in the Documentum CM Server configuration program, from <`$DM_HOME>\install`, run `Server_Configuration_Program.exe` (Windows) or `dm_launch_server_config_program.sh` (Linux) to launch the configuration program.

2. On the **Vault configuration** page, do not select the **Enable Vault** check box.
3. Click **Repository** and click **Next**.
4. Type the installation owner password and click **Next**.
5. Click **Upgrade an existing repository** and click **Next**.
6. Click **Upgrade AEK key** and click **Next**.
7. Select **Use existing AEK key or Create new AEK key** and click **Next**.
If you select **Use existing AEK key**, proceed from [step 9](#).
8. Click **Local AEK key** and click **Next**.
9. Provide the local AEK setting information for **AEK passphrase**. If you do not specify AEK passphrase, it takes a default passphrase for the AEK.



Note: If you select **Create new AEK key** in [step 7](#), make sure that you follow the password complexity rules. ["Password complexity rules" on page 52](#) contains detailed information.

10. Provide the connection broker information and click **Next**.
11. Click a connection mode for the repository and click **Next**.
12. Select the type of mail server.

- If you select **Office365 Configuration** for the OAuth support, register the application in the Microsoft Entra admin center.

For more information about registering an application, see *Microsoft* documentation.

After registering the application, click **Next**, and provide the information for the following parameters:

- **Tenant Id:** Microsoft 365 Copilot tenant ID.
- **Client Id:** Microsoft 365 Copilot client ID.
- **Client Secret:** Password information.
- **Proxy Host:** The host name or IP address of the proxy server. This is an optional parameter for both Windows and Linux.
- **Proxy Port:** The port through which the proxy communicates. This is an optional parameter for both Windows and Linux.
- **Owner's email address:** The installation owner's email address that Documentum CM Server uses for email notifications.

After providing the Microsoft 365 Copilot configuration information, click **Next** to proceed from [step 18](#).

- If you select **SMTP Configuration** and click **Next**, proceed from [step 17](#).

13. Provide the SMTP information and click **Next**.

- **SMTP server name:** The name of the SMTP server or the IP address.
If a valid SMTP server host name is not available, enter an invalid one. If the configuration program fails to connect to the SMTP server you provide, a warning is displayed, but you can still proceed with the installation.
- **SMTP Port:** The port reserved for the SMTP server. This is an optional parameter for both Windows and Linux. The default port is 25 for native and 465 for secure connections.
- **Authentication Required:** Specifies if authentication is required. The default value is **No**. If you select **Yes**, provide the information for **SMTP Username** and **SMTP Password**.
- **SSL Enabled:** Specifies if you want to enable SSL communication. The default value is **No**.
- **Owner's email address:** The installation owner's email address that Documentum CM Server uses for email notifications.

You can modify the SMTP configuration after installation.

14. **[Optional]** Select **Enable External User Metrics Service** if you want to track external user transactions and click **Next**.
15. **[Optional]** Select the **XML Store** check box if you want XML Store, specify the following information, and click **Next**:
 - **XML Store Host:** Host name of the machine where BaseX server is installed and running.
 - **XML Store Port:** Port on which BaseX server is running.
 - **XML Store Administrator/Superuser password:** Password of the admin user in BaseX.
16. Specify the data file path for BaseX and click **Next**.
17. Type the **Repository owner password** and click **Next**.
18. Review the summary information and click **Next**.
19. Select to perform additional configuration or finish configuration.

Always click **Finish** to exit the Documentum CM Server configuration program. Otherwise, you must manually start the application server services.

2.5.3.5 Viewing the configuration summary

On the **Configuration options** page, select **Configuration Summary** and click **Next** to view the details of key components configured on the host, such as repository, connection broker, and Java Method Server.

The following options are not available:

- Viewing the details of the Java Method Server instance.
- Viewing the user accounts in repository.
- Viewing the user accounts details in each repository by default.

2.5.3.6 Viewing the version details of installed products or components

Documentum CM Server installation program records the version information of all the products or components installed in a registry file on the target host. The information includes the product or component version, patch information, build number, installation date and so on. The registry file is located in the software installation root directory `$DOCUMENTUM/DctmRegistry.xml`. There could be multiple registry files if you install multiple OpenText Documentum CM products on one target host located in a different installation root directory.

2.5.3.7 Using Azure SQL service



Note: Documentum CM Server supports Azure SQL service on Windows only.

1. Set up Azure.
2. Create a resource group. A resource group in Azure is a folder to keep your collection. It does not serve any other purpose.
3. Make sure that you have the `Microsoft.Sql` and `Microsoft.SqlVirtualMachine` services registered in your subscription.
4. Deploy the preconfigured database servers (standalone or managed).
Microsoft Azure documentation contains detailed information.
5. (Only for configuring Documentum CM Server in Windows VM on Azure)
Deploy the SQL virtual machine in the network where your database instance is deployed. You must have access to the instance from VM.



Note: OpenText does not recommend the spot instance.

6. Launch `Server_Configuration_Program.exe` located in `$DM_HOME\product\<release-version>\install`.
7. Perform all the required steps from **step 5** through **step 14** as described in “[To create a repository with local- or HashiCorp Vault-based AEK key:](#)” on page 60.

8. On the **Configure repository – select database and server account** page, select the **Create a new SQL Server user account and database** option, and click **Next**.

If you select to use an existing database, administrator access details are not prompted. The database administrator can create a database before starting the Documentum CM Server configuration program and select to use an existing database to create a repository. However, the database administrator access details are mandatory if you select to create a new database.



Note: Azure elastic pool is not supported.

9. For replicating the on-premises type of deployment in Windows virtual machine on Azure, select **Standalone Database Server**, click **Next**, and then perform all the relevant steps described from [step 12](#) in this procedure.
 10. For creating a repository on Azure SQL service, select **Database Service (PaaS)**, and click **Next**.
 11. On the **PaaS Deployment Options** page, perform one of the following tasks:
 - For creating an Azure standalone instance, select **Azure Standalone Instance**, specify the following information, click **Next**, and then perform all the relevant steps described from [step 12](#) in this procedure:
 - **Service Objective (mandatory)**
 - **Maxsize (optional)**
 - **Edition (optional)**
 - Or
 - To create an Azure managed instance, select **Azure Managed Instance**, click **Next**, and then perform all the relevant steps described from [step 12](#) in this procedure.
12. Type or review the following information:
 - **Data source name:** The data source used to connect to the database server.
 - **Administrator name and Administrator password:** The database administrator account has privileges to create and delete databases and perform other database administrative tasks.
 - **Database name:** The name of the database, Documentum CM Server uses to store content metadata as well as system and repository information.
 - **Repository Owner Account Name and Password:** The database user account with the database owner role that has read and write access rights to the database. Make sure that you follow the password complexity rules.
[“Password complexity rules” on page 52](#) contains detailed information.
 13. Click **Next**.

14. For Standalone Database Server only – If you want to proceed using the **Use Particular Tablespace** option, log in as an Azure SQL user and create a folder called db_<RepositoryName>.dat.mdf (for data file) or db_<RepositoryName>.dat.ldf (for log file) in C:\Program Files\<database>\<supported database version number>\data\.
If you want to proceed using the **Use Default Tablespace** option, you need not manually create the folder.
15. For Standalone Database Server only – Configure the data file or data devices information. The configuration program automatically fills in the information for **Data device file path** and **Log device file path**.
16. Click **Next**.
17. Perform all the required steps from [step 17](#) through [step 23](#) as described in “[To create a repository with local- or HashiCorp Vault-based AEK key:](#)” on page 60.

2.5.3.7.1 Migrating data from on-premises to Azure cloud platform

Perform the following tasks to migrate data from the SQL Server installed in an on-premises environment to Azure SQL service:

1. Stop the repository.
2. Take a backup of the database. You can use any third-party tools of your choice.
3. Take a backup of the contents of the dba, config and data folders in the existing Documentum CM Server and place them in the machine where you are deploying Documentum CM Server.
4. Download and install the Data Migration Assistant (DMA) tool from the Microsoft Azure website.
5. Create the database instances (with the same on-premises SQL collation) in the respective type of Azure SQL services.
6. Perform all the required steps using the DMA tool to migrate metadata as described in *Microsoft Azure* documentation.
7. After the metadata is migrated, update the path (if different) specified in the location objects with the same path as the existing directory.

2.5.4 Installing and configuring Documentum CM Server using command line

You can install, configure, and also uninstall Documentum CM Server from the command line (silent installation). You can perform the following operations:

- Installing Documentum CM Server
- Adding a new connection broker
- Upgrading connection broker
- Deleting connection broker
- Adding a new repository
- Adding a new server for repository
- Upgrading repository
- Deleting repository
- Uninstalling Documentum CM Server

2.5.4.1 Creating the silent installation files

1. Create the silent installation files when you are installing Documentum CM Server using the GUI by running the following commands from the command line:

For Windows:

- Documentum CM Server:

```
serverSetup.exe -r <C:|silent|Windows|<Silent installer Property file name>>
```

- Connection broker and repository:

```
Server_Configuration_Program.exe -r <C:|silent|Windows|<Silent installer property file name>>
```

- remote Content Server:

```
cfsConfigurationProgram.exe -r <C:|silent|Windows|<Silent installer property file name>>
```

- Java Method Server:

```
jmsConfig.exe -r <C:|silent|Windows|<Silent installer property file name>>
```

For Linux:

- Documentum CM Server:

```
serverSetup.bin -r <Path of the Silent installer property file name>/<Silent installer property file name>
```

- Connection broker and repository:

```
dm_launch_server_config_program.sh -r <Path of the Silent installer property file>/<Silent installer property file name>
```

- remote Content Server:

```
dm_launch_cfs_server_config_program.sh -r <Path of the Silent installer  
property file>/<Silent installer property file name>
```

- Java Method Server:

```
jmsConfig.sh -r <Path of the Silent installer property file>/<Silent installer  
property file name>
```

2. After the files are generated, use a text editor to open the files to update or change the values of the variables, where required.
3. **Optional** To enable SMTP for the repository configuration, add the following parameter in the repository silent configuration properties file and set the value to true as follows:

```
SERVER.SMTP_CALL_EBS=true
```
4. **Optional** To track the external user transactions, add the following parameter in the repository silent configuration properties file and set the value to true as follows:

```
SERVER.EXTERNAL_USER_METRICS=true
```
5. **Optional** To enable HashiCorp Vault, add the following parameters in the silent installer property file:

```
IS_VAULT_ENABLED=true  
DSIS_URL=http://localhost:8200/dsis  
DSIS_TOKEN=<DSIS token value>
```

The preceding list of parameters must be added in all the silent configuration files to configure with HashiCorp Vault.

- a. Store all the passwords in the HashiCorp Vault server. For more information about storing all the passwords in the HashiCorp Vault server, see *HashiCorp* documentation.
- b. Configure the silent installer property file and replace all password with secret ID as described in “[Configuring Documentum CM Server to use HashiCorp Vault](#)” on page 38.
- c. If you want to enable certificate-based SSL communication, prepare the certificate passwords. The contents of the broker.pwd and server.pwd files must be updated with the <secret_name>/<key_name>. For example:

In the broker.pwd file:

```
DOCBROKER_CERT_PASSWORD/<host name of connection broker>
```

In the server.pwd file:

```
DOCBASE_CERT_PASSWORD/<repository name>
```

- d. Save the configuration file.



Note: You can also use the silent installation template files provided to you for reference or customization. The silent installation template files are located in \$DOCUMENTUM/product/<version>/install/silent/templates.

You can create the silent installation file for remote Content Server using the silent installation file of connection broker and repository. To achieve this, you can use the silent installation tool located in \$DOCUMENTUM/product/<version>/install/silent/silenttool. The Readme.txt at this location contains the instructions.

2.5.4.2 Running the installation and configuration from the command line

1. Log in to the host system using your host login.
2. Change the value of INSTALLER_UI to <silent> in all the silent installation files.



Note: If you select to configure the connection mode of the connection broker and the repository as **Native and Secure**, make sure that you change the value of SERVER.DOCBROKER_CONNECT_MODE and SERVER.CONNECT_MODE to <dual>.

3. **Optional** To enable HashiCorp Vault, add the following parameters in the silent installer property file:

```
IS_VAULT_ENABLED=true
DSIS_URL=http://localhost:8200/dsis
DSIS_TOKEN=<DSIS token value>
```

The preceding list of parameters must be added in all the silent configuration files to configure with HashiCorp Vault.

- a. Store all the passwords in the HashiCorp Vault server. For more information about storing all the passwords in the HashiCorp Vault server, see *HashiCorp* documentation.
- b. Configure the silent installer property file and replace all password with secret ID as described in “[Configuring Documentum CM Server to use HashiCorp Vault](#)” on page 38.
4. Run the silent installation program using the following commands from the command line:

For Windows:

- Installing Documentum CM Server program files:

```
serverSetup.exe -f <C:\silent\Windows|<Silent installer property file name>>
```

- Creating a connection broker and repository:

```
Server_Configuration_Program.exe -f <C:\silent\Windows|<Silent installer property file name>>
```

- Installing remote Content Server program files:

```
cfsConfigurationProgram.exe -f <C:\silent\Windows|<Silent installer property file name>>
```

- Installing Java Method Server program files:

```
jmsConfig.exe -f <C:\silent\Windows|<Silent installer property file name>>
```

For Linux:

- Installing Documentum CM Server program files:

```
serverSetup.bin -f <Path of the Silent installer property file name>/<Silent  
installer property file name>
```

- Creating a connection broker and repository:

```
dm_launch_server_config_program.sh -f <Path of the Silent installer property  
file name>/<Silent installer property file name>
```

- Installing remote Content Server program files:

```
dm_launch_cfs_server_config_program.sh -f <Path of the Silent installer  
property file name>/<Silent installer property file name>
```

- Installing Java Method Server program files:

```
jmsConfig.sh -f <Path of the Silent installer property file name>/<Silent  
installer property file name>
```

You may not see the progress message of the silent installation when it is run from the command line as per the InstallAnywhere design. All messages are saved in the installation log file. On Linux, the silent installation command execution does not return until the installation is complete. However, on Windows, the silent installation command execution returns even when the silent installation is in progress. To avoid this behavior on Windows, you can launch the silent installation command using the start /w silent-install-command command.



Note: If an error occurs while running the installation from the command line, the installation stops. Add KEEP_TEMP_FILE and set it to <true> to make sure that the temporary files are retained for troubleshooting, if needed.

2.6 Completing the installation

2.6.1 Reviewing Documentum CM Server installation and configuration logs

Documentum CM Server installer and configuration program both create log files. The log files are stored in one or many of the following directories:

- Installation owner's desktop (Windows).
- User home directory (Linux).
- The current working directory.

For installer, the current working directory is the directory from which you started the program. For the configuration program, the current working directory is typically <\$DM_HOME>\install (Windows and Linux).

- The parent directory of the installation directory, if the installation owner does not have write permission on the current working directory.
- The user's home directory, if the installation owner does not have write permission on the parent directory.

The log filenames are `install.log`, `log4j.log`, and `UniversalServerConfigurator_Install_<mm_dd_yyyy_hh_mm_ss>.log`.

To verify the successful installation or upgrade in Windows, check the `install.log` located in `$DOCUMENTUM/product/<release version>/install/`.

To verify the successful installation or upgrade in Linux, check the following log files:

- `install.log` located in `$DOCUMENTUM/product/<release version>/install/`
- `log4j.log` located in `$DOCUMENTUM/logs`

Each script that runs during the repository configuration creates a log file. These are stored in the `$DOCUMENTUM/dba/config/<repository_name>` directory.

Documentum CM Server stores other log files in the `$DOCUMENTUM/dba/log` directory. After you install or upgrade Documentum CM Server, examine the log file for the repository for error reports. The log is called `<repository_name>.log.save.<date>.<time>` where `<repository_name>` is the name of the repository you created or upgraded, and `<date>` and `<time>` are the date and time the log was saved.

2.6.2 Configuring symbolic link path for Java

During Documentum CM Server installation, a symbolic link path for Java is installed at `<$DOCUMENTUM>/java64/<JAVA_LINK>`.

The symbolic link path points to the actual Java path and is independent of the specific Java version. The symbolic link path is used in Documentum CM Server scripts to specify the actual Java executable path. You only need to change the target of the symbolic link and need not update the scripts after the Java upgrade. The symbolic link is created on Linux. On Windows, it is only available on the systems that support mklink and NTFS symbolic link.

2.6.3 Enabling the purge audit job

The purge audit job deletes old audit trail objects from the repository. The job runs as the installation owner. However, when a repository is created, the installation owner is not granted sufficient extended privileges to run the job.

After you create a repository, create a new user with superuser privileges, connect as that user, and grant the installation owner account Purge Audit extended privileges.

2.7 Post-installation requirements and tasks

After you have successfully installed Documentum CM Server, make sure that Java Method Server is up and running, and then perform the tasks described in this section to configure the system as required and get it up and running.

2.7.1 Licensing OpenText Documentum CM

2.7.1.1 Procuring license file from OpenText

Procure the license file from OpenText as described in the *Obtain the license key* section in OpenText Documentum Content Management License Management (https://support.opentext.com/csm?id=kb_article_view&sysparm_article=KB0834991).

2.7.1.2 Configuring OTDS and license

This section provides the information about configuring OTDS and license.

- For information about configuring OTDS using an automated method and configuring license using a manual method, see “[To configure OTDS using an automated method and to configure license using a manual method](#):” on page 86.
- For information about configuring OTDS and license using a manual method, see “[To configure OTDS and license using a manual method](#):” on page 90.

To use these instructions, ensure that you have completed the installation of OTDS. For more information about installing OTDS, see *OpenText Directory Services - Installation and Administration Guide* (OTDS250400-IWC).

To configure OTDS using an automated method and to configure license using a manual method:

1. Sign in to the OTDS Admin website using the following information:
 - URL: URL to access the OTDS Admin website.
<fully qualified domain name of server>:<web application server port number>/otds-admin
 - User name: OTDS Admin user name.
For example: otadmin@otds.admin
 - Password: OTDS Admin password.
2. Synchronize the resources to the Documentum CM repository using the following steps:
 - a. Click **Resources**.
 - b. Click *<your resource name>* > **Actions** > **Consolidate**.

For example: Your resource name can be `otdctmresource`.

3. Import the license file in OTDS using the following steps:
 - a. On the **License Keys** page, click **Add**.
 - b. On the **General** tab, provide values for the following fields:
 - **License Key Name:** Unique name.
For example: `otdctmlicense`
 - **Resource ID:** License linked to the resource.
 - c. On the **License Key** tab, click **Get License File**, browse and select the license file.
 - d. Click **Save**.
4. Create a `businessadmin` user in the `otds.admin` partition using the following steps:
 - a. Click **Partitions**.
 - b. Click **otds.admin > Actions > View Members..**
 - c. Click **Add > New User**.
 - d. On the **General** page, in the **User Name** box, type a name for this user as `businessadmin`.
 - e. On the **Account** page, in the **Password Options** area, do the following:
 - i. Click **Do not require password change on reset** from the list.
 - ii. Clear the **User cannot change password** check box, if selected.
 - iii. Select the **Password never expires** check box and click **Save**.

Additional information and tasks

Documentum CM client

Create OTDS users with the user name as `d2_mail_manager` and `d2_wf_notification_user` and keep the passwords as blank in a new non-synchronized partition which is not associated to any resources in OTDS.

By default, OTDS is enabled for client configuration. If you disable OTDS, add another user with the user name as `install_owner_user` to the same partition with the password as blank.

In the **Password Options** area, click **Do not require password change on reset** from the list, and do the following:

- Select the **User cannot change password** check box.
- Select the **Password never expires** check box.

This is applicable only for the system account user partition.

Workflow Designer

If you want to use Workflow Designer with the skip SSO feature, only a user with the `install_owner` privilege can access without a license.

Advanced Workflow

Advanced Workflow is available with advanced license or as an add-on. As a user, you can build processes using Advanced Workflow but to install these processes, the xDA Documentum CM repository endpoint user must have advanced Documentum CM or an add-on license.

To start a workflow at runtime from any Documentum CM client components, you must have advanced Documentum CM or an add-on license and the required transaction capability. The transaction counter increments at runtime when a user creates a new instance of workflow irrespective of the state of the workflow.

Reports

Create an OTDS user with the user name as `dctmreports` and keep the password as blank.

In the **Password Options** area, click **Do not require password change on reset** from the list, and do the following:

- Select the **User cannot change password** check box.
- Select the **Password never expires** check box.

OpenText Documentum CM for Microsoft 365, OpenText Documentum CM Online Editing Service, and Notification Service

Create an OTDS user with the user name as `M365_SERVICE` with the password as `Xecmserviceuser@123`. The password is case-sensitive.

In the **Password Options** area, click **Do not require password change on reset** from the list, and do the following:

- Clear the **User cannot change password** check box.
- Select the **Password never expires** check box.

Documentum Archive Services for SAP Solutions

Create an OTDS user with the user name as `installownerusername` with the password as `installownerpassword`. This is a service account.

The system automatically changes the password after the service is started.

5. Add the `businessadmin` user to the `otdsbusinessadmins` group in OTDS using the following steps:
 - a. Click **Users & Groups**, and select the **Groups** tab.
 - b. In the **Search** box, type `otdsbusinessadmins` to find the `otdsbusinessadmins@otds.admin` group.

- c. On the **Groups** tab, click **Actions > Edit Membership**.
 - d. On the **otdsbusinessadmins@otds.admin** page, on the **Members** tab, click **Add Member**.
 - e. In the **Search** box, type **businessadmin** to find the **businessadmin@otds.admin** member.
 - f. Select the **businessadmin@otds.admin** check box, and click **Add Selected**.
6. In the Documentum CM Server machine, run the following command in IAPI to create the **dm_otds_license_config** object.

```
create,c,dm_otds_license_config
set,c,1,otds_url
<otds_url_including_rest>
set,c,1,license_keyname
<license_keyname>
set,c,1,business_admin_name
<business_adminname>
set,c,1,business_admin_password
<password>
save,c,1
```

For example:

```
create,c,dm_otds_license_config
set,c,1,otds_url
http://documentumcm:8080/otdssws/rest
set,c,1,license_keyname
otdctmlicense
set,c,1,business_admin_name
businessadmin
set,c,1,business_admin_password
Password-1234567890
save,c,1
```



Notes

- Alternatively, you can create the **dm_otds_license_config** object using the Documentum Administrator graphical user interface.

For instructions, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

- If you modify the **dm_otds_license_config** object after the first time, you must run the **apply,c,NULL,FLUSH_OTDS_CONFIG** command in IAPI.

7. Allocate a license in OTDS to a partition using the following steps:

- a. Click **Partitions > <your desired partition> > Actions > Allocate to License**.
- b. On the **Allocate to License** page, click the relevant counter from the list.

You can also allocate the license to users and groups. When you allocate a license, ensure that you allocate the relevant counter to a user or group.

- c. Click **Allocate to License**.



Notes

- If you modify the allocated license file after the initial deployment, you must restart OTDS.

- You can allocate users to your license any time, but a user must exist before you can allocate the user to a license. The changes to the allocation, deallocation, and revocation take effect after 24 hours for releases prior to and including the 24.4 release.

- For the 25.2 release, run the following command in IAPI to get the allocation, deallocation, revocation, or deletion with immediate effect:

```
apply,c,NULL,FLUSH_JMS_OTDS_CACHE
```

- From the 25.4 release, run the following command in IAPI to get the allocation, deallocation, revocation, or deletion with immediate effect:

```
apply,c,NULL,FLUSH_OTDS_CACHE
```

8. In the Documentum CM Server machine, open the `otdsauth.properties` file in the `$DM_HOME\OTDSAuthLicenseHttpServerBin\config` folder and verify if information for `certificate`, `otds_rest_credential_url`, `otds_rest_ticket_url`, and `admin_username` are updated.
9. Sign in to a deployed application (for example, Documentum Administrator) with any licensed user to verify the license configuration.
Ensure that the user authentication is successful.

To configure OTDS and license using a manual method:

1. Sign in to the OTDS Admin website using the following information:
 - URL: URL to access the OTDS Admin website.
<fully qualified domain name of server>: <web application server port number>/otds-admin
 - User name: OTDS Admin user name.
For example: `otadmin@otds.admin`
 - Password: OTDS Admin password.
2. Create a non-synchronized user partition in OTDS using the following steps:
 - a. Click **Partitions > Add > New Nonsynchronized User Partition**.
 - b. In the **Name** box, type a name for your user partition.
For example: `otdctmpartitions`
 - c. Click **Save**.

For information to create a synchronized user partition in OTDS, see *OpenText Directory Services - Installation and Administration Guide (OTDS250400-IWC)*.
3. Create a synchronized resource in OTDS using the following steps:
 - a. Click **Resources > Add**.

- b. On the **General** page, do the following:
 - i. In the **Resource name** box, type a name for the resource.
For example: otdctmresource
 - ii. In the **Description** box, type a description for the resource.
 - iii. On the **Synchronization** tab, select the **User and group synchronization** check box and click **REST(Generic)** for **Synchronization connector**.
 - iv. Click **Next**.
 - c. On the **Resources > <your resource name> > Actions** page, select **Properties**.
For example: Your resource name can be otdctmresource.
 - i. On the *<your resource name>* page, select the **Connection Information** tab.
 - ii. On the **Connection Information** page, provide the value for the following fields:
 - **Base URL:** URL endpoint for user or group provisioning REST API.
Use the following format:
`http://<host name>:<port>/dmotdsrest`
For example: `http://dcs-pg-jms-service:9080/dmotdsrest`
 - **Username:** Installation owner user name for the repository.
Use the following format:
`<repository name>\<installation owner user name>`
For example: `docbase1\dmadmin`
 - **Password:** Installation owner password for the repository.
 - d. Click **Test Connection**.
 - e. On the **User Attribute Mappings** tab, click **Reset to Default**.
 - f. For **User Attribute Mappings**, add the `client_capability` attribute with Format value of 2.
Add the `default_folder` attribute with OTDS Attribute value of `cn` and Format value of `/%`.
 - g. Retain the default value for all other settings.
 - h. Click **Next**.
 - i. On the **Group Attribute Mappings** tab, click **Reset to Default**.
 - j. Click **Save**.
4. Click **Access Roles** and go to **Access to <your resource name>** (for example, otdctmresource).
 - a. Click **Actions** and select **Include Groups**.
 - b. Click **Actions** and select **View Access Role Details**.

- c. On the **User Partitions** tab, add the partition you created.
 - d. Click **Save**.
5. Click **Resources** > *<your resource name>* > **Actions** > **Consolidate** to synchronize the members to your repository.
 6. Click **OAuth clients** and create an OAuth client.
 - a. For **Redirect URLs**, add the following to the **Redirect URLs** list:
 - <Ingress URL>/D2/d2_otds.html
 - <Ingress URL>D2/OTDSLogoutResponse.html
 - <Ingress URL>/D2-Config/d2config_otds.html
 - <Ingress URL>/D2-Config/OTDSLogoutResponse.html
 - <Ingress URL>D2-Smartview/ui
 - <Ingress URL>oes-connector
 - <Ingress URL>AdminConsole
 - <Ingress URL>d2-rest
 - <Ingress URL>
-  **Note:** If you have disabled OTDS for client configuration, do not add the following URLs to the **Redirect URLs** list:
- <Ingress URL>/D2-Config/d2config_otds.html.
 - <Ingress URL>/D2-Config/OTDSLogoutResponse.html.
- b. Click **Save**.
7. Click **Auth Handlers**, select **http.negotiate**, click **Action**, and select **Disable**.
 8. Create roles using the following steps:
 - a. Click **Partitions** > *<your desired partition>* > **Actions** > **View Members**.
 - b. On the **Roles** tab, click **Add** > **New Role**.
 - c. Add the following roles:
 - Client Capabilities: Client_Consumer, Client_Contributor, Client_Coordinator, and Client_System_Administrator.
 - User Privileges: privilege_createtype, privilege_createcabinet, privilege_creategroup, privilege_sysadmin, and privilege_superuser.
-  **Note:** The role name is case-sensitive.
- d. Go to **Application Roles** to view the list of added roles.

9. Import the license file in OTDS using the following steps:
 - a. On the **License Keys** page, click **Add**.
 - b. On the **General** tab, provide values for the following fields:
 - **License Key Name:** Unique name.
For example: otdctmlicense
 - **Resource ID:** License linked to the resource.
 - c. On the **License Key** tab, click **Get License File**, browse and select the license file.
 - d. Click **Save**.
10. Create a **businessadmin** user in the **otds.admin** partition using the following steps:
 - a. Click **Partitions**.
 - b. Click **otds.admin > Actions > View Members..**
 - c. Click **Add > New User**.
 - d. On the **General** page, in the **User Name** box, type a name for this user as **businessadmin**.
 - e. On the **Account** page, in the **Password Options** area, do the following:
 - i. Click **Do not require password change on reset** from the list.
 - ii. Clear the **User cannot change password** check box, if selected.
 - iii. Select the **Password never expires** check box and click **Save**.

Additional information and tasks

Documentum CM client

Create OTDS users with the user name as **d2_mail_manager** and **d2_wf_notification_user** and keep the passwords as blank in a new non-synchronized partition which is not associated to any resources in OTDS.

By default, OTDS is enabled for client configuration. If you disable OTDS, you must add another user with the user name as **install_owner_user** to the same partition with the password as blank.

In the **Password Options** area, click **Do not require password change on reset** from the list, and do the following:

- Select the **User cannot change password** check box.
- Select the **Password never expires** check box.

This is applicable only for the system account user partition.

Workflow Designer

If you want to use Workflow Designer with the skip SSO feature, only a user with the **install_owner** privilege can access without a license.

Advanced Workflow

Advanced Workflow is available with advanced license or as an add-on. As a user, you can build processes using Advanced Workflow but to install these processes, the xDA Documentum CM repository endpoint user must have advanced Documentum CM or an add-on license.

To start a workflow at runtime from any Documentum CM client components, you must have advanced Documentum CM or an add-on license and the required transaction capability. The transaction counter increments at runtime when a user creates a new instance of workflow irrespective of the state of the workflow.

Reports

Create an OTDS user with the user name as `dctmreports` and keep the password as blank.

In the **Password Options** area, click **Do not require password change on reset** from the list, and do the following:

- Select the **User cannot change password** check box.
- Select the **Password never expires** check box.

OpenText Documentum CM for Microsoft 365, OpenText Documentum CM Online Editing Service, and Notification Service

Create an OTDS user with the user name as `M365_SERVICE` with the password as `Xecmserviceuser@123`. The password is case-sensitive.

The password is updated when the notification service is started.

In the **Password Options** area, click **Do not require password change on reset** from the list, and do the following:

- Clear the **User cannot change password** check box.
- Select the **Password never expires** check box.

Documentum Archive Services for SAP Solutions

Create an OTDS user with the user name as `installownerusername` with the password as `installownerpassword`. This is a service account.

The system automatically changes the password after the service is started.

11. Add the `businessadmin` user to the `otdsbusinessadmins` group in OTDS using the following steps:
 - a. Click **Users & Groups**, and select the **Groups** tab.
 - b. In the **Search** box, type `otdsbusinessadmins` to find the `otdsbusinessadmins@otds.admin` group.
 - c. On the **Groups**, click **Actions > Edit Membership**.

- d. On the **otdsbusinessadmins@otds.admin** page, on the **Members** tab, click **Add Member**.
 - e. In the **Search** box, type **businessadmin** to find the **businessadmin@otds.admin** member.
 - f. Select the **businessadmin@otds.admin** check box, and click **Add Selected**.
12. In the Documentum CM Server machine, run the following command in IAPI to create the **dm_otds_license_config** object:

```
create,c,dm_otds_license_config
set,,l,otds_url
<otds_url_including_rest>
set,,l,license_keyname
<license_keyname>
set,c,l,business_admin_name
<business_adminname>
set,c,l,business_admin_password
<password>
save,c,l
```

For example:

```
create,c,dm_otds_license_config
set,c,l,otds_url
http://documentumcm:8080/otdsws/rest
set,c,l,license_keyname
otdctmlicense
set,c,l,business_admin_name
businessadmin
set,c,l,business_admin_password
Password-1234567890
save,c,l
```



Notes

- Alternatively, you can create the **dm_otds_license_config** object using the Documentum Administrator graphical user interface.

For instructions, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

- If you modify the **dm_otds_license_config** object after the first time, you must run the **apply,c,NULL,FLUSH_OTDS_CONFIG** command in IAPI.

13. Allocate a license in OTDS to a partition using the following steps:

- a. Click **Partitions > <your desired partition> > Actions > Allocate to License**.
- b. On the **Allocate to License** page, click the relevant counter from the list.



Note: You can also allocate the license to users and groups. When you allocate a license, ensure that you allocate the relevant counter to a user or group.

- c. Click **Allocate to License**.



Notes

- If you modify the allocated license file after the initial deployment, you must restart OTDS.

- You can allocate users to your license any time, but a user must exist before you can allocate the user to a license. The changes to the allocation, deallocation, and revocation take effect after 24 hours for releases prior to and including the 24.4 release.
 - For the 25.2 release, run the following command in IAPI to get the allocation, deallocation, revocation, or deletion with immediate effect:

```
apply,c,NULL,FLUSH_JMS_OTDS_CACHE
```

- From the 25.4 release, run the following command in IAPI to get the allocation, deallocation, revocation, or deletion with immediate effect:

```
apply,c,NULL,FLUSH_OTDS_CACHE
```

14. Import all the inline and Lightweight Directory Access Protocol (LDAP) users to OTDS:

- On Windows:

In the Documentum CM Server machine, copy the dfc.properties file to the \$DOCUMENTUM\Shared folder. At the command prompt, go to the \$DOCUMENTUM\Shared folder, and then run the following command:

```
java --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/
java.io=ALL-UNNAMED --add-opens=java.base/java.util=ALL-UNNAMED --add-
opens=java.base/java.util.concurrent=ALL-UNNAMED --add-opens=java.rmi/
sun.rmi.transport=ALL-UNNAMED --add-exports=java.base/
sun.security.provider=ALL-UNNAMED --add-exports=java.base/
sun.security.pkcs=ALL-UNNAMED --add-exports=java.base/sun.security.x509=ALL-
UNNAMED --add-exports=java.base/sun.security.util=ALL-UNNAMED --add-
exports=java.base/sun.security.tools.keytool=ALL-UNNAMED -
cp ..\dfc.jar:dfc.properties:* com.documentum.fc.tools.MigrateInlineUsersToOtds
<repository name> <installation owner user name> <installation owner password>
<non-synchronized partition name>
```

- On Linux:

In the Documentum CM Server machine, copy the dfc.properties file to the \$DOCUMENTUM\dfc folder. At the command prompt, go to the \$DOCUMENTUM \dfc folder, and then run the following command:

```
java --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/
java.io=ALL-UNNAMED --add-opens=java.base/java.util=ALL-UNNAMED --add-
opens=java.base/java.util.concurrent=ALL-UNNAMED --add-opens=java.rmi/
sun.rmi.transport=ALL-UNNAMED --add-exports=java.base/
sun.security.provider=ALL-UNNAMED --add-exports=java.base/
sun.security.pkcs=ALL-UNNAMED --add-exports=java.base/sun.security.x509=ALL-
UNNAMED --add-exports=java.base/sun.security.util=ALL-UNNAMED --add-
exports=java.base/sun.security.tools.keytool=ALL-UNNAMED -
cp ..\dfc.jar:dfc.properties:* com.documentum.fc.tools.MigrateInlineUsersToOtds
<repository name> <installation owner user name> <installation owner password>
<non-synchronized partition name>
```

15. Configure the OTDS authentication using the following steps:

- a. In the Documentum CM Server machine, open the otdsauth.properties file in the \$DM_HOME\OTDSAuthLicenseHttpServerBin\config folder.
- b. Update the values for the following variables:

- certificate

Run the following URL to obtain the certificate value:

```
http://<OTDS server name>:8080/otdswebs/rest/systemconfig/certificate_content
```

- otds_rest_credential_url

For example: otds_rest_credential_url=http://192.168.2.105:8080/otdswebs/rest/authentication/credentials

- otds_rest_ticket_url

For example: otds_rest_ticket_url=http://192.168.2.105:8080/otdswebs/rest/authentication/resource/validation

- admin_username

For example: admin_username=dmadmin

16. To verify the license configuration, sign in to any deployed application (for example, Documentum Administrator) with any licensed user.

Ensure that user authentication is successful.

2.7.1.3 Creating new users, allocating license, and applying roles in OTDS

1. Go to **Partitions** > *<partition name>* > **Actions** > **View Members** > **Add** > **New user**.
2. Create a new user with a desired name (for example, otdctmuser), and set all the attributes including password.
3. Click **Save**.
4. Select the user you created and click **Actions** > **Allocate to License**.
5. On the **Allocate to License** dialog box, select the relevant counter and click **Allocate to License**.
6. Select the user you created and click **Actions** > **Edit Application Roles**.
7. Click **Assign Roles** and select a desired role for the user.

For more information about the list of roles, see [step 8.c.](#)



Note: Ensure da_privilege_enabled=T and lss_cc_enabled=T are added in the <Java Method Server Home>/webapps/dmotdsrest/WEB-INF/classes/dmotds.properties file in the Documentum CM Server machine.

8. Click **Add Selected** and then click **Close**.

2.7.1.4 Troubleshooting license configuration

The following table describes the license-related errors captured in the \$DOCUMENTUM/dba/log/otdsauth.log file.

Error code	Description	Solution
DM_LICENSE_E_NO_LICENSE_CONFIG	The license configuration is not created in the repository.	Ensure that you activate the license from IAPI or Documentum Administrator.
DM_LICENSE_E_CONFIG_MISSING_PARAMS	One of the following mandatory parameters is not available: OTDS URL, or OTDS business administrator credentials, or License key	Ensure that you have provided valid values in IAPI or Documentum Administrator.
DM_LICENSE_E_SERVER_NOTREACHABLE	The OTDS URL is not reachable.	Ensure that the OTDS installation is active and reachable.
DM_LICENSE_E_BAD_ADMIN_CRED	OTDS business administrator credentials are incorrect or locked.	Before you activate the license, ensure that you have provided valid values in IAPI or Documentum Administrator.
DM_LICENSE_E_REQ_BUSINESS_ADMIN	The OTDS business administrator user is not added to the otdsbusinessadmins group in OTDS.	Ensure that you add the business administrator user to the otdsbusinessadmins group in OTDS.
DM_LICENSE_E_NO_LICENSE	The license key is not configured with the license file in OTDS.	Ensure that you upload the license file in OTDS to generate the license key and provide the correct license key.
DM_LICENSE_E_INVALID_LICENSE	The license file is invalid.	Ensure that you have uploaded a valid license file in OTDS.
DM_LICENSE_E_USER_NOT_FOUND_OR_DUPLICATE	The user is not found in OTDS or the user is not unique in OTDS.	Ensure that the user exists in OTDS and is unique.
DM_LICENSE_E_USER_NO_LICENSE_ALLOCATED	The user is not allocated with a counter.	Ensure that you allocate a counter to the user in OTDS.
DM_LICENSE_E_USER_NO_ACCESS	The user is not allocated to the relevant counter or all the licenses for users or transactions are consumed.	Ensure that you allocate the user to the relevant counter in OTDS.

Error code	Description	Solution
DM_LICENSE_E_UNEXPECTED_ERROR	There is an unexpected error in the licensing code.	Analyze the <code>otdsauth.log</code> file to troubleshoot the issue.
DM_LICENSE_E_SYSTEMACCT_MISUSE	When a system account user is allocated to any other counters.	Remove the <code>SYSTEMACCT</code> or other counters.
DM_LICENSE_E_SERVICEACCT_MISUSE	When a service account user is allocated to any other counters.	Remove the <code>SERVICEACCT</code> or other counters.

2.7.2 Configuring Java Method Server for certificate-based SSL communication

Install Documentum CM Server in Secure (certificate-based) mode. After the Documentum CM Server installation is complete, stop the Java Method Server service and proceed with configuration for certificate-based SSL communication manually.

For more information about configuring Java Method Server for certificate-based SSL communication, see *Apache Tomcat* documentation.

1. Enable the SSL connector and configure the keystore, keystore password, and cipher suite attributes appropriately in the `$CATALINA_BASE/conf/server.xml` file.

For example:

```
<Connector port="9553"
    protocol="org.apache.coyote.http11.Http11NioProtocol"
    maxThreads="200"
    scheme="https"
    SSLEnabled="true"
    secure="true">
<SSLHostConfig ciphers="TLS_RSA_WITH_AES_128_CBC_SHA">
<Certificate certificateKeystoreFile=
"C:\Documentum\tomcat\conf\<self-signed certificate>.p12"
certificateKeystorePassword="<keystore password>"/>
</SSLHostConfig>
</Connector>
```

2. Generate the self-signed certificates and keys for Java Method Server and Documentum CM Server and then store them in the respective keystore file. The keystore file must in the PKCS#12 format.

Use the following example command format to generate:

```
keytool -genkeypair -alias <self-signed certificate> -keyalg RSA -keysize 4096 -
validity 720 -keystore C:\Documentum\tomcat\conf\<self-signed certificate>.p12 -
storepass <keystore password> -keypass <keystore password>
```

Use the following example command format to extract the certificate from keystore:

```
keytool -exportcert -alias <self-signed certificate> -file C:\Documentum\tomcat\conf\<self-signed certificate>.cer -keystore C:\Documentum\tomcat\conf\<self-signed certificate>.p12 -storepass <keystore password>
```

3. Create the trust store in the PKCS#7 format from the Java Method Server certificate.

Use the following example command format:

```
openssl x509 -inform der -in C:\Documentum\tomcat\conf\<self-signed
certificate>.cer -out jms.pem
openssl crl2pkcs7 -nocrl -certfile broker crt.pem -certfile jms.pem -outform der -
out server-trust.p7b
```

4. Configure the cipherlist and trust store created in step 3 in the `server.ini` file.

For example, copy the `server-trust.p7b` trust store file to <location where Documentum CM Server is installed>\dba\secure.

For more information about the cipherlist and trust store, see *OpenText Documentum Content Management - Server Administration and Configuration Guide (EDCCS250400-AGD)*.

5. Modify the `dm_jms_config` object, update `base_uri`, and `supported_protocol` for the corresponding application.

For example:

```
set,c,1,base_uri[0]
https://cswinsql58.otxlab.net:9553/DmMethods/servlet/DoMethod
set,c,1,base_uri[1]
https://cswinsql58.otxlab.net:9553/XMLStoreService/servlet/XMLStoreServiceServlet
set,c,1,base_uri[2]
https://cswinsql58.otxlab.net:9553/DmMail/servlet/DoMail
set,c,1,supported_protocol[0]
https
set,c,1,supported_protocol[1]
https
set,c,1,supported_protocol[2]
https
save,c,1
```

6. Start the Java Method Server.

The Java Method Server starts successfully.

2.7.2.1 Supported SSL communication configurations

This table contains the information on the supported SSL communication configurations:

Documentum CM Server/ Connection Broker/ Foundation Java API	Java Method Server	Client
Non-anonymous	Non-anonymous	Non-anonymous
Anonymous	Anonymous/Non-anonymous	Non-anonymous

In case of a combination of Java Method Server:

- Java Method Server 1 and Java Method Server 2 can be configured for anonymous communication with Documentum CM Server.

- Java Method Server 1 and Java Method Server 2 can be configured for non-anonymous communication with Documentum CM Server.
- When using client such as Documentum Administrator, if Java Method Server is configured for non-anonymous SSL communication with Documentum CM Server, you can find the *dfc.security.ssl.use_existing_truststore* parameter in *dfc.properties*. If set to true, import all certificates (Documentum CM Server, connection broker, and Java Method Server) in default Java trust store on the client. If set to false, provide the *dfc.security.ssl.truststore* and *dfc.security.ssl.truststore_password* parameters in *dfc.properties* on the client and import all certificates (Documentum CM Server, connection broker, and Java Method Server) in Foundation Java API trust store.



Notes

- The Distributed Content components (clients such as Accelerated Content Services, OpenText Documentum Content Management (CM) Branch Office Caching Services, and OpenText Documentum Content Management (CM) Messaging Service) in anonymous mode is not supported.
- The following Java Method Server and SSL mixed mode configurations are not supported:
 - JMS1 in non-anonymous SSL mode, JMS2 in anonymous SSL mode, and Documentum CM Server in anonymous SSL mode.
 - JMS1 in non-anonymous SSL mode, JMS2 in both (anonymous and non-anonymous) SSL mode, and Documentum CM Server in non-anonymous SSL mode.
 - JMS1 in non-anonymous SSL mode, JMS2 in native mode, and Documentum CM Server in non-anonymous SSL mode.

2.7.3 Changing the connection mode from native to secure

You can change the connection mode from a native to a secure connection to enable SSL encryption for traffic between Foundation Java API and Documentum CM Server. Follow the instructions in this section to change the secure connection mode from native to secure connection to enable SSL at the Documentum CM Server level.

- Update the dm_server_config object:

- Connect to IAPI as installation owner.
- Run the following commands:

```

API> retrieve,c,dm_server_config
(This command retrieves the object ID.)
API> dump,c,1
(Verify that the secure_connect_mode has changed to
native by viewing the list.)
API> set,c,<objectID>,secure_connect_mode
SET> secure
API> save,c,1
API> dump,c,1
  
```

(Verify that the secure_connect_mode has changed to secure by viewing the list.)

2. Update the dfc.properties file.
 - a. Navigate to the \$DOCUMENTUM/config directory
 - b. Back up the original dfc.properties file.
 - c. Change the entry dfc.session.secure_connect_default=try_native_first to dfc.session.secure_connect_default=secure.
 - d. Save the dfc.properties file.
3. Update the docbroker.ini file.
 - a. Navigate to the \$DOCUMENTUM/dba directory.
 - b. Back up the original docbroker.ini file.
 - c. Change the entry secure_connect_mode=<blank> to secure_connect_mode=secure.
 - d. Save the docbroker.ini file.
4. Restart the Documentum CM Server.
 - Restart **Master Service** (from Services).
 - Wait until all OpenText Documentum CM services have started, (repository, connection broker, Java Method Server and Master Service).
5. Verify if the connection broker and repository are connecting to a secure port.
 - a. Navigate to the \$DOCUMENTUM/dba/logs directory.
 - b. View the connection broker and repository logs and verify that they have information about secure port 1490.

2.7.4 Configuring PostgreSQL

2.7.4.1 Creating clustered indexes

When a table is clustered, it is physically reordered based on the index information. Clustering is a one-time operation. When the existing table is subsequently updated, the changes are not clustered. No attempt is made to store new or updated rows according to their index order. So, recluster by issuing the CLUSTER command again. To achieve this, expose a new parameter, DM_CLUSTER_INDEX to the POST_UPGRADE_ACTION apply method. When this is called with the table_name, force clustering is applied to that table. If table_name is not specified, force clustering is applied to all tables.

Use the following apply method:

```
apply,c,NULL,POST_UPGRADE_ACTION,EXECUTION_MODE,S,  
DM_CLUSTER_INDEX,TABLE_NAME,S,dm_location_s,TRACE_ON,B,T,  
FORCE_FLAG,B,F
```

2.7.4.2 Enabling data partition in PostgreSQL database

- Run the following command in DQL:

```
EXECUTE partition_operation WITH "operation"='create_scheme',
"partition_scheme"='<name of the scheme>',
"partition_name"='<name of the first partition, for example P1>',
"range"=<specify the range, for example 100>,
"tablespace"='DM_<repository name>_DOCBASE',
"partition_name"='<name of the second partition>', "range"=<specify the range>,
"tablespace"='DM_<repository name>_DOCBASE',
"partition_name"='<name of the nth partition>', "range"=<specify the range>,
"tablespace"='DM_<repository name>_DOCBASE'
```

- Run the following command in DQL:

```
EXECUTE partition_operation WITH "operation"='db_partition',
"partition_scheme"='<name of the scheme>'
```

- Run the following command in DQL:

```
EXECUTE get_file_url FOR <object id generated from step 2> WITH
"format"='text'
```

- Run the following command in IAPI:

getpath,<session id>, <object id generated from step 2>This generates the database script to enable the data partition.

- Stop the Documentum CM Server.

- Connect to the PostgreSQL database as a repository owner and run the script generated from step 4.

- After enabling the data partition, run the following database queries:

```
drop view dm_resync_dd_attr_info;
drop view dm_resync_dd_type_info;
drop view dm_dd_policies_for_attrs;
drop view dm_dd_policies_for_type;
delete from dmi_vstamp_s where i_application='dm_dd_attr_info_view_tag' or
i_application='dm_dd_type_info_view_tag' or
i_application='dm_dd_policies_views_stamp';
```

- Start the Documentum CM Server.



Notes

- All the supertype partitions are inherited to the current type. If none of the supertypes are partitioned, then you have to enable data partition for the corresponding type.

For example:

```
create type <type name> (<name of the attribute 1> <type>,
<name of the attribute n> <type>) with supertype dm_sysobject
```

The created type inherits all the partitioned properties from dm_sysobject.

- Creating a partitionable type is not supported. As an alternative, you can create a custom type object and enable data partition only to this custom type using EXECUTE partition_operation with db_partition by specifying the value for the type_name attribute. For example:

```
create type <type name, for example 'example_type'>
(<name of the attribute 1> <type>, <name of the attribute n>
<type>) with supertype NULL
```

```
EXECUTE partition_operation WITH "operation"='db_partition',
"partition_scheme"='<scheme name>',type_name='example_type'
```

Continue from [step 3 of the “Enabling data partition in PostgreSQL database” on page 103](#) procedure to create a partitionable type.

Also, you may encounter missing `i_partition` attribute for custom type objects after partitioning. To fix, after the data partitioning script is run against the database server successfully, restart the Documentum CM Server and then run the following DQL statement to complete the type partitioning process:

```
ALTER TYPE <type_name> ENABLE PARTITION
```

- Limited support for data partition.

2.7.4.3 Tuning PostgreSQL database

1. Append the following lines at the end of `<path>\<to>\postgresql.conf`:

For example:

```
temp_buffers = 32MB
work_mem = 32MB
checkpoint_segments = 32
checkpoint_timeout = 10min
checkpoint_completion_target = 0.5
random_page_cost = 2.0
default_statistics_target = 500
maintenance_work_mem = 256MB
shared_buffers = <1/4 times of physical memory>
effective_cache_size = <3/4 times of physical memory>
wal_buffers = 16MB
synchronous_commit = off
```

2. Restart the PostgreSQL instance.
3. Login on the `psql` command line prompt and check for modified parameter to verify if the changes are applied as:

```
# show random_page_cost;
```

4. Create new pattern indexes:

- `dm_sysobject_s`:

```
create index test_ops_indx on dm_sysobject_s (r_object_id bpchar_pattern_ops);
```

- `dm_sysobject_r`

- `dm_acl_s`:

```
create index test_dm_acl_s_indx on dm_acl_s (r_object_id bpchar_pattern_ops);
```

5. Execute `cluster` for clustered indexes maintenance:

```
# cluster;
```

6. Recreate all indexes in the database to remove index fragmentation:

```
# reindex database "dm_DOCREPO_docbase";
```

7. Execute *vacuum full* and analyze to remove table fragmentation and also update database statistics:

```
# VACUUM FULL VERBOSE ANALYZE;
```

8. Run the query to check dead tuples in the database:

```
SELECT psut.relname,
       to_char(psut.last_vacuum, 'YYYY-MM-DD HH24:MI') as last_vacuum,
       to_char(psut.last_autovacuum, 'YYYY-MM-DD HH24:MI') as last_autovacuum,
       to_char(pg_class.reltuples, '9G999G999G999') AS n_tup,
       to_char(psut.n_dead_tup, '9G999G999G999') AS dead_tup,
       to_char(CAST(current_setting('autovacuum_vacuum_threshold') AS bigint)
              + (CAST(current_setting('autovacuum_vacuum_scale_factor') AS numeric)
                 * pg_class.reltuples), '9G999G999G999') AS av_threshold,CASE
WHEN CAST(current_setting('autovacuum_vacuum_threshold') AS bigint)
     + (CAST(current_setting('autovacuum_vacuum_scale_factor') AS numeric)
        * pg_class.reltuples) < psut.n_dead_tup
THEN '*'
ELSE ''
END AS expect_av
FROM pg_stat_user_tables psut
JOIN pg_class ON psut.relid = pg_class.oid
ORDER BY 1;
```

You should not see a huge number for the *dead_tup* column.

9. Run the query to check index fragmentation:

```
with indexBloat as
(
SELECT nspsname as schema, c.relname as table_name,
       i.relname as index_name,
       ROUND(ROUND(100 * pg_relation_size(indexrelid) / pg_relation_size(indrelid), 2)
            / 100, 2) AS iratio,
       pg_size_pretty(pg_relation_size(indexrelid)) as index_size,
       pg_size_pretty(pg_relation_size(indrelid)) AS table_size,
       pg_relation_size(indexrelid) as isize_byte
FROM pg_index x
JOIN pg_class c ON c.oid = x.indrelid
JOIN pg_class i ON i.oid = x.indexrelid
JOIN pg_namespace n ON (n.oid = c.relnamespace)
WHERE nspsname NOT IN ('pg_catalog', 'information_schema', 'pg_toast')
AND i.relkind = 'i'
AND c.relkind = 'r'
AND pg_relation_size(indrelid) > 0
)
SELECT *
FROM indexBloat WHERE schema != 'snapshots'
ORDER BY isize_byte desc;
```

For any index (> 50MB), the *iratio* should be less than 80%. Only if the index is composite, the *iratio* should be close to 80-85% and not higher than the table itself. Ignore empty or small tables (8KB or few KB tables).

2.7.5 Backing up keystore files

After you install the Documentum CM Server and repository, back up the keystore files located in the \$DOCUMENTUM/dba/secure directory. Make sure that you back up and save the aek.key file.

If the key becomes corrupted and you do not have a backup, your repository cannot be started, and you cannot access encrypted files. Backing up the external keystore is a best practice and must be done in addition to backing up the aek.key file.

2.7.6 Changing the default passphrase

During Documentum CM Server installation, a keystore is created that contains a passphrase that is used for encryption. After installation, you can change the default passphrase to a custom passphrase ensuring that you comply with the password complexity rules (see “[Password complexity rules](#)” on page 52). If you create a custom passphrase after Documentum CM Server installation, any time you restart the server host, you need to run the dm_crypto_boot utility. For more information about the instructions and details on encryption, keystores, and passphrases, see *OpenText Documentum Content Management - Server Administration and Configuration Guide* (EDCCS250400-AGD).

2.7.7 Binding Documentum CM Server to a network card

To configure Documentum CM Server to use a different network card, create an initialization file for the connection broker. The file must include a [DOCBROKER_CONFIGURATION] section to identify the IP address of the network card. Use the following format:

```
[DOCBROKER_CONFIGURATION]
host=<IP_address_string>
service=<service_name>
port=<port_number>
```

where <IP_address_string> is the IP address of the network card.

The service name is the connection broker's service name, defined in the host's services file. The port number is the port defined in the service.

- If you include a service name, the connection broker starts by using that service.
- If you include a port number, the connection broker starts by using that port.
- If you do not include a service name or a port number, the connection broker uses the default port number 1489. If you are using the default port number, make sure that the next port number (1490) is available for use because the connection broker requires that two ports be reserved.

For more information about binding Documentum CM Server to a network card, see *OpenText Documentum Content Management - Administrator User Guide* (EDCAC250400-UGD).

2.7.8 Installing Documentum CM Server client

Install the following Documentum CM Server client applications:

- Documentum Administrator, the primary web-based administrative client tool for configuring and administrating Documentum CM Server and repositories. For more information about Documentum Administrator, see *OpenText Documentum Content Management - Server and Server Extensions Installation Guide (EDCSY250400-IGD)*.
- Documentum Webtop, a web-based client application for accessing and managing content in the repository.

2.7.9 Installing Documentum CM Server using an existing database account

When you select the option to use an existing database account while configuring repository, you can proceed only if you have a database administrator account.

If you cannot provide the database administrator account at this stage, you can perform these actions after the repository configuration is complete:

- Create the Data Management System user in the database instance.
- Grant the Data Management System user permissions to access the Data Management System schema tables.

2.7.10 Using the migration utility

The migration utility is used for the following:

- Changing the repository ID
- Changing the repository name
- Changing the server configuration name
- Changing the host name of the Documentum CM Server machine
- Changing the installation owner

The following files are available in the `<$DM_HOME>\install\external_apps\MigrationUtil` folder:

- `MigrationUtil.jar`
- `config.xml`
- `MigrationUtil.sh` and `MigrationUtil.csh` (Linux)
- `MigrationUtil.bat` and `WinOSService.dll` (Windows)

Make sure that you perform the following before running the migration utility:

- Take a complete backup of Documentum CM Server installation manually. Also, for Windows, take a backup of the registry files. The migration utility, specifically changing the repository ID is database intensive. Make sure that you take a backup of the database manually.
- Download all the required JDBC drivers in your file system. (For example, `ojdbc.jar` (Oracle) and `sqljdbc.jar` (SQL Server))
- Append the class path location of the JDBC drivers in the `MigrationUtil.bat` (Windows) or `MigrationUtil.sh` or `MigrationUtil.csh` (Linux).
- Stop all OpenText Documentum CM services such as the connection broker, repository, Java Method Server, and so on.
- In `config.xml`, provide the inputs for the following parameters:
 - `dbms`: Database on which utility is run.
 - `tgt_database_server`: IP address of the target database server.
 - `port_number`: Listening port number of the database.
 - `InstallOwnerPassword`: Password of the installation owner.
 - `DocbaseName.1`: Name of the existing repository. If you have more than one repository, provide the list of repository names.
 - `DocbasePassword.1`: Password of the existing repository owner. If you have more than one repository, provide the list of repository owner passwords.
 - `isRCS`: Set to `<yes>` if you are running the migration utility on the secondary Documentum CM Server. Otherwise, set to `<no>`.
 - `vaultEnabled`: Set to `<yes>` if HashiCorp Vault is configured in Documentum CM Server. If set to `<yes>`, you must provide a HashiCorp Vault secret for `DocbasePassword`, `InstallOwnerPassword`, and `NewInstallOwnerPassword` instead of plain passwords.
Otherwise, set to `<no>`.
 - `dsis_url`: URL to connect to the DSIS daemon agent. The format is `http://localhost:<DSIS daemon connection port>/dsis`. This value is ignored if HashiCorp Vault is not configured.
 - `dsis_token`: Token to authenticate with the DSIS daemon agent. Set the value to the same value provided for `dsis.dctm.token`. This value is ignored if HashiCorp Vault is not configured.

The migration utility must be run on the target Documentum CM Server host after the successful Documentum CM Server migration from source to the target version.



Notes

- The migration utility supports Documentum CM Server only.
- The migration utility does not support custom types created on OOTB Documentum CM Server installation except for using the utility for changing

the repository ID. Administrator must update the custom types that has any references to repository name, server configuration name, host name, and installation owner.

- After running the migration utility, the log files are generated in the `MigrationUtil_logs` folder. Review the log files and make sure that there are no errors. In case of errors, try and fix it manually. If you are not able to fix the error(s) manually then restore the backup version of databases and Documentum CM Server, fix the errors, and then run the utility again.

2.7.10.1 Changing the repository ID

The migration utility facilitates the change in `Repository ID` of each repository in the Documentum CM Server installation. You can change only one `Repository ID` for each time the utility is run. Running the utility is both time and space consuming. For example, if you have 10 million objects you may need a 50 GB disk space because of large database transactions. With the change in repository name, host name, and installation owner along with this, the entire process results in cloning of the existing repository into another. Each repository runs as an independent identity. The change to `Repository ID` affects all metadata associated with the repository objects.

In the `config.xml` file, provide the inputs for the following parameters:

- `ChangeDocbaseID`: Set to `<yes>`.
- `Docbase_name`: Same name as specified for `DocbaseName.1`.
- `NewDocbaseID`: New Repository ID.

2.7.10.2 Changing the repository name

The migration utility facilitates the change in `Repository Name` of each repository in the Documentum CM Server installation.

In the `config.xml` file, provide the inputs for the following parameters:

- `ChangeDocbaseName`: Set to `<yes>`.
- `NewDocbaseName.1`: Provide an unique repository name.



Note: You can change the repository name for multiple repositories. For example, provide a new name for `NewDocbaseName.2` for the corresponding `DocbaseName.2`.

2.7.10.3 Changing the server configuration name

The migration utility facilitates the change in Server Configuration Name of each repository in the Documentum CM Server installation.

In the config.xml file, provide the inputs for the following parameters:

- ChangeServerName: Set to <yes>.
- NewServerName.1: Provide a new server name.



Note: You can change the server configuration name for multiple repositories. For example, provide a new name for NewServerName.2 for the corresponding DocbaseName.2.

2.7.10.4 Changing the host name of the Documentum CM Server machine

The migration utility facilitates the change in host name of the Documentum CM Server machine with multiple repositories.

Make sure the following before running the migration utility:

- Change the host name of the Documentum CM Server machine.
- List all the repositories of Documentum CM Server in config.xml.

In the config.xml file, provide the inputs for the following parameters:

- ChangeHostName: Set to <yes>.
- HostName: host name of the Documentum CM Server machine you manually changed as a prerequisite step before running the migration utility.
- NewHostName: Name of the new host name you changed.

2.7.10.5 Changing the installation owner

The migration utility facilitates the change in Installation Owner of the Documentum CM Server installation with multiple repositories.

Make sure the following before running the migration utility:

- Create a new installation owner account with similar privileges as the existing installation owner.
- Documentum CM Server is installed in a common location accessible to all with relevant permissions.
- List all the repositories of Documentum CM Server in config.xml.

In the config.xml file, provide the inputs for the following parameters:

- ChangeInstallOwner: Set to <yes>.

- InstallOwner: Name of the existing installation owner from server.ini of the repositories.
- NewInstallOwner: Name of the new installation owner you created.
- NewInstallOwnerPassword: Password for the new installation owner.

After changing the installation owner, on Linux, run \$DOCUMENTUM/dba/dm_root_task_cs16.4 as root.

Notes

- Changing the installation owner for domain user does not provide the permission for the Documentum folder. Run the following commands manually:

```
icacls <Documentum>/grant <domainuser>:(OI)(CI)F /t /q
icacls C:\Documentum /grant <domainuser>:F /t /q
```

- Perform the following steps manually to change member repository references in the host to successfully run the federation job:
 1. Change r_member_docbases attribute of dm_federation object.
 2. Rename .cnt files for member repositories at the %DOCUMENTUM%\dba\config\<governing_docbase>*.cnt location.
 3. Change object name of jobs referring to older member repositories and Master and replica-docbases in method_arguments. For example:

```
<select r_object_id,object_name from dm_job
where object_name like 'dm_ACLRep1%'>
```

4. Change references of member repositories in federation.cnt at * %DOCUMENTUM%\dba\config\<governing_docbase>.

2.7.10.6 Running the migration utility

For Windows, run the MigrationUtil.bat file.

For Linux, run the MigrationUtil.sh or Migration Util.csh file.

2.7.11 Configuring Documentum CM Server to install DAR using Documentum Administrator

Run the following commands using IAPI in Documentum CM Server:

```
retrieve,c,dm_server_config
append,c,1,app_server_name
install_dar
append,c,1,app_server_uri
http://<IP address of Documentum CM Server>:9080/DmMethods/servlet/InstallDar
save,c,1
```

For more information about installing DAR files using Documentum Administrator, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

2.7.12 Enabling xPlore search for new emails



Note: This procedure is applicable only for xPlore 20.2 and earlier versions.

To enable the xPlore search for emails imported using MailApp.dar, perform the following:

1. Run the following query in the repository:

```
ALTER ASPECT mdmo_message_aspect FULLTEXT SUPPORT ADD ALL
```
2. Stop the xPlore service in the index machine.
3. Clear the BOF cache at <xPlore Home>\<WildFly_version>\server\DtcmServer_Indexagent\data\Indexagent\cache\Documentum CM Server version\bof\repository_name.
4. Start the xPlore service in the index machine.
5. Start the indexing using Documentum Administrator.

2.7.13 Getting Started with Documentum CM Server

2.7.13.1 Starting or stopping Documentum CM Server on Linux

To start Documentum CM Server on Linux:

1. Start the connection broker by running <\$DOCUMENTUM>/dba/dm_launch_<docbrokerName>, where <docBrokername> is the name of the connection broker.
2. Start the repository by running <\$DOCUMENTUM>/dba/dm_start_<serverconfigname>, where <serverconfigname> is the object name of the Documentum CM Server server config object, which consists of the host name and the repository name.
3. Start the application server by running <\$DM_JMS_HOME>/bin/startMethodServer.sh.

To Stop Documentum CM Server on Linux:

1. Stop the application server by running <\$DM_JMS_HOME>/bin/stopMethodServer.sh.
2. Stop the repository by running <\$DOCUMENTUM>/dba/dm_shutdown_<serverconfigname>, where <serverconfigname> is the object name of the Documentum CM Server server config object, which consists of the host name and the repository name.
3. Stop the connection broker by running <\$DOCUMENTUM>/dba/dm_stop_<docbrokerName>, where <docBrokername> is the name of the connection broker.

2.8 Configuring operation mode

OpenText Documentum CM is designed to run as schema owner (Documentum CM Server database user) during both administration and daily operation modes.

Starting with the 16.7 release, privileges are categorized as schema owner (administration) and the normal database user with restricted privileges (operation mode) for the Oracle database only. This is used for performing the daily operations, such as DML operations using normal database user with restricted privileges. Schema owner account is used during the maintenance window such as upgrade, migration, and so on.

Privileges associated with the restricted user are:

- CONNECT that is used for creating the database session.
- DML (SELECT, UPDATE, INSERT, DELETE) that is used on all the tables and views except the INDEX tables.
- Unlimited quota for TABLESPACE.
- Available on the USERS tablespace.



Notes

- All DDLs associated with CREATE, DROP, ALTER keywords are not allowed.
- Schema owner account is inactive and locked during the operation window.
- Activate and unlock the schema owner account during the maintenance window.
- When the repository owner grants the index tables, it may result in the ORA-01720 error. You can ignore this error.

2.8.1 Setting up restricted user for operation mode

The authentication using password and without using password is used for establishing a session between Documentum CM Server and database (support for Oracle only).

For example:

1. You can create the restricted user using the dmdbtest utility or directly in the database.
 - If you want to create the restricted user through the dmdbtest utility, run the utility using the following arguments:

```
dmdbtest
-M<restrictedusermode>
-S<connection_string>
-U<docbase_owner>
-P<Password>
-R<Restricted_user_name>
```

```
-r<restricted_user_password> // Optional and only for authentication using
password
-u<system_user>
-p<system_user_password>
```

- If you have already created the user in the database, then run the dmdbtest utility using the following arguments:

```
dmdbtest
-Mrestrictedusermode
-S<connection_string>
-U<docbase_owner>
-P<Password>
-R<Restricted_user_name>
```

Example for authentication using password:

```
dmdbtest
-Mrestrictedusermode
-SORCL
-Utestenv
-Pn0lif3
-Rrestricteduser
-rpassword_123
-usystem
-ppassword
```

Example for authentication without using password and using Kerberos as a security protocol:

```
dmdbtest
-Mrestrictedusermode
-SORCL
-Utestenv
-Pn0lif3
-Rrestricteduser
-usystem
-ppassword
```

After the dmdbtest is successful, update server.ini with the following attributes:

```
database_user = <restricted_user_name>
database_auth_type = 1 // for authentication without password
database_password_file // remove this entry for authentication without password
```

2. To authenticate using password, take a backup of the existing dbpasswd.txt file and create an empty dbpasswd.txt file.
3. Run the following query directly on database to grant select privileges on v\$session system table:


```
GRANT SELECT ON V_$SESSION TO "<restricted_user_name>"
```
4. To authenticate using password, run the dm_encrypt_password utility to set the password in the newly created dbpasswd.txt file using the following command:

```
dm_encrypt_password
-docbase <docbase_name>
-rdbms -encrypt <restricted_user_password>
-keyname <AEK_key_name>
```

For example:

```
dm_encrypt_password
-docbase testenv
```

```
-rdbms -encrypt password_123
-keyname CSaek
```

5. To authenticate without using password, update `sqlnet.ora` with the following attributes:

```
SQLNET.AUTHENTICATION_SERVICES=(kerberos5pre)
SQLNET.KERBEROS5_REALMS = /etc krb5 realms
SQLNET.KERBEROS5_CONF=/krb5/krb5.conf
SQLNET.KERBEROS5_KEYTAB=/krb5/krb41160
SQLNET.AUTHENTICATION_KERBEROS5_SERVICE=oracle
SQLNET.KERBEROS5_CONF_MIT=true
SQLNET.KERBEROS5_CC_NAME=/krb5/krbcache
```

6. Restart Documentum CM Server.

Documentum CM Server uses the new restricted user to make connections to the database.

After the session is established with the Oracle database, Documentum CM Server alters the current schema of the user with the schema of the schema owner. This is for seamless access to all OpenText Documentum CM tables.

7. Verify if the setting up of restricted user is successful using the following ways:

- Search for the `<database_user>` attribute in `server.ini`. In a normal operation, the attribute is not available while in a restricted user operation, the attribute is available.
- Enable the SQL trace on the Documentum CM Server. In a normal operation, the SQL trace is LOGON while in a restricted user operation, the SQL trace is OCI_SESSION.

For example:

```
apply,c,NULL,SET_OPTIONS,OPTION,S,sqltrace,VALUE,B,T
```

- Search for the database session ID. In a normal operation, the session ID is viewable while in a restricted operation, it is non-viewable.
- Search for the user name and schema for the corresponding db_session_id. Run the following query in the database:

```
SELECT SID, USERNAME, SCHEMANAME FROM V_$SESSION WHERE SID=<obtained_sid>;
```

In a normal operation, the user name is the name of the schema owner while in a restricted user operation, the user name is the name of the restricted user. In both the operations, the schema is the schema owner.



Notes

- You must run the dmdbtest utility whenever you switch from administration mode to operation mode.
- The functional operation of the normal database user (restricted_user) is restricted with only few privileges assigned. If more privileges needs to be granted, then grant those privileges directly in the database.
- If you have multiple repositories within one Documentum CM Server instance and want to configure for authentication without using password,

use the same restricted database user across all repositories. This is because, an Oracle client supports only one AD user. This is a Oracle limitation.

- OpenText recommends to rollback to normal mode of Documentum CM Server during any administrative operations such as installation, upgrade, alteration of schema, and so on.

2.9 Configuring Always On availability group of Microsoft SQL Server

1. Create a Microsoft Windows failover cluster. The *Microsoft* documentation contains the information.
2. Create a new `Role` under the failover cluster for the SQL Server availability group.
3. Add an SQL Server availability group Listener IP address for the new SQL Server availability group.
4. Connect to the SQL Server using the IP address and create a database for the repository. Take the full backup of the database and then add the new database to the availability group.
5. Create a system data source network using the same Listener IP address in ODBC data source administrator and verify the connection.

After the connection is successful, select the **use existing database** option, connect to the SQL Server, and select the new database you created in step 4. Then, create the repository.

2.10 Java Method Server for High-Availability

2.10.1 Overview

Java Method Server is a customized version of an application server for executing Documentum CM Server Java methods. OpenText Documentum CM provides a servlet called `DO_METHOD` to execute Documentum CM Server methods. The method server itself is a Java-based web application. It communicates with the Documentum CM Server via HTTP calls. Each time a method is invoked, the Documentum CM Server makes an HTTP request passing the name of the Java class which implements the method along with any specified arguments to a servlet which knows how to execute the specified method. Java Method Server supports high-availability (HA). Java Method Server supports two HA types: failover and load balancing.

- Failover

In a failover setup, if one of the Java Method Server fails, the other Java Method Server in the failover setup continues the method running activity from the beginning from the same Documentum CM Server.



Note: Java Method Server that is set for failover is considered for a new method execution also only when the Java Method Server that is set for load balancing is down.

- Load balancing

If you configure Java Method Server load balancing, method executes in a round robin manner per Documentum CM Server instance.

2.10.2 Installation of additional Java Method Server

You can use Independent Java Method Server to install additional Java Method Server on the same host or different hosts from Documentum CM Server. “[Independent Java Method Server](#)” on page 167 provides the details.

2.10.3 Enabling Java Method Server HA feature

From the 7.3 release, you have changes to the Java Method Server HA feature.

To enable the changed Java Method Server HA feature, set `r_module_name` to `JMS_HA_SETUP_ENABLED` and `r_module_mode` to 1. To use the unchanged Java Method Server HA feature, set `r_module_mode` to 0.

For example:

```
append,c,docbaseconfig,r_module_name
JMS_HA_SETUP_ENABLED
append,c,docbaesconfig,r_module_mode
1
save,c,docbaseconfig
```

Then, restart the repository to enable the Java Method Server HA feature.



Note: The 7.2 version of the Installation Guide contains more information about the unchanged Java Method Server HA features.

2.10.4 Java Method Server HA configurations

After you enable the Java Method Server HA feature, you should configure the following attributes in the `dm_server_config` object per each Documentum CM Server instance. You can also configure using Documentum Administrator.

Server configuration object of each Documentum CM Server that maintains the `dm_jms_config` objects list. It represents `jms_mode` for each `dm_jms_config` object. For example:

- `jms_config_id`:

```
jms_config_id [0]: <jms_config_id>
jms_config_id [1]: <jms_config_id1>
jms_config_id [2]: <jms_config_id2>
```

- `jms_type`:

```
jms_type: 1
```

where 1 means that it supports round robin load balancing only and reserved for future release.

- jms_mode:

```
jms_mode [0]: 1/2/3  
jms_mode [1]: 1/2/3  
jms_mode [2]: 1/2/3
```

where jms_mode indicates if it has been configured for failover and/or load balancing.

- Load balancing: 1 (0001)
 - Failover: 2 (0010)
 - Load balancing and Failover: 3 (0011)
- jms_proximity: Reserved for future release.

For default configuration, you must use the following command per Documentum CM Server instance:

```
apply,c,NULL,REFRESH_JMS_CONFIG_LIST,DEFAULT_JMS_VALUES,B,T
```

where default configuration means Documentum CM Server fetches all dm_jms_config objects from the repository, adds dm_jms_config objects as jms_config_id configuration object, and sets the value of the corresponding jms_mode to 3 (for both failover and load balancing) for each jms_config_id.



Notes

- dm_server_config should have at least one dm_jms_config object that has either 1 or 3.
- If you want to add additional Java Method Server, then you must configure jms_config_id and jms_mode values manually in the server configuration object for newly created dm_jms_config object. Otherwise, it resets to default values for all Java Method Servers when you run the REFRESH_JMS_CONFIG_LIST apply command.

2.10.5 Method location options for Java Method Server HA

- LOCAL: Methods with JMS_LOCATION=LOCAL means, it must be run on the same host Java Method Server as the originating Documentum CM Server.
- ANY: Methods with JMS_LOCATION=ANY means, it can be run on any Java Method Server as long as it is on same LAN as the originating Documentum CM Server.



Note: If you do not configure any JMS_LOCATION attribute, Documentum CM Server uses the JMS_LOCATION=ANY option.

- REMOTE: Methods with JMS_LOCATION=REMOTE means, it runs on a remote Java Method Server. Remote Java Method Server is associated to a remote Content Server. If a Documentum CM Server needs to run a method with setting as JMS_

LOCATION=REMOTE, it must send HTTP POST request to its associated remote Java Method Servers.

- JMS_ID: Method JMS_LOCATION=<jms_config_id> means, it runs on a particular Java Method Server.

2.10.6 Pre-installation requirements and tasks

- For methods requiring trusted authentication, you must set a_extended_properties to JMS_LOCATION=LOCAL. Otherwise, setting these methods results in authentication issues when executed in cross communication between Documentum CM Server and Java Method Server.
- After enabling the Java Method Server HA feature in the docbase_config object, you must manually run the following command for each Documentum CM Server:

```
apply,c,NULL,REFRESH_JMS_CONFIG_LIST,DEFAULT_JMS_VALUES,B,T
```

Restart the repository. Check for dm_jms_config entries in dm_server_config. The values of jms_config_id and jms_mode attributes is added.

- In Java Method Server HA, the a_special_app attribute does not honor the Workflow job keyword. The value of this attribute must be either workflow or blank.
- In dm_jms_config, the value of servlet_name[0] must be do_method and the value of base_url[0] must be the respective URL.
- In the r_module_name of dm_docbase_config, in a Linux setup, note the following information:
 - If you do not add *JMS_HA_AUTO_REFRESH_DISABLED*, then the Java Method Server HA auto refresh operation happens. The behavior is similar to pre-21.2 versions.
 - If you set the value of *JMS_HA_AUTO_REFRESH_DISABLED* to 1, then the Java Method Server HA auto refresh does not happen.
 - If you set the value of *JMS_HA_AUTO_REFRESH_DISABLED* to 0, then the Java Method Server HA auto refresh operation happens where Change checker launches the Java Method Server HA Health checker in a separate process.
- If the system and custom methods need Java Method Server failover, make sure to expose the failover to the method. Set is_restartable to T. By default, it is F.

2.10.7 Java Method Server status

You can run the following commands to:

Refresh the Java Method Server:

```
apply,c,NULL,REFRESH_JMS_CONFIG_LIST
```

Check the status of Java Method Server:

```
apply,c,NULL,DUMP_JMS_CONFIG_LIST
```

2.10.8 Enabling and understanding logs

- Enable the debug tracing on %DM_JMS_HOME%\webapps\dmMethods\WEB-INF\web.xml of Java Method Server, set the value of trace to t.
The trace details are logged in catalina<timestamp>.log.
- Enable trace_launch on the Documentum CM Server. Run the following commands from IAPI:

For HTTP POSTs:

```
apply,c,NULL,SET_OPTIONS,OPTION,S,  
trace_http_post,VALUE,B,T
```

For complete launch:

```
apply,c,NULL,SET_OPTIONS,OPTION,S,  
trace_complete_launch,VALUE,B,T
```

The trace details are logged in \$DM_HOME\log\catalina<timestamp>.log.

- Enable trace_launch at the method level while running the method enable trace launch option.

The trace details are logged in a separate log file for each method under \$DOCUMENTUM\dba\log\xxxxxx\sysadmin\<Method_Name>.txt

- Installer log files

2.10.9 Error messages

Error message	Description
DM_METHOD_E_HTTP_COMMUNICATION	Application server to which the Documentum CM Server wants to send the POST request to, is unavailable or down.
DM_METHOD_E_HTTP_POST_APP_SERVER_NAME_NOTFOUND	No matching app_server_name or servlet_name value found in any of the dm_server_config or dm_jms_config objects.
DM_METHOD_E_NO_JMS_AVAILABLE2	Documentum CM Server understands that all the Java Method Servers are currently down and do not accept any POST requests.

Error message	Description
DM_METHOD_E_HTTP_POST_FAILED	Documentum CM Server could not send the POST request to the application server.

2.10.10 Supported load balancing and HA configurations

Java Method Server supports the following HA configurations:

- Documentum CM Server and Java Method Servers on a single host
- Documentum CM Server and Java Method Server on two hosts
- Documentum CM Server and Java Method Servers on multiple hosts

Documentum CM Server HA deployment involves two or more Documentum CM Servers. Java Method Server High-Availability involves adding Java Method Servers to additional Documentum CM Servers such that each Documentum CM Server has a dedicated Java Method Server. Java Method Server High-Availability is automatically enabled by associating each Documentum CM Server with their dedicated Java Method Server.

2.10.10.1 Documentum CM Server and two or more Java Method Servers on a single host

The illustration depicts two Documentum CM Servers and their embedded instances of Java Method Server set up on a single machine, serving one repository.

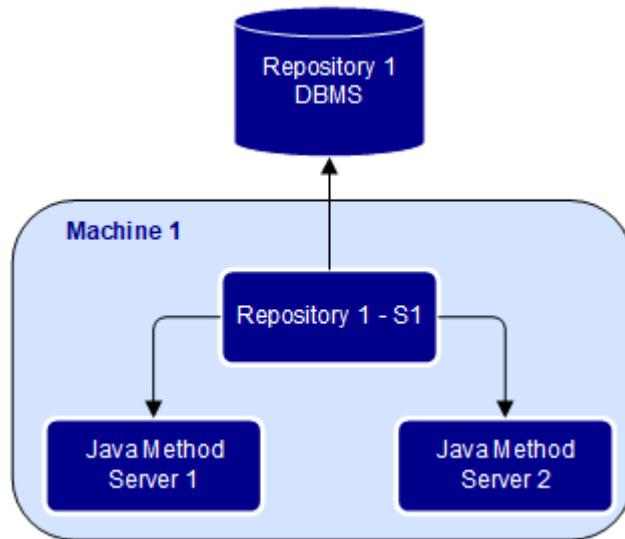


Figure 2-2: Documentum CM Server and two or more Java Method Servers on a single host

When installing the first Documentum CM Server, Java Method Server is installed by default. When adding a Documentum CM Server on the same host, the Java

Method Server for the second Documentum CM Server is not installed. Instead, it shares the Java Method Server of the first Documentum CM Server.

To install and configure the main Documentum CM Server:

1. Install the main Documentum CM Server.
2. After the installation completes, run the Documentum CM Server configuration program. The Documentum CM Server Configuration program guides you through creating a repository.

The installation of the main Documentum CM Server also installs an instance of Java Method Server. A single dm_jms_config object for that instance of Java Method Server is created in the repository.

3. Add the additional Java Method Servers.

2.10.11 Java Method Server for HA on multiple hosts

The illustration depicts two embedded Java Method Server instances, each connected to its own Documentum CM Server on two hosts supporting one repository to two Java Method Server instances set up for HA with Documentum CM Servers on two hosts supporting one repository.

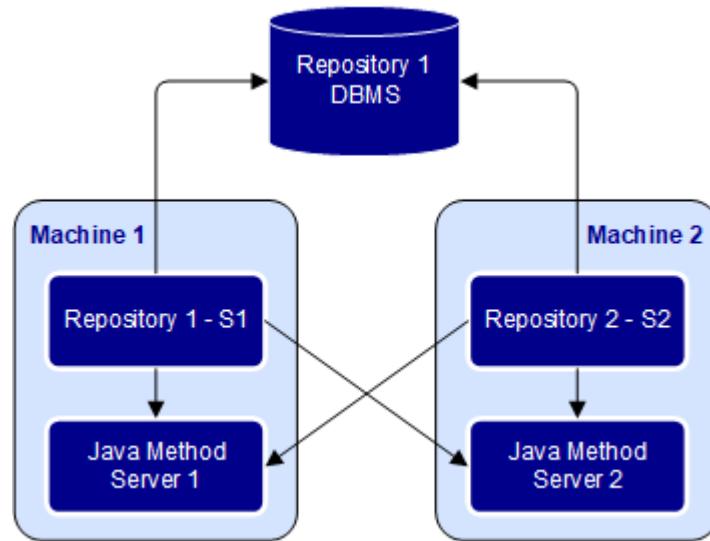


Figure 2-3: Java Method Server for HA on multiple hosts

Each connected to its own Documentum CM Server on multiple hosts supporting one repository to multiple Java Method Server instances set up for HA with Documentum CM Servers on multiple hosts supporting one repository.

To install and configure the main Documentum CM Server:

1. Install the main Documentum CM Server.
2. After the installation completes, run the Documentum CM Server configuration program. The Documentum CM Server Configuration program guides you through creating a repository.

The installation of the main Documentum CM Server also installs an instance of Java Method Server.

To add Documentum CM Servers:

1. On multiple hosts, install the main Documentum CM Server.
2. On Windows, run `<$DM_HOME>\install\cfsConfigurationProgram.exe` as the installation owner; on Linux, run `<$DM_HOME>/install/cfsConfigurationProgram.bin` as the installation owner.
3. Provide the appropriate information when prompted to do so by the program.

To configure Java Method Server for HA:

1. Enable the `<JMS_HA_SETUP_ENABLED>` variable.
2. Add the Java Method Server configuration IDs or set the `jms_mode` attribute for each Documentum CM Server's server configuration object.

Or

Run the following command for each Documentum CM Server instance to configure the defaults values:

`apply,c,NULL,REFRESH_JMS_CONFIG_LIST,DEFAULT_JMS_VALUES,B,T`
3. Set `a_extended_properties` to `JMS_LOCATION=LOCAL` for all methods requiring trusted authentication.
4. In Java Method Server HA, the `a_special_app` attribute does not honor the `Workflow job` keyword. The value of this attribute must be either `workflow` (if BPM is installed) or blank.
5. Make sure that in `dm.jms_config`, the value of `servlet_name[0]` is `do_method` and the value of `base_url[0]` is the respective URL.
6. Restart the repository.

2.10.12 Two or more Documentum CM Servers and two or more Java Method Servers on a multiples host

You can install and configure mixed mode Documentum CM Server and Java Method Server combinations.

[“Documentum CM Server and two or more Java Method Servers on a single host” on page 121](#) and [“Java Method Server for HA on multiple hosts” on page 122](#) contains the instructions.

2.11 Installing Documentum CM Server with Microsoft Cluster Services

This section describes how to install and configure Documentum CM Server to provide failover support under Microsoft Cluster Services.

2.11.1 Microsoft Cluster Services overview

Microsoft Cluster Services supports the Active/passive clusters and Active/active forms of clustering.

In a cluster environment, every service that the cluster runs uses resources of the cluster node. Every service has its own resources, such as hard drive, IP address, and network name, assigned to it. All resources that a clustered service uses form a resource group. The connection broker and Documentum CM Server are a part of this resource group. In a cluster, all resources form a virtual server that can move from one physical server to another to provide failover support.

2.11.2 Choosing a configuration

A Documentum CM Server installation supports two types of cluster service configurations:

- active/passive
- active/active

This section provides detailed installation instructions for both configurations. Select the configuration based on available hardware and your organization's business needs. The figure illustrates Documentum CM Server and connection broker setup in an active/passive cluster.

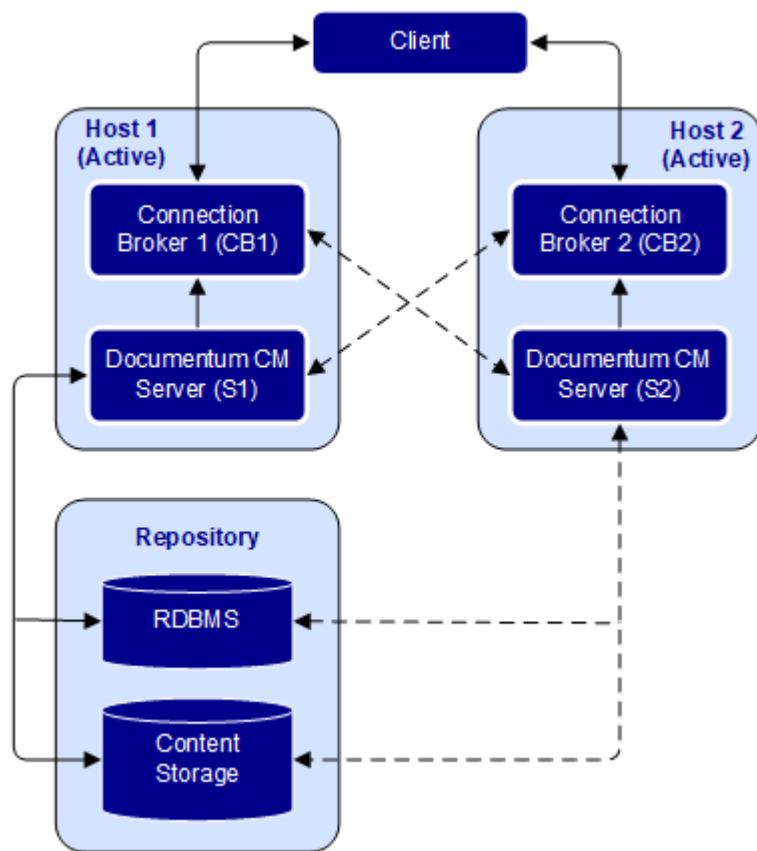


Figure 2-4: Active/passive cluster

In an active/passive configuration, both nodes support the same repository. Documentum CM Server and connection broker run on the primary node. If the primary node fails, the secondary node that was on standby takes ownership of the resource group. After the cluster resources are brought online on the secondary node, the connection broker and Documentum CM Server start on the secondary mode.

In an active/active configuration, each node supports a different repository. Each node is considered the standby to the other node. Each node owns its own resource group. Each resource group has its own virtual IP address, a virtual host name, shared disk, connection broker, and Documentum CM Server. The figure illustrates Documentum CM Server and connection broker setup in an active/active cluster.

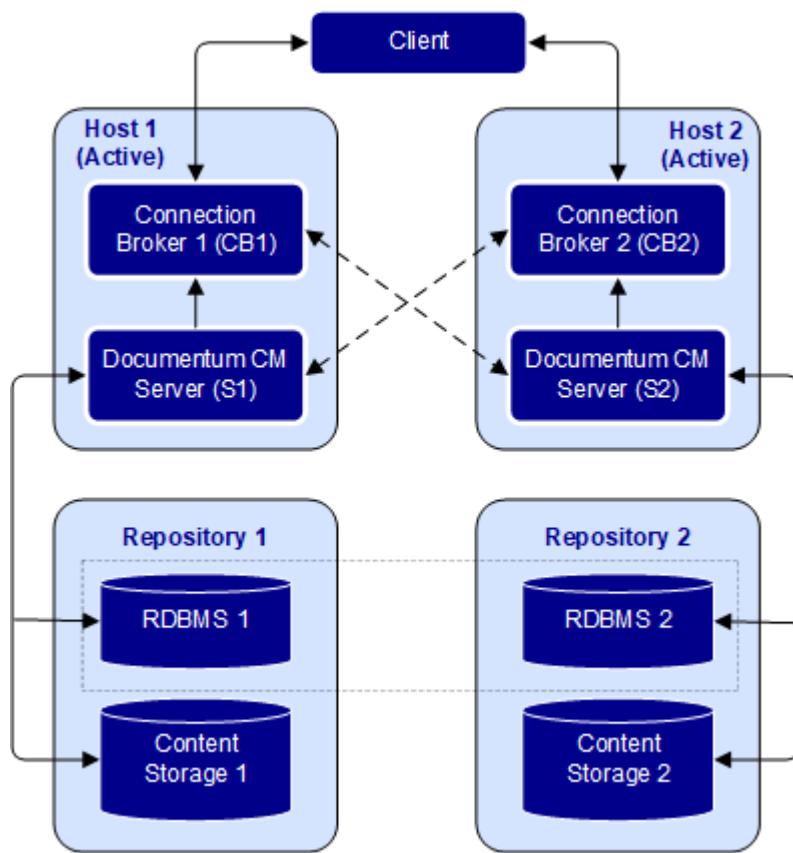


Figure 2-5: Active/active cluster

If a node fails or is taken offline, its resource group is moved to the remaining node in the cluster. The remaining node then manages two resource groups. When the failed node is running again, the cluster administrator can move one resource group back.

If you switch the content transfer from one node to the other during a content transfer operation, it fails. This occurs because Accelerated Content Services may not be available on the second node to perform the content transfer. If a subsequent content transfer operation is done, the content transfer takes place through the Application Server (for example, Webtop) and is expected to succeed. If there are issues in transferring large files, configuration changes may be required.

2.11.3 Pre-installation requirements

Before you install and configure Documentum CM Server and a repository under Microsoft Cluster Services, perform the pre-installation tasks. “[Prerequisites](#)” on page 18 contains the information.

Whether you are configuring an active/passive cluster or an active/active cluster, set up the shared disks to be used by the repositories. Make sure that the shared disks include a directory to use for content storage.

If you are configuring an active/active cluster, the user who configures Microsoft Cluster Services must have read and write permissions on both nodes on the directories where the connection broker logs reside.

2.11.4 Configuring an active/passive cluster

Follow these procedures to configure an active/passive cluster and install servers and a Documentum CM Server.



Note: When setting up an active/passive cluster in a Windows cluster environment, configuration of a repository on the second cluster node can fail if you are using different aek.key files and dbpassword.txt files for each node. In active/passive cluster configuration, both the primary and secondary node must have the same aek.key file (located in \Documentum\dba\secure) and dbpasswd.txt (located in \Documentum\dba\config\docbase_name). When creating the environment manually copy the two files from the primary node to the secondary node.

2.11.4.1 Creating the cluster resource group

Use this procedure to set up the cluster resource group.

To create the cluster resource group:

1. Use the MSCS Cluster Administrator utility to create a cluster resource group that contains the following resources:

- A virtual IP address

Documentum CM Server uses this IP address. Other products can share the IP address because the connection broker and Documentum CM Server only listen on particular ports.

- A virtual network host name

The virtual network host name is for the virtual IP address.

- A shared disk partition

This is the location of the repository data directory, where content files are located.

2. Move the resource group between the primary and standby hosts to make sure that the virtual IP address, virtual network host name, and shared disk partition fail over properly.
 - a. In the Cluster Administrator, right-click the cluster group name.
 - b. Click **Move Group**.
The resource group is toggled between the hosts and the owner name changes.
3. Make sure that the first node owns the resource cluster group.

2.11.4.2 Installing Documentum CM Server software on the nodes

The first part of the installation process copies files from the installation media to the hard disk. On each of the two nodes, copy the files from the installation media to create a Documentum CM Server installation on each node. Use the same drive letter on each node for the installation.

2.11.4.3 Configuring Documentum CM Server

The second part of the installation involves configuring Documentum CM Server and its associated connection broker, which includes configuring them for Microsoft Cluster Services.

To configure Documentum CM Server on Node1:

1. Move the cluster control to Node1.
2. Extract the installation files, launch `serverSetup.exe` and install Documentum CM Server on Node1.
3. Configure the connection broker and repository on Node1.



Note: Specify the location of the Data folder which will be the location of your file stores. It is recommended that this folder be on a shared drive on the Microsoft Cluster setup or on a shared drive accessible to both the nodes.

To configure Documentum CM Server on Node2:

1. Turn off the connection broker and repository services on Node1 and move the cluster control to Node2.
2. On Node2, follow the steps used on Node1 using the same **Repository name** and **Repository ID** as in Node1.
3. In the **Select Database and Server Account** page, select **Use existing SQLServer user account and database** to use the database created earlier.
4. Installer prompts you to enter the details of the existing database. Type the details.

5. Type the information of the SMTP server and the repository owner's email address. Type the same details which were provided in Node1.
6. Select the same optional modules selected on Node1.
7. Installer proceeds with the connection broker configuration and an error message is shown due to failure to connect to the repository. At this stage stop the installation.
8. Now turn off all the connection broker and repository services on Node2. Copy the <\$DOCUMENTUM>\dba\secure\aekey.key and <\$DOCUMENTUM>\dba\config\<repository_name>\dbpasswd.txt files from Node1 to corresponding locations on Node2.
9. To configure more nodes, move the cluster control to that node and follow steps on Node2.

The setup of connection broker and repository configuration on two nodes required for the purpose of using Microsoft Cluster Services for failover is complete.

2.11.4.4 Configuring the connection brokers

You need to configure the connection brokers on both nodes to listen on the virtual network host.

To configure connection brokers:

1. Click **Documentum > Documentum Server Manager** to start Server Manager.
2. Click **Docbroker**, then click **Edit Service**.
3. Add the following line to the service command:

```
-host <virtual_network_host_name>
```

For example, if the virtual network host name is dmcluster1:

```
-host dmcluster1
```

Click **OK**.

4. Highlight **repository** and click **Edit server.ini**.
5. Edit the DOCBROKER_PROJECTION_TARGET section of the server.ini file:

```
[DOCBROKER_PROJECTION_TARGET]
host=<virtual_network_host_name>
```

6. Save the server.ini file.
7. Navigate to C:\Documentum\config and open the dfc.properties file in a text editor.
8. Edit the primary connection broker entry of the dfc.properties file:

For example:

```
dfc.docbroker.host[0] = dmcluster1
```

9. Save the `dfc.properties` file.
10. Repeat this procedure on the other node.

2.11.4.5 Creating additional cluster resources on Microsoft Cluster Services

Use this procedure to create cluster resources for the connection broker and Documentum CM Server. Perform the procedure only on the node that presently owns the existing resource group.

To configure the connection broker and repository for failover:

1. Navigate to **Administrative Tools > Failover Cluster Management**.
2. Right-click the services node and select the service you created.
3. Select **Generic Service** to configure for high availability and click **Next**.
4. Select the **Documentum Docbroker Service** connection broker service from the list, proceed with the wizard and complete configuring the connection broker and repository for failover.
5. Right-click the new service, and select **Properties**.
6. In the General tab, select **NODE1** and **NODE2** in the Preferred owners list. Click **OK**.
7. Right-click the **Documentum Docbroker Service** connection broker service, and select **Properties**.
8. In the **General** tab, make sure that the **Use Network Name for computer name** option is selected by default, and click **OK**.
9. Right-click the **Documentum Docbroker Service** connection broker service in the console, and select the **Bring this resource online** option. The resource is brought online and the Online icon is displayed in the Status column of the Summary page.
10. Right-click the configured service, select **Add a resource > Generic Service**.
11. Select **Documentum Docbase Service** in the **Select Service** list of the **Select Service** page.
12. Click **Next** and click **Next** again.
13. Click **Finish**. The **Documentum Docbase Service** is added to the Other Resources list.
14. Right-click the **Documentum Docbase Service** in the summary page, and select **Properties**.
15. In the **Dependencies** tab, add the **Documentum Docbroker Service**.

16. In the **General** tab, make sure that the Use Network Name for computer name option is selected, and click **OK**.
17. Right-click the **Documentum Docbase Service** in the Summary page console, and select the **Bring this resource online** option. The resource is brought online and the Online icon is displayed in the Status column of the Summary page.
18. Right-click the configured service, select **Add a resource > Generic Service**.
19. Select **Documentum Java Method Server** in the **Select Service** list of the **Select Service** page.
20. Click **Next** and click **Next** again.
21. Click **Finish**. The **Documentum Java Method Server** is added to the Other Resources list.
22. Right-click the **Documentum Java Method Server** in the Summary page, and select **Properties**.
23. In the **Dependencies** tab, add the **Documentum Docbroker Service** and **Documentum Docbase Service**.
24. In the **General** tab, make sure that the Use Network Name for computer name option is selected, and click **OK**.
25. Right-click the **Documentum Java Method Server** in the Summary page console, and select the **Bring this resource online** option. The resource is brought online and the Online icon is displayed in the Status column of the Summary page.

2.11.4.6 Verifying failover

After you complete the preceding procedures, verify that failover works properly.



Note: Make sure that you run the `dm_crypto_boot` utility on all the Documentum CM Server nodes only if it has the custom passphrase before switching Documentum CM Server services to any node.

1. On a client computer, make sure that the `dfc.properties` entries refer to the virtual network host name or virtual IP address.
2. Connect to the repository from the client computer.
3. Start the Cluster Administrator utility.
4. Move the resource group from the node where it is running to the other node.
5. After the resource group comes online on the other node, verify that the client can run queries.

2.11.5 Configuring an active/active cluster

In an active/active cluster, each node initially has its own repository and Documentum CM Server. You create two cluster resource groups, and each node owns one. If a Documentum CM Server fails on one node, a second Documentum CM Server starts on the second node to keep the repository on the first node running.

Each cluster resource group has the following:

- virtual IP address
- virtual network host name
- one shared disk drive (where the repository resides)
- one connection broker
- one Documentum CM Server

2.11.5.1 Creating the first cluster resource group

Create the first cluster resource group. “[Creating the cluster resource group](#)” on page 127 contains the information.

2.11.5.2 Installing Documentum CM Server software on the hosts

On each of the two nodes, copy the files from the installation media to create a Documentum CM Server installation on each node.



Note: Use the same drive letter on each node.

2.11.5.3 Configuring Documentum CM Server on the first and second nodes

In an active/active configuration, configure a Documentum CM Server on each of the two nodes. The procedure for configuring Documentum CM Server in an active/active cluster is the same as configuring Documentum CM Server in an active/pассиве cluster. Follow the instructions on the first and the second node in the cluster.

2.11.5.4 Configuring the second cluster resource group

Create the second cluster resource group.

2.11.5.5 Modifying server.ini and dfc.properties

You might want to edit `server.ini` and `dfc.properties` on both nodes to make sure that each repository projects to the connection brokers on the two nodes.

In the following examples, assume that the virtual network hosts for the two cluster resource groups are called `dmcluster1` and `dmcluster2`. It does not matter which cluster resource group is primary and which is backup.

Edit all four of the `server.ini` files so that they read as follows:

```
[DOCBROKER_PROJECTION_TARGET]
host=dmcluster1
[DOCBROKER_PROJECTION_TARGET_1]
host=dmcluster2
```

Edit the two `dfc.properties` files so that they read as follows:

```
dfc.docbroker.host[0]=dmcluster1
dfc.docbroker.host[1]=dmcluster2
```

2.11.5.6 Configuring the connection broker and repository for failover

The procedure for configuring the connection broker and repository for failover in an active/active cluster is the same as configuring them in an active/passive cluster. But the configuration for failover needs to be done separately for each of the two repository services binding them with their respective dependant resources.

2.11.5.7 Verifying failover

After you complete the preceding procedures, verify that failover works properly.

To verify failover:

1. On a client computer, make sure that the `dfc.properties` entries refer to both virtual network host name or virtual IP address.
2. Connect to both repositories from the client computer.
3. Start the Cluster Administrator utility.
4. Move the two resource groups back and forth between the nodes.
5. After a resource group comes online on a new node, verify that the client can run queries.

2.12 Uninstalling Documentum CM Server

This section explains how to delete a repository or connection broker and how to uninstall an existing Documentum CM Server. Do not uninstall an existing installation to upgrade to a new Documentum CM Server release, because all upgrades are based on an existing installation. Use the procedures in this section only if you want to uninstall an existing Documentum CM Server, a repository and its contents, a connection broker, or a Documentum CM Server software installation.

To delete a repository or connection broker or uninstall Documentum CM Server, you need to meet the following requirements:

- Be able to log in as the installation owner
- Have sufficient database privileges to drop tablespaces or databases

2.12.1 Uninstalling components

Uninstall the software components in this order:

1. Stop the Java Method Server service.
2. Delete the configuration and uninstall OpenText Documentum Content Management (CM) Thumbnail Server, if available.
3. Delete the repository.
4. Delete any connection broker located on the current host.
5. Uninstall the Documentum CM Server software.
6. Uninstall the Index Agent configuration program if full-text indexing is installed.

2.12.2 Deleting a repository

To delete a repository you need to meet the following requirements:

- Be able to log in as the installation owner
- Have sufficient RDBMS privileges to drop tablespaces or databases.



Note: If the repository has a OpenText™ Documentum™ Content Management Transformation Services product installed on it, you need to uninstall the OpenText Documentum Content Management (CM) Transformation Services product before deleting the repository. If you do not, the Transformation Services product will not be available in all other repositories.

Log in to the host as the Documentum CM Server installation owner, start the Documentum CM Server configuration program, and select **Delete an existing repository** to delete a repository.

2.12.3 Deleting a connection broker

Log in to the host as the installation owner, start the Documentum CM Server configuration program, and select **Delete an a connection broker** to delete a connection broker.

2.12.4 Uninstalling Documentum CM Server using GUI

1. Delete all configuration of Thumbnail Server and then uninstall the Thumbnail Server software from the host, if available.
2. Delete all repositories and connection brokers from the host where Documentum CM Server is installed.
3. Run the Documentum CM Server uninstallation program.
 - On Windows, run <\$DOCUMENTUM>/uninstall/server/uninstall.exe
 - On Linux, run <\$DOCUMENTUM>/uninstall/server/uninstall Optionally, run <\$DOCUMENTUM>/uninstall/dfc/uninstall to uninstall the **Documentum Foundation Classes Runtime Environment**.



Note: Do not uninstall the **Documentum Foundation Classes Runtime Environment** if any other OpenText Documentum CM software is installed on the host.

2.12.5 Uninstalling Documentum CM Server using command line

1. Update the `linux_install.properties` or `win_install.properties` file with the following content only:

```
INSTALLER_UI=silent
```

2. Run the following command from the command line:

For Windows:

```
%Documentum%\uninstall\server>Uninstall.exe -f <Path of the Silent installer property file name>/<Silent installer property file name>
```

For Linux:

```
$DOCUMENTUM/uninstall/server>./Uninstall -f <Path of the Silent installer property file name>/<Silent installer property file name>
```

2.13 Troubleshooting

This section contains the information for troubleshooting the common Documentum CM Server installation problems.

2.13.1 Identifying the problem and resolution

When experiencing a problem, perform the following preliminary actions:

- Make sure that you are connected as the installation owner.
- On Linux, make sure that the environment variables are set correctly in the installation owner's environment.
- Review the Documentum CM Server installation logs.

Symptom	Cause	Fix
Documentum CM Server installation or upgrade fails.	You are trying to install or upgrade the Documentum CM Server but you are not connected as the installation owner.	Connect using the installation owner account.
While installing a repository, you see an error message that indicates that the user is not a valid Linux user: Configuration of the docbase fails with the message 'user must be a valid UNIX user' exec(): 0509-036 Cannot load program /u01/app/documentum/product/5.2/bin/dmisvaliduser because of the following errors: 0509-150 Dependent module libldap50.so could not be loaded. 0509-022 Cannot load module libldap50.so. 0509-026 System error: A file or directory in the path name does not exist.	Three possible causes: 1. The installation owner account does not have the installation owner group designated as the user's primary group. Group ownership of the OpenText Documentum CM binaries is incorrect. 2. The shared library path environment variable is not set correctly.	To fix: 1. Make the installation owner group the installation owner's primary group. 2. Set the shared library path environment variable correctly. <i>"Setting the required environment variables" on page 32</i> contains the information.
Documentum CM Server upgrade appears unresponsive.	There might be a cyclic group.	

Symptom	Cause	Fix
<p>On Windows hosts, you see the following error during installation:</p> <pre>Could not initialize interface awt exception ExceptionInitialization Error</pre>	The correct video driver for the video card is not installed on the host.	Review the hardware and software configuration of the host.
<p>You see the following error during an upgrade of an older repository:</p> <pre>Failed to retrieve serverconfig object with name <serverconfigname>. ***Failed to encrypt passwords for docbase ec_epac, status -1057226550 **Operation failed ** [DM_CRYPTO_E_NO_LOCAL_COMPONENT_STORE] error: "No local component store for server" Please read error log C:\WINNT\Temp \dm_chec_bin.ServerConfigurator.log for more information.</pre>	The dm_ContentReplication method has some parameter arguments left over from EDMS98.	Delete the following entry from the dm_ContentReplication method: <pre><serverconfigname> [domain \]user,passwd</pre>

Symptom	Cause	Fix
You see the following errors during upgrade from 5.3 SP6 on Oracle 10: Tue Feb 22 21:48:08 2005 098000 [DM_SESSION_I_INIT_BEGIN]]info: "Initialize dmContent." Tue Feb 22 21:48:08 2005 567000 [DM_SESSION_I_INIT_BEGIN]]info: "Initialize dmiSubContent." Tue Feb 22 21:48:08 2005 598000 [DM_TYPE_MGR_E_CANT_FIND_TABLE]error: "Failure to find table dmi_subcontent_sv as part of fetch of type dmi_subcontent: error from database system is ORA-24372: invalid object for describe" Tue Feb 22 21:48:08 2005 598000 [DM_SESSION_E_INIT_FAILURE1]error: "Failure to complete dmiSubContent initialization." You might also see this message: ORA-24372: invalid object for describe	Invalid Oracle views belonging to types _sv, _sp, _rv, and _rp. A view in Oracle becomes invalid when the base tables or references change (for example, by adding/dropping a column, or dropping a unique constraint index).	Make the views valid before upgrading Documentum CM Server. To determine which views in the Oracle installation are invalid, you can run the following query from SQLPLUS logging in as the repository owner: <pre>select object_name, object_type from user_objects where status='INVALID';</pre> To recompile the views: <pre>ALTER VIEW <view_name> COMPILE;</pre> The Oracle-supplied package named DBMS_UTLITY has a procedure named COMPILE_SCHEMA. This procedure will compile all stored code, views, and so on, for the schema provided. The best way to compile all database objects that are invalid is to use a script in the \$ORACLE_HOME/rdbms/admin directory named utlrp.sql. This script finds all objects in the data dictionary that are invalid and compiles them. This script is typically mentioned in patch notes but you can use it any time a schema change occurs.
Unable to start the repository.	Changing the default passphrase to a custom passphrase.	See " Changing the default passphrase " on page 106.

2.13.2 Installer hangs when the number of mount points exceeds 4000 while installing on Linux

On a Linux machine, installation can hang when there are too many mount points open in the box. Check the output in bytes by entering the following command:

```
mount | wc -c
```

Installer hangs when the number returned by the previous command exceeds 4000.

Workaround: Copy and paste the following into your Telnet session and run the installation as usual:

```
cd /tmp
mkdir bin.$$
cd bin.$$
cat > mount <<EOF
#!/bin/sh
exec /bin/true
EOF
chmod 755 mount
export PATH=`pwd`:$PATH
```

After the installation completes, delete /tmp/bin.\$\$ by running the following command (in the same Telnet session):

```
rm -r /tmp/bin.$$
```

2.13.3 Recovering from a failed repository configuration or upgrade

If repository configuration fails, whether you are upgrading an existing repository or creating a new one, you can recover from the failure.

Typical reasons for a failure include problems with the database connection or errors in Documentum CM Server creation. Before you proceed with the following instructions, read the Documentum CM Server installation logs and correct any problems:

To recover from a failed installation or upgrade:

1. Correct any problems noted in the Documentum CM Server installation logs.
2. Restart the Documentum CM Server configuration program.
3. Select **Repository > Upgrade an existing repository**.
4. Select the repository where the failure occurred.
5. Check **Upgrade**.

This takes you through the configuration steps again and reruns the scripts that create the repository.

2.13.4 Recovering from a stalled Documentum CM Server upgrade

A Documentum CM Server upgrade that stalls in the middle or takes hours to complete can be caused by cyclic groups. A cyclic group is a subgroup of a member group, causing the Documentum CM Server to cycle during the upgrade. If the Documentum CM Server has encountered a cyclic group, the last line of the Documentum CM Server log is:

```
Thu Jun 28 14:00:14 2007 715540
[DM_SESSION_I_INIT_BEGIN]info:"Initialize dmGroup."
```

Use the following instructions to identify the cyclic group. After you locate the cyclic group, contact OpenText Global Technical Services for assistance in correcting the problem, which requires direct SQL Server statements in the database:

To identify and correct a cyclic group:

1. From the operating system, stop the Documentum CM Server startup.
 - On Windows, open Task Manager, select the correct Documentum CM Server process on the Processes tab, and click **End Process**.
 - On Linux, determine the correct Documentum CM Server process and use the kill command to end the process.
2. If you are on Linux, restart the Documentum CM Server using the `-osqltrace` option:


```
dm_start_<repositoryname> -osqltrace
```
3. If you are on Windows, edit the Documentum CM Server startup command, then restart the Documentum CM Server.
 - a. Click **Documentum Server Manager**.
 - b. Select the correct repository.
 - c. Click **Edit Service**.
 - d. In the **Command** field, add `<-osqltrace>` after the repository name. Click **OK**.
 - e. Restart the Documentum CM Server.
4. When the Documentum CM Server appears to be unresponsive, open the Documentum CM Server log and identify the query that is looping.

If there is a cyclic group, the last query in the log is recorded multiple times and takes this format:

```
Thu Jun 28 13:33:17 2007 435439: 21547[1]
SELECT SB_.R_OBJECT_ID FROM <repository_owner>.dm_group_s SB_
WHERE (SB_.R_OBJECT_ID=:objectp AND SB_.I_VSTAMP=:versionp)
Thu Jun 28 13:33:17 2007 435608: 21547[1] :objectp = 1200fb8080000909
Thu Jun 28 13:33:17 2007 435608: 21547[1] :versionp = 0
```

In the preceding example, the cyclic group has the r_object_id of 1200fb8080000909.

5. Run the following query:

```
SELECT group_name
FROM dm_group_s
WHERE r_object_id='<r_object_id_of_cyclic_group>'
```

This query returns the name of the group, which you need for determining which group is the cyclic group.

6. Run the following query:

```
SELECT groups_names
FROM dm_group_r
WHERE r_object_id = '<r_object_id_of_cyclic_group>'
```

The query returns the names of each group that is a member of the problem group.

7. For each of the group names returned, run this query:

```
SELECT r_object_id from dm_group_s
where group_name = '<member_group_name>'
```

The query returns the r_object_id for each member group.

8. Repeat steps 6 and 7 iteratively for each subgroup until you locate the cyclic group.
9. Contact OpenText Global Technical Services for assistance in correcting the problem.

2.13.5 Identifying issues for certificated-based SSL communication

This section lists some of the common issues that occur during configuration of certificated-based SSL communication, causes, and its resolutions.

2.13.5.1 Connection broker startup fails

1. Check if the connection broker keystore and keystore password files are available in the \$DOCUMENTUM/dba/secure directory.
2. Check if an entry for the connection broker keystore and keystore password files is available in the connection broker configuration file (docbroker.ini).

Make sure that you add the following entry in the docbroker.ini file:

```
[DOCBROKER_CONFIGURATION]
secure_connect_mode=secure
keystore_file=broker.p12
keystore_pwd_file=broker.pwd
cipherlist=AES128-SHA
crypto_keyname = <CSaek>
```

- crypto_keyname = <CSaek> where <CSaek> is the AEK key name that is used to encrypt the password in the broker.pwd file. Use the following syntax:

```
dm_encrypt_password -encrypt password -file broker.pwd -keyname <CSaek>
[-passphrase <AEKPassphrase>]
```

- When running dm_encrypt_password, use the following syntax:

```
dm_encrypt_password -encrypt password -file broker.pwd -keyname <CSaek>
[-passphrase <AEKPassphrase>]
```

3. Check if the format of the connection broker keystore is PKCS#12.

The following commands list the keys in the keystore if the keystore is in the PKCS#12 format:

Using OpenSSL:

```
openssl pkcs12 -info -in <keystore>
```

Using Keytool:

```
keytool -list -v -storetype pkcs12 -keystore <keystore>
```

4. Check if the password in the keystore password file is correct.



Note: Specify the password in the plain text (without encryption) format, to perform the test.

2.13.5.2 Server startup fails

1. Check if the server keystore, server keystore password, and server trust store files are available in the \$DOCUMENTUM/dba/secure directory.
2. Check if the entry for the server keystore, server keystore password, and server trust store files is available in the server configuration file (server.ini).
3. Check if the format of the server keystore is PKCS#12.
4. Check if the server trust store is in the PKCS#7 binary (der) format. For verification, check if the following command dumps it successfully:

```
openssl pkcs7 -in <Keystore> -inform der
```

Example:

```
openssl pkcs7 -in server-trust.p7b -inform der
```

2.13.5.3 Server not able to connect to connection broker

1. Check whether the connection broker keystore contains the correct key and public certificate.

Use the following command to print the keys in keystore:

```
openssl pkcs12 -info -in <keystore>
```

Using Keytool:

```
keytool -list -storetype pkcs12 -keystore <keystore>
```

2. Check whether the connection broker is sending proper certificates.

The following command starts a simple client that connects to the SSL Server and displays the certificate chain sent by the Server:

```
openssl s_client -showcerts -debug -connect
<SSL_Server_IP>:<SSL_Server_Port>
```

Example:

```
openssl s_client -showcerts -debug -connect 10.8.53.24:1490
```

3. Check if the server trust store contains the connection broker's public certificate or CA certificate chain used to sign connection broker's public certificate.

Use the following command to display all certificates in trust store:

```
openssl pkcs7 -in <trust store> -inform der -print_certs -text
```

4. Check if the server's trust store contains the connection broker's public certificate or CA certificate chain used to sign connection broker's public certificate.

Use the following command to dump the trust store:

```
openssl pkcs7 -in <trust store> -inform der -print_certs -text > CA.PEM
```

Use the following command to check connectivity using this trust store:

```
openssl s_client -showcerts -debug -Cafie CA.pem -connect
<SSL_Server_IP>:<SSL_Server_IP>
```

Example:

```
openssl s_client -showcerts -Cafie CA.pem -debug -connect 10.8.53.24:1490
```

If there are no problems with the trust store , the verification will return (ok).

2.13.5.4 Clients are unable to connect to the connection broker

1. Check if proper entries are available in the dfc.properties file and if the trust store file is available.
2. Check if the Foundation Java API trust store contains the connection broker's public certificate or CA certificate used to sign connection broker's public certificate.
3. Run the following command to dump the trust store contents (do not specify the storetype as the default type is JKS):

```
keytool -list -keystore <keystore> -storepass <storepass>
```

2.13.5.5 Clients are unable to connect to the Server

1. Check if proper entries are available in the `dfc.properties` file and if the trust store file is available.
2. Check if the Foundation Java API trust store contains the server's public certificate or CA certificate used to sign the server's public certificate.
3. Run the following command to dump the contents of the trust store (do not specify the store type as the default type is JKS):


```
keytool -list -keystore <keystore> -storepass <storepass>
```
4. Check if the correct value has been specified for the trust store password in the `dfc.properties` file. For verification, specify the password in plain text.

2.13.5.6 Additional information

Run the following command to verify whether the connection is secure and if the encryption algorithm used while starting the client is correct:

```
jvm parameter -Djavax.net.debug=all
```

2.13.5.7 Documentum CM Server error messages and causes

Error message	Cause
[DM_SERVER_E_START_NETWISE]error: "The server failed to start due to an error returned by the Netwise networking subsystem. Failed API: nl_init(). Error (705) SSL error(10100) in call to SSL_CTX_new(). Winsock error: 10100." [DM_SERVER_E_START_NETWISE]error: "The server failed to start due to an error returned by the Netwise networking subsystem. Failed API: nl_open(). Error (501) Network Library not initialized. Extended network error: 0."	<ul style="list-style-type: none"> • Server keystore is missing • Format of the Server keystore is different from PKCS#12 • Server keystore is corrupt • Server keystore password file contains the wrong password • Server keystore password file is missing
[ERROR] [AGENTEXEC 3088] Detected during program initialization: Command Failed: connect,<server_name.docbase_name>,<user>,",,try_native_first,status: 0, with error message [DM_SESSION_E_RPC_ERROR]error: "Server communication failure" javax.net.ssl.SSLHandshakeException: Received fatal alert: handshake_failure	Server keystore is empty

Error message	Cause
<p>[DM_SERVER_E_START_NETWISE]error: “The server failed to start due to an error returned by the Netwise networking subsystem. Failed API: nl_init(). Error Unknown error code 10008 (_nl_error_ = 0). Extended network error: 0.”</p> <p>[DM_SERVER_E_START_NETWISE]error: “The server failed to start due to an error returned by the Netwise networking subsystem. Failed API: nl_open(). Error (501) Network Library not initialized. Extended network error: 0.”</p>	<ul style="list-style-type: none"> • Server trust store is missing • Server trust store is corrupt
<p>[DM_SERVER_E_START_NETWISE]error: “The server failed to start due to an error returned by the Netwise networking subsystem. Failed API: nl_init(). Error Unknown error code 1795 (_nl_error_ = 0). Extended network error: 0.”</p> <p>[DM_SERVER_E_START_NETWISE]error: “The server failed to start due to an error returned by the Netwise networking subsystem. Failed API: nl_open(). Error (501) Network Library not initialized. Extended network error: 0.”</p>	Server trust store is in the wrong format
<p>[DM_DOCBROKER_E_NETWORK_ERROR]error: “An error occurred performing a network operation: (Unknown error code 112 (_nl_error_ = 0)). Network specific error: (Extended network error: 0.”)</p> <p>[DM_DOCBROKER_E_CONNECT_FAILED_EX]error: “Unable to connect to DocBroker. Clients please check your dfc.properties file for a correct host. Server please check your server.ini or target attributes in server config. Network address: (INET_ADDR: family: 2, port: <port>, host: <host_name> (<host_ip>, 1835080a).”</p>	<ul style="list-style-type: none"> • Server trust store is empty • Connection broker keystore is empty

Error message	Cause
<p>[ERROR] [AGENTEXEC 2692] Detected during program initialization: Command Failed: connect,<server_name.docbase_name>,<user>,",,try_native_first, status: 0, with error message [DFC_DOCBROKER_REQUEST_FAILED] Request to Docbroker <docbroker_name>:<port>" failed</p> <p>[DM_SESSION_E_RPC_ERROR]error: "Server communication failure" javax.net.ssl.SSLException: java.lang.RuntimeException: Unexpected error: java.security.InvalidAlgorithmParameterException: the trustAnchors parameter must be non-empty java.lang.RuntimeException: Unexpected error: java.security.InvalidAlgorithmParameterException: the trustAnchors parameter must be non-empty java.security.InvalidAlgorithmParameterException: the trustAnchors parameter must be non-empty</p>	Foundation Java API trust store is missing

Error message	Cause
<p>[ERROR] [AGENTEXEC 2968] Detected during program initialization: Command Failed: connect,<server_name.docbase_name>>user >,",,try_native_first, status: 0, with error message [DFC_DOCBROKER_REQUEST_FAILED] Request to Docbroker "<docbroker_name>:<port>" failed</p> <p>[DM_SESSION_E_RPC_ERROR]error: "Server communication failure" javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target target[ERROR] [AGENTEXEC 2436] Detected during program initialization: Command Failed: connect,<server_name.docbase_name>,<user >,",,try_native_first, status: 0, with error message [DM_SESSION_E_RPC_ERROR]error: "Server communication failure" javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target target</p>	The Server Certificate is missing in the Foundation Java API trust store

Error message	Cause
<p>[ERROR] [AGENTEXEC 1672] Detected during program initialization: Command Failed:</p> <pre>connect,<server_name.docbase_name>,<user>,",,try_native_first, status: 0, with error message [DFC_DOCBROKER_REQUEST_FAILED] Request to Docbroker "<docbroker_name>:<port>" failed [DM_SESSION_E_RPC_ERROR]error: "Server communication failure" java.net.SocketException: java.security.NoSuchAlgorithmException: Error constructing implementation (algorithm: Default, provider: SunJSSE, class: com.sun.net.ssl.internal.ssl.DefaultSSLContextImpl)java.security.NoSuchAlgorithmException: Error constructing implementation (algorithm: Default, provider: SunJSSE, class: com.sun.net.ssl.internal.ssl.DefaultSSLContextImpl) java.io.IOException: Invalid keystore format</pre>	Foundation Java API trust store is in a different format
<p>[ERROR] [AGENTEXEC 2608] Detected during program initialization: Command Failed:</p> <pre>connect,<server_name.docbase_name>,<user>,",,try_native_first, status: 0, with error message [DFC_DOCBROKER_REQUEST_FAILED] Request to Docbroker "<docbroker_name>:<port>" failed [DM_SESSION_E_RPC_ERROR]error: "Server communication failure" java.net.SocketException: java.security.NoSuchAlgorithmException: Error constructing implementation (algorithm: Default, provider: SunJSSE, class: com.sun.net.ssl.internal.ssl.DefaultSSLContextImpl)java.security.NoSuchAlgorithmException: Error constructing implementation (algorithm: Default, provider: SunJSSE, class: com.sun.net.ssl.internal.ssl.DefaultSSLContextImpl) java.security.cert.CertificateException: Unable to initialize, java.io.IOException: DerInputStream.getLength(): lengthTag=127, too big. java.io.IOException: DerInputStream.getLength(): lengthTag=127, too big.</pre>	Foundation Java API trust store is corrupt

Error message	Cause
[ERROR] [AGENTEXEC 5720] Detected during program initialization: Command Failed: connect,<server_name.docbase_name>,<user>,",,try_native_first, status: 0, with error message [DFC_DOCBROKER_REQUEST_FAILED] Request to Docbroker “<docbroker_name>:<port>” failed [DM_SESSION_E_RPC_ERROR]error: “Server communication failure” java.net.SocketException: java.security.NoSuchAlgorithmException: Error constructing implementation (algorithm: Default, provider: SunJSSE, class: com.sun.net.ssl.internal.ssl.DefaultSSLConte xtImpl)java.security.NoSuchAlgorithmException: Error constructing implementation (algorithm: Default, provider: SunJSSE, class: com.sun.net.ssl.internal.ssl.DefaultSSLConte xtImpl) java.io.IOException: Keystore was tampered with, or password was incorrect java.security.UnrecoverableKeyException: Password verification failed	Foundation Java API trust store contains the wrong password

2.13.6 Recovering from a failed filestore configuration

If the configuration of a filestore to use a mapped drive as the file_system_path fails, the following error message appears:

```
DiUtil: Unable to set permission of file:  
X:\data: Unable to get DACL - Access is denied
```

Documentum CM Server is unable to connect to the drive even though you have full control on the target shared drive. This error does not occur in partitions configured as FAT32.

The reason for this failure is because of the way in which the NTFS security handles the mapped drives as compared to FAT32.

You can recover from the failure by specifying a UNC path as the file_system_path name for the filestore.

To specify a UNC path name for the filestore:

1. Log in to the Documentum CM Server repository.
2. Set the storage area offline.

```
EXECUTE set_storage_state  
WITH store = 'filestore_name', offline = TRUE
```

3. Copy the files in the storage area to the new directory.
4. Update file_system_path property of the dm_location object associated with the file store object to point to the new directory for the storage area.

```
UPDATE "dm_location" OBJECT
SET "file_system_path" = '\\server_name\share'
WHERE "object_name" = location_object_name
```

5. Reinitialize the server to make the change visible.
6. Set the storage area online.

```
EXECUTE set_storage_state
WITH store = 'filestore_name', offline = FALSE
```



Note: You can also use Documentum Administrator to set the storage area offline or online.

2.14 Documentum CM Server installation directories and repository configuration scripts

This section describes the file structure, scripts, and configuration objects that are a part of a Documentum CM Server installation.

2.14.1 Documentum CM Server installation file structure

A Documentum CM Server installation consists of a number of files distributed among several directories. Some of these files, such as the executable files, are supplied as part of the Documentum CM Server installation package. Others, such as the Documentum CM Server startup file, are created during the installation process.

Documentum CM Server installation program or Documentum CM Server configuration program creates the subdirectories in \$DOCUMENTUM directory.

- On Windows: apptoken, cache, Checkout, config, data, dba, Export, fulltext, java64, jmsTools, local, logs, patch, presets, product, server_uninstall, share, Shared, tcf, temp, tools, uninstall, and tomcat
- On Linux: apptoken, cache, checkout, config, data, dba, dfc, export, fulltext, java64, jmsTools, local, logs, patch, presets, product, server_uninstall, share, tcf, temp, tools, uninstall, and tomcat.

2.14.1.1 **uninstall**

This directory contains the Documentum CM Server uninstallation program.

2.14.1.2 **data**

The files and directories in this category are the content storage areas. These directories must exist and location objects must be defined for them in the repository before you start Documentum CM Server. The installation procedure creates a default storage area and associated location object and a default full-text index object and associated location object.

The data directory contains the directories that store the data manipulated by users and Documentum CM Server. The installation procedure creates a subdirectory for the repository in the data directory and in that repository subdirectory, creates a content storage area.

The data includes the full-text indexes and the content files associated with objects in the repositories. The location of these directories is the most flexible component of the configuration.

Most sites will want to add more storage areas and index directories, particularly as the repository grows larger. For more information about adding additional storage areas and full-text index storage directories, see *OpenText Documentum Content Management - Server Administration and Configuration Guide* (EDCCS250400-AGD).

2.14.1.3 **dba**

The dba directory contains the log and config directories and several files.

- The log directory is where the Documentum CM Server places any log files generated by user actions during a session with the Documentum CM Server. The Documentum CM Server creates any necessary subdirectories for these log files under the log directory.
- The config directory includes a subdirectory for each repository that contains the startup files for Documentum CM Server.

2.14.1.4 **fulltext**

The fulltext directory contains the third-party full-text indexing software.

2.14.1.5 product

The product subdirectory contains the Documentum CM Server executables.

2.14.1.6 server_uninstall

This directory contains a script that you can run manually to destroy a repository's database tables after you delete the repository.

2.14.1.7 share

The share directory holds all the files that can be shared by the Documentum CM Server and the clients. Clients that connect to the share directory remotely can benefit in file sharing and event notification. The client must use the NFS software to receive these benefits. For more information about the share directory, see *OpenText Documentum Content Management - Server Administration and Configuration Guide* (EDCCS250400-AGD).

data: The data directory contains the data that is read and written by the Documentum CM Server and the clients. The data directory has several subdirectories. Make sure that these subdirectories can be mounted by clients.

- events

The events subdirectory contains a file for any user who has queued inbox items that have not been viewed. The files are empty. They serve as a flag to the Documentum CM Server that items that have not been viewed are in that user's inbox.

- common

The common subdirectory is where the Documentum CM Server puts copies of requested content files if users are not using client local areas and if users do not specify an alternate location for the files.

- clients

The clients subdirectory contains the win and Linux subdirectories, which respectively contain the files and executable for Windows and Linux clients.

- temp

The temp subdirectory is used by the Documentum CM Server as a temporary storage space. For example, results generated by the execution of a procedure by using the Apply method's DO_METHOD function are stored here.

- sdk

The sdk subdirectory contains two subdirectories of files that are useful to software developers. The two subdirectories are:

- Include

This subdirectory contains the dmapp.h file and the import libraries.

- example

This subdirectory contains the code examples.

2.14.1.8 Additional directories

Directory	Description
bin	Contains the Documentum CM Server software.
convert	Contains the transformation engine executable files.
dba/auth	<p>When you install Documentum CM Server, a default base directory is created. Under default base directory, installer creates a subdirectory specific to the repository. The repository configuration also creates an auth_plugin location object that points to the base directory and sets the auth_plugin_location attribute in the server to the name of the location object. Any plug-in installed in this directory is loaded into every server at startup for all repositories. To use a plug-in only with a particular repository, place the plug-in in the repository-specific dba/auth directory.</p> <p>When Documentum CM Server starts, it loads the plug-ins found in its repository-specific directory first and then those found in the base directory.</p>

Directory	Description
dba/secure/ldapdb	<p>Contains the secure connection attributes. You need to define the following setup values to use a secure LDAP connection:</p> <ul style="list-style-type: none"> • SSL mode • SSL port • Certificate database location <p>The <code>ssl_mode</code> attribute in the <code>ldap_config</code> object defines whether the LDAP server is using a secure or nonsecure connection. You need to set this when defining the LDAP setup values.</p> <p>To configure a secure connection, chose Secure as the SSL mode. When you do, the interface lets you edit the SSL port field. SSL port, represented in the <code>ldap config</code> object by the <code>ssl_port</code> attribute, identifies the port the LDAP server uses for the secure connection. This value is 636 by default.</p> <p>Certificate database location, represented in the <code>ldap_config</code> object by the <code>certdb_location</code> attribute, identifies the location of the certificate database. The attribute value is the name of the location object pointing to the certificate database. The value is <code>ldapcertdb_loc</code>. The directory that <code>ldapcertdb_loc</code> points to is <code>DOCUMENTUM\dba\secure\ldapdb</code>.</p>
example*	Contains the code examples.
external_apps	Contains a shared library.
include*	Contains the header files for any external applications that will communicate with Documentum CM Server.
install	Contains the installation scripts.
java64	Contains the symbolic link to the installation location of Java specified during Documentum CM Server installation.
messages	*.e files (error messages).
Oracle	Contains the language files needed by Oracle. During installation, the environment variable <code>ORA_NLS33</code> is set to that location. Do not remove that directory or reset the that variable.

Directory	Description
tcf	References the task chaining framework, which is related to lifecycles and is part of the BPS and BPM group.
Uniscape	Contains the NLS files for server code page conversions.
Linux*	Contains the libraries for a Linux client.
unsupported	Contains the executable files that are provided for your convenience but that are not supported by Documentum CM Server.
webcache	Includes webcache.ini. The documentation for Documentum Interactive Delivery Services/Interactive Delivery Services Accelerated contains the details.
thumsrv	Installation directory for Thumbnail Server.
win*	Contains the executable files for a Microsoft Windows client. These include IAPI and IDQL for MS Windows and the DDE server and libraries.
tomcat	Contains the application server installation files used to create an instance for the Java Method Server.

An asterisk (*) indicates that the directory is optional.

2.14.1.9 Scripts run during installation or upgrade

During repository configuration, the following scripts are run, whether you are installing a new repository or upgrading an existing repository:

Script name	Location	Purpose	Other
headstart.ebs	\$DM_HOME/install/admin	Loads the initial default objects for the repository. Creates mount point objects, location objects, file store objects, and method objects.	
dm_apply_formats.ebs	\$DM_HOME/bin	Creates or updates format objects, which are required for content file operations.	

Script name	Location	Purpose	Other
dm_cas_install.ebs	\$DM_HOME/install/admin	Creates a method, location, template type, folder structure, and template object for use of the electronic signature.	
csec_plugin.ebs	\$DM_HOME/install/admin		
dm_routerconv_install.ebs	\$DM_HOME/install/admin	Loads methods that are used for converting routers to workflow template.	Run only during an upgrade.
templates.ebs	\$DM_HOME/install/admin	Creates default templates that are used by OpenText Documentum CM clients for creating the new documents in the repository.	
replicate_bootstrap.ebs	\$DM_HOME/bin	Creates objects and registered tables that are required for replication.	
desktop_client.ebs	\$DM_HOME/install/desktop_client	Creates folders required by Documentum Desktop and installs the default SmartList.	
disable_fulltext_jobs.ebs	\$DM_HOME/install/admin		Run only during an upgrade.
dfc.ebs	\$DM_HOME/install/admin	Loads default objects required by the Foundation Java API.	
Dfc_bof2.ebs	\$DM_HOME/install/admin	Creates the types for dm_module, dmc_jar, and dmc_java_library and configures a repository to use Foundation Java API version 5.3 SP1 and later.	

Script name	Location	Purpose	Other
dfc_javadbexpr.ebs	\$DM_HOME/install/admin	Creates types, relation types, acls, and repository folders for Foundation Java API evaluation of validation expression constraints in Java.	
dm_bpmmODULES_INSTALL.ebs	\$DM_HOME/install/admin		
createMethodServerObject.ebs	\$DM_HOME/install/admin		
csec_plugin_upgrade_53.ebs	\$DM_HOME/install/admin	Upgrades the plug-in for using content-addressable storage areas.	Run only during an upgrade.
toolset.ebs	\$DM_HOME/install/admin	Installs repository administration tools.	
dm.bpm_install.ebs	\$DM_HOME/install/admin		
dm_wfTimer_upgrade.ebs	\$DM_HOME/install/admin	Converts workflow pre- and post-timers set up in repositories prior to version 5.3 to the version 6.5 timer implementation. <code>dmbasic -f dm_wfTimer_upgrade.ebs -e Install --<repository_name><user><password></code>	Run only during an upgrade.
dm_setup_java_lifecycle.ebs	\$DM_HOME/install/admin		
create_fulltext_objects.ebs	\$DM_HOME/install/admin	Creates repository objects related to full-text indexing.	
dm_ldap_install.ebs	\$DM_HOME/install/admin	Creates or upgrades the ldap config object type and upgrades any existing ldap config objects.	
dm_storageservices_install.ebs	\$DM_HOME/install/admin		
dm_emailTemplate_install.ebs	\$DM_HOME/install/admin		

Script name	Location	Purpose	Other
dm_xml_install.ebs	\$DM_HOME/install/admin	Installs object types and formats for XML files.	
dm_gwm_install.ebs	\$DM_HOME/bin	Executes scripts that install workflow-related types, methods, folders, and jobs.	
upgrade_java_methods_51.ebs	\$DM_HOME/install/tools	Upgrades existing Java methods.	
ci_schema_install.ebs	\$DM_HOME/install/tools	Installs the object types used by Documentum Content Intelligence Services.	
display_config_setup.ebs	\$DM_HOME/install/tools	Configures the repository for the Documentum Offline Client.	
offline_config_setup.ebs	\$DM_HOME/install/tools	Migrates offline configuration settings from the offline_config object to the docbase config object.	
upgrade_contentreplication_job.ebs	\$DM_HOME/install/admin		Run only during an upgrade.
dm_acs_install.ebs	\$DM_HOME/install/admin		
dd_populate.ebs	\$DM_HOME/bin	Populates the data dictionary with attribute and type information from datafiles.	
dm_extuser_table.ebs		Creates the dm_extuser_transaction table and then register with the specified permission.	

2.14.1.10 Configuration objects

Each repository contains the objects that together define your configuration. These objects include:

- Server config object
- Docbase config object
- Fulltext index objects
- Location objects
- Mount point objects
- Storage objects
- Format objects
- Method objects

As you make choices about how to configure the installation and repositories, modify these objects or add a new ones. For more information about the configuration, see *OpenText Documentum Content Management - Server Administration and Configuration Guide (EDCCS250400-AGD)*.

2.15 Object type categories for Oracle database storage

This section lists the object types by their size category. An object type's size category is used in two contexts:

- To determine where to create the object type's tables and indexes if the optional [FUNCTION_SPECIFIC_STORAGE] parameters are defined in the server.ini file
- To determine the default initial and next extent allotments for the object type's tables in the RDBMS

The categories for each context are not the same.

2.15.1 Type categories for tablespace specifications

By default, the tables and indexes for all object types are created in the same tablespace. However, you can set parameters in the server.ini file to define alternate tablespaces for large and small object types. When you do so, the system sorts the objects into large and small for the purposes of determining which object types to create in which tablespace.

The majority of the object types are considered small for this purpose. The following list shows the object types that are considered large. Any type not appearing on this list is considered small.

dm_acl	dm_process	dmi_dump_object_record
dm_assembly	dm_reference	dmi_linkrecord
dm_audittrail	dm_relation	dmi_load_object_record
dm_composite	dm_router	dmi_otherfile
dm_document	dm_script	dmi_queue_item
dm_folder	dm_smart_list	dmi_replica_record
dm_locator	dm_sysobject	dmrContainment
dm_note	dm_workflow	dmr_content
dm_procedure	dm_workitem	

2.15.2 Type categories for extent allocation

This section lists object type size categorizations for extent allocation and the default initial and next extent storage parameters for each category.

The object types are categorized as large, small, or default based on how many objects of the type will be created in the repository. For example, dm_document is categorized as large because most enterprises create large numbers of documents in a repository. Similarly, dm_docbase_config is categorized as small because a repository has only one docbase config object. Those types that do not fall into either the large or small category are categorized as default.

2.15.2.1 Object types categorized as large

The object types categorized as large are created with an initial extent size of 100 K. The next extent size is 1 M. The following object types are categorized as large for the purposes of allocating extents:

dm_acl	dm_reference	dmi_dump_object_record
dm_assembly	dm_relation	dmi_load_object_record
dm_document	dm_router	dmi_object_type
dm_folder	dm_sysobject	dmi_queue_item
dm_locator	dmiContainment	dmi_replica_record
dm_note	dmr_content	dmi_subcontent

2.15.2.2 Object types categorized as small

The object types categorized as small are created with an initial extent size of 10 K. The next extent size is 50 K. The following object types are categorized as small for the purposes of allocating extents:

dm_alias	dm_filestore	dm_relation_type
dm_blobstore	dm_foreign_key	dm_server_config
dm_distributed_store	dm_format	dm_store
dm_dump_record	dm_fulltext_index	dmi_change_record
dm_docbase_config	dm_linkedstore	dmi_expr_code
dm_docbaseid_map	dm_load_record	dmi_recovery
dm_extern_file	dm_location	dmi_session
dm_extern_free	dm_mount_point	dmi_sequence
dm_extern_store	dm_opticalstore	dmi_tdk_collect
dm_extern_url	dm_outputdevice	dmi_tdk_index
dm_federation	dm_registered	dmi_vstamp

2.15.2.3 Object types categorized as default

The object types categorized as default are created with an initial extent size of 20K. The next extent size is 100K. The following object types are categorized as default for the purposes of allocating extents:

dm_acs_config	dm_cabinet	dm_domain
dm_activity	dm_client_registration	dm_expression
dm_aggr_domain	dm_client-rights	dm_federation_log
dm_application	dm_cond_expr	dm_func_expr
dm_app_ref	dm_cond_id_expr	dm_group
dm_aspect_type	dm_component	dm_job
dm_audittrail	dm_cont_transfer_config	dm_job_request
dm_bocs_config	dm_dd_attr_info	dm_key
dm_builtin_expr	dm_dd_info	dm_lightweight
dm_cabinet	dm_dms_config	dm_user
dm_literal_expr	dm_public_key_certificate	dm_value_assist
dm_media_profile	dm_query	dm_value_func
dm_method	dm_qual_comp	dm_value_list
dm_nls_dd_info	dm-retainer	dm_value_query

dm_plugin	dm_script	dm_validation_descriptor
dm_policy	dm_smart_list	dm_workflow
dm_process	dm_staged	dm_workitem
dm_procedure	dm_type	dmc_wfsd_element_string
dmc_aspect_type	dmc_relationship_def	dmc_wfsd_parent
dmc_calendar	dmc_routcase_condition	dmc_wfsdrp_boolean
dmc_calendar_event	dmc_scope_config_relation	dmc_wfsdrp_date
dmc_completed_workitem	dmc_transition_condition	dmc_wfsdrp_double
dmc_datatable	dmc_type_info	dmc_wfsdrp_integer
dmc_datatable_row	dmc_wf_package_skill	dmc_wfsdrp_parent
dmc_datatable_schema	dmc_wfsd_element	dmc_wfsdrp_string
dmc_datatable_settings	dmc_wfsd_element_boolean	dmc_workqueue
dmc_module	dmc_wfsd_element_date	dmc_workqueue_policy
dmc_preset_info	dmc_wfsd_element_double	dmi_autittrail_attrs
dmc_preset_package	dmc_wfsd_element_integer	dmi_registry
dmi_dist_comp_record	dmi_package	dmi_wf_timer
dm_expr_code	dmi_transactionlog	dmi_workitem

2.16 Defining Oracle database parameters for repository tables

To improve performance and increase the throughput of the system, you might want to control where repository information is stored. For example, you can store frequently used data on different disks than less frequently used data. Defining database parameters to store data in different tablespaces also partitions data into smaller, more manageable pieces.

When a repository is created, the system automatically creates object-type tables and indexes in the underlying database. For more information about the object-type tables and indexes, see *OpenText Documentum Content Management - Server Fundamentals Guide (EDCCS250400-GGD)*.

If you do an express installation of Documentum CM Server, by default, Documentum CM Server creates all object-type tables and indexes in the same tablespace. The size and number of the extents allotted for each table are determined by default configuration parameters. If you do a custom Documentum CM Server installation, you are prompted to configure the object-type tables and indexes, and you can create them in separate tablespaces.

You can edit the `server.ini` file to change any configuration parameters when the repository is created, before you start the Documentum CM Server.

- On Oracle, you can change two parameters:
 - The tablespace for the object-type tables and indexes
 - The size of the extents allotted for system-defined object types

You cannot change the number of extents allotted for the object types.
- Under Oracle 10, always create tablespaces as locally managed tablespaces (LMTs) using the LOCAL value. If you have dictionary managed tablespaces (DMTs) under Oracle 10, use the Oracle DBMS_SPACE_ADMIN package to convert DMTs to LMTs, for example,

```
SQL> exec dbms_space_admin.Tablespace_Migrate_TO_Local('Table_space1');
```

The *Oracle* documentation set contains the details on extent management and DMT-to-LMT conversion.

2.16.1 Defining the tablespace

The parameters in the [FUNCTION_SPECIFIC_STORAGE] and [TYPE_SPECIFIC_STORAGE] sections of the server.ini file define the tablespace in which to create the object-type tables and indexes.

2.16.1.1 FUNCTION_SPECIFIC_STORAGE

Set the parameters in the [FUNCTION_SPECIFIC_STORAGE] section to define the tablespace for the type tables and indexes for a particular category of object types. OpenText Documentum CM sorts object types into the categories large and small for the purposes of defining their tablespace.

- Object types in the large category are those that are expected to have a large number of object instances. For example, dm_SysObject is in the large category.
- Object types in the small category are those that are expected to have very few object instances. For example, dm_docbase_config is in the small category. Each repository has only one Docbase config object.

The format of the [FUNCTION_SPECIFIC_STORAGE] section is:

```
[FUNCTION_SPECIFIC_STORAGE]
database_table_large=<tablespace_name>
database_table_small=<tablespace_name>
database_index_large=<tablespace_name>
database_index_small=<tablespace_name>
```

For example, to define a tablespace for the object-type tables in the large category, include the following lines in the server.ini file, substituting the name of the tablespace:

```
[FUNCTION_SPECIFIC_STORAGE]
database_table_large=<tablespace_name>
```

For example, to put the indexes for the large category in the tablespace named production_1, include the following lines in the server.ini file:

```
[FUNCTION_SPECIFIC_STORAGE]
database_index_large=production_1
```

You can specify the function-specific parameters singularly or in any combination.

2.16.1.1.1 TYPE_SPECIFIC_STORAGE

Set the parameters in the [TYPE_SPECIFIC_STORAGE] section in the `server.ini` file to define a tablespace for the type tables or indexes for a specific object type.

The format of the [TYPE_SPECIFIC_STORAGE] section is:

```
[TYPE_SPECIFIC_STORAGE]
database_table_<typename>=<tablespace_name>
database_index_<typename>=<tablespace_name>
```

You can specify the type-specific parameters individually. For example, to put the object-type tables for the `dm_SysObject` type the tablespace named `sysobj_space`, include the following lines in the `server.ini` file:

```
[TYPE_SPECIFIC_STORAGE]
database_table_dm_sysobject=sysobj_space
```

If you want to put both the tables and indexes for an object type nondefault tablespaces, define the tablespace for each. Defining a tablespace for an object type's tables does not affect where the type's indexes are stored. The system creates the indexes in the default tablespace. Defining a tablespace for a type's indexes does not affect where the type's tables are stored.

For example:

```
[TYPE_SPECIFIC_STORAGE]
database_table_<dm_sysobject>=<sysobj_space>
database_index_<dm_sysobject>=<sysobj_idx_space>
```

The object-type tables and indexes of any object type not specified in a type-specific parameter are created in the default tablespace or, if specified, in the tablespace for the type's category.

If the `server.ini` file includes both function-specific and type-specific parameters that apply to an object type, the type-specific parameters override the function-specific parameters. For example, suppose you add the following function-specific and type-specific parameters to the file:

```
[FUNCTION_SPECIFIC_STORAGE]
database_index_large=production_1
[TYPE_SPECIFIC_STORAGE]
database_table_dm_sysobject=sysobj_space
```

Both parameters apply to the `dm_SysObject` type because `dm_SysObject` is in the large category. The object-type tables for `dm_SysObject` are created in the `sysobj_space` tablespace because the type-specific parameter overrides the function-specific parameter.

2.16.1.2 Defining the Oracle extent sizes

For the purposes of extent allocation, the OpenText Documentum CM object types are sorted into three categories: large, small, and default. The category name describes the quantity of expected objects of the type. For example, dm_document is considered a large type because most enterprises generate large quantities of documents. In contrast, dm_repository_config is a small type because there is only one docbase config object in a repository. Those object types that typically do not have large numbers of objects or very small numbers of objects fall into the default category.

A type's category determines how much database storage is allocated to it by default. Object types categorized as:

- Large object types receive an initial extent of 100 KB and a next (second, third, fourth.) extent of 1 MB.
- Small object types receive an initial extent of 10 KB and a next extent of 50 KB.
- Default object types receive an initial extent of 20 KB and a next extent of 100 KB.

The default storage parameters set the initial and next extent sizes. There are also parameters that define the default minimum and maximum number of extents allotted to an object type table and the percentage increase of extents allotted after the second extent. The minimum number of allotted extents is 1 and the maximum number is determined by Oracle, based on the data block size. By default, object-type tables and indexes are allocated the minimum number of extents when they are created.

The percentage increase default is 10 percent. This means that extents allotted after the second extent are increased in size by 10 percent over the previously allocated extent. For example, if the second extent's size is 100 KB, then the size of the third extent is 110 KB, 10 percent greater than 100 KB. The fourth extent would be 121 KB, 10% greater than 110 KB.

You can change the initial and next extent default sizes for an individual object type or for an entire category by setting parameters in the `server.ini` file before the repository is created.

You can change the following parameters by using the Oracle ALTER TABLE command through sqlplus:

- Next extent
- Minimum extent
- Maximum extent
- Percentage increase

The *Oracle* documentation contains the instructions.

2.16.1.3 Changing storage parameters for individual object types on Oracle

To change the initial and next extent parameters for an object type, add a [TYPE_EXTENT_SIZE] section to the server.ini file. This section has the following format:

```
[TYPE_EXTENT_SIZE]
database_ini_ext_<typename>=<new_value>[K|M]
database_next_ext_<typename>=<new_value>[K|M]
```

where

- <typename> must be the internal name of a system-defined object type. It cannot be a user-defined object type.
- The *database_ini_ext_typename* parameter defines the size of the initial extent allotted to the type.
- The *database_next_ext_typename* parameter defines the size of the second extent allotted to the type.
- *new_value* is an integer. If you include K, the value is interpreted as Kilobytes. If you include M, the value is interpreted as Megabytes. If you include neither K nor M, the value is interpreted as bytes.

For example, to change the defaults for dm_sysobject, add the following lines to the server.ini file:

```
[TYPE_EXTENT_SIZE]
database_ini_ext_dm_sysobject=new_value[K|M]
database_next_ext_dm_sysobject=new_value[K|M]
```

You can set either parameter or both for an object type. The section can include parameter definitions for more than one object type. For example:

```
[TYPE_EXTENT_SIZE]
database_ini_ext_dm_sysobject=new_value[K|M]
database_next_ext_dm_sysobject=new_value[K|M]
database_next_ext_dm_user=new_value[K|M]
```

2.16.1.4 Changing storage parameters for categories of types on Oracle

To change the initial and next extent parameters for all object types in one category, add a [FUNCTION_EXTENT_SIZE] section to the server.ini file. This section has the following format:

```
[FUNCTION_EXTENT_SIZE]
database_ini_ext_large=new_value[K|M]
database_ini_ext_small=new_value[K|M]
database_ini_ext_default=new_value[K|M]
database_next_ext_large=new_value[K|M]
database_next_ext_small=new_value[K|M]
database_next_ext_default=new_value[K|M]
```

- The *database_ini_ext_large* parameter defines the size of the initial extent allotted by default to object types categorized as large.
- The *database_ini_ext_small* parameter defines the size of the initial extent allotted by default to object types categorized as small.

- The *database_ini_ext_default* parameter defines the size of the initial extent allotted by default to object types categorized as default.
- The *database_next_ext_large* parameter defines the size of the second extent allotted by default to object types categorized as large.
- The *database_next_ext_small* parameter defines the size of the second extent allotted by default to object types categorized as small.
- The *database_next_ext_default* parameter defines the size of the second extent allotted by default to object types categorized as default.
- *new_value* is an integer. If you include K, the value is interpreted as Kilobytes. If you include M, the value is interpreted as Megabytes. If you include neither K nor M, the value is interpreted as bytes.

For example, to change the default extent sizes for all large object types, add the following to the `server.ini` file:

```
[FUNCTION_EXTENT_SIZE]
database_ini_ext_large=new_value[K|M]
database_next_ext_large=new_value[K|M]
```

You can set any combination of the parameters. It is not necessary to set the parameters for all three categories. You can also set only one of the parameters for a category. To illustrate, the following example sets the initial extent for objects categorized as large and the next extent for object types categorized as default:

```
[FUNCTION_EXTENT_SIZE]
database_ini_ext_large=200K
database_next_ext_default=120K
```

2.16.1.5 User-defined object types

A user-defined object type derives its database storage parameters from its supertype. If the type has no supertype, then the type is assigned to the large category for tablespace assignment and to the default category for the extent allocations.

You cannot change the storage parameters for user-defined object types.

2.17 Independent Java Method Server

A separate installer for Independent Java Method Server is available for Documentum CM Server. You can install and configure additional Java Method Server using Independent Java Method Server. Independent Java Method Server is lightweight and granular as it just deploys the Tomcat application server. In addition, it enhances scalability with almost similar performance as remote Content Server. Independent Java Method Server also supports all the load balancing and HA configurations as described in “[Supported load balancing and HA configurations](#)” on page 121.



Note: XHive operations do not work with Independent Java Method Server.

2.17.1 Installing, configuring, and uninstalling Independent Java Method Server

You can install, configure, and uninstall Independent Java Method Server using both the graphical user interface (GUI) and the command line (silent or unattended) options.

2.17.1.1 Pre-installation requirements and tasks

- Download and install the supported version of Oracle JDK/OpenJDK from the Oracle JDK/OpenJDK website. “[Installing JDK](#)” on page 35 contains more information.
- Set the JAVA_HOME environment variable to point to the installed Oracle JDK/OpenJDK version.
- Set the PATH environment variable to JAVA_HOME\bin.
- Remove all anon entries from the list of disabled algorithms. “[Using SSL communication](#)” on page 36 contains more information.
- Use a 64-bit machine.
- Independent Java Method Server installed machine must be in the same time zone as in Documentum CM Server machine.
- Install both the Independent Java Method Server and Documentum CM Server on machines with same operating system.
- Installation owner of both Independent Java Method Server and Documentum CM Server must be same even on different environments.
- (Only for Linux) Make sure that the required RPM packages are installed.

2.17.1.2 Installing Independent Java Method Server using GUI

1. If you want to use HashiCorp Vault, perform all the tasks as described in “[Configuring Documentum Secret Integration Service](#)” on page 49 and make sure that DSIS is running.
2. Run jmsStandaloneSetup.exe (Windows) or jmsStandaloneSetup.bin (Linux) to launch Independent Java Method Server installer.
3. Accept the license agreement and click **Next**.
4. If you want to configure the HashiCorp Vault secrets, on the **Vault configuration** page, select the **Enable Vault** check box.
5. If you select the **Enable Vault** check box, provide the following information and click **Next**:
 - a. **DSIS URL:** Provide a value in the following format:
`http://localhost:<port mentioned in application.properties>/dsis`

- b. **DSIS Token:** Provide the `dsis.dctm.token` token value as described in “[Configuring Documentum Secret Integration Service](#)” on page 49.

! **Important**

If you enabled the HashiCorp Vault configuration, the installer retrieves all the password information automatically from the HashiCorp Vault server. Make sure that you have stored all the required secrets as described in “[Configuring Documentum CM Server to use HashiCorp Vault](#)” on page 38.

6. (Only for Windows) Specify an installation directory for Independent Java Method Server. The installation directory name must not contain spaces. For example, do not use `c:\Documentum Products` as the name of the installation directory.

On Linux, export the `JMS_INSTALL_DIR` and `DOCUMENTUM` environment variables.

Click **Next**.

7. On Windows, provide the installation owner password and click **Next**.

- If HashiCorp Vault is enabled, then the installation owner password is retrieved from the HashiCorp Vault server.
- If HashiCorp Vault is not enabled, then type the installation owner password.

8. Set the administrator password and specify an available listening port for the embedded application server used by Independent Java Method Server:

- **Admin User Password:** Make sure that you follow the password complexity rules. “[Password complexity rules](#)” on page 52 contains detailed information.
 - If HashiCorp Vault is enabled, then the administrator user password is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, then type the administrator user password.

- **Listen Port:** The port on which the application server listens for standard administration connections. A total of 20 ports, starting from the one you specify, is used by the application server, and all of them must be available.

The default port number is 9180. You must change the default port to another one and make sure that the specified port is available.

Click **Next**.

9. Browse and specify the directory where the supported version of JDK is installed and click **Next**.

10. Review the installation summary and click **Install**.

11. Installer prompts you if you want to configure Independent Java Method Server.

- **Configure now:** Configure now and have the installer launch the configuration program. If you select this option, skip to [step 3](#).
- **Configure after the installation:** Configure after the installation and exit the installer. If you select this option, skip to [step 2](#).



Note: If Documentum CM Server is enabled with SSL communication, copy the \$DOCUMENTUM\dba\secure\dfc.keystore file from Documentum CM Server and place it in the Independent Java Method Server host machine. Update the copied dfc.keystore file path and the truststore_password value.

If HashiCorp Vault is enabled, provide the secret ID as described in [“Configuring Documentum CM Server to use HashiCorp Vault” on page 38](#), otherwise type the trust store password in the dfc.properties file in the Independent Java Method Server host machine. In addition, if HashiCorp Vault is enabled, you must update the following entries in the dfc.properties file:

```
dfc.dsds.enabled=true  
dfc.dsds.daemon.url=http://localhost:8200/dsis  
dfc.dsds.daemon.token=<token generated for the DSIS daemon agent>
```

2.17.1.3 Configuring Independent Java Method Server using GUI

1. If you want to use HashiCorp Vault, perform all the tasks as described in [“Configuring Documentum Secret Integration Service” on page 49](#) and make sure that DSIS is running.
 2. Launch the configuration program and resume with the configuration as follows:
 - Windows: Run jmsStandaloneConfig.exe
 - Linux: Run jmsStandaloneConfig.bin
 3. If you want to configure the HashiCorp Vault secrets, on the **Vault configuration** page, select the **Enable Vault** check box.
 4. If you select the **Enable Vault** check box, provide the following information and click **Next**:
 - a. **DSIS URL:** Provide a value in the following format:
`http://localhost:<port mentioned in application.properties>/dsis`
 - b. **DSIS Token:** Provide the dsis.dctm.token token value as described in [“Configuring Documentum Secret Integration Service” on page 49](#).
- ! Important**
If you enabled the HashiCorp Vault configuration, the installer retrieves all the password information automatically from the HashiCorp Vault server. Make sure that you have stored all the required secrets as described in [“Configuring Documentum CM Server to use HashiCorp Vault” on page 38](#).

5. Perform the configuration depending on the following scenarios:
 - If you are installing Independent Java Method Server in a machine that has Documentum CM Server, go to **step 8** and proceed with the configuration.
 - If you are installing Independent Java Method Server in a machine that does not have Documentum CM Server, start from **step 6** and proceed with the configuration.
 6. Type the connection broker host and port and click **Next**. The default port is 1489.
 7. Provide the global registry repository details. Type the repository name and password and then click **Next**.
 - If HashiCorp Vault is enabled, then the global registry password is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, then type the global registry password.
 8. Provide the installation owner password and click **Next**.
 - If HashiCorp Vault is enabled, then the installation owner password is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, then type the installation owner password.
-  **Note:** Make sure that you use the Documentum CM Server installation owner credentials.
9. Type the fully qualified domain name (FQDN) of the Independent Java Method Server host.

 **Note:** Update /etc/hosts of Documentum CM Server machine with FQDN to provide host name mapping, if needed.
 10. Select the repositories that the Java Method Server node serve to and click **Next**.
 11. Click **Done** to finish the configuration.
 12. Restart the repositories.
-  **Note:** Some Java Methods (for example, ESIGN methods) must be run only on Java Method Server located in the same host as Documentum CM Server.

2.17.1.4 Installing Independent Java Method Server using command line

The silent installation invokes the installation program from the command line and provides a configuration file enabling the installation to proceed without further interaction.

2.17.1.4.1 Creating the silent response files

Use the following commands to generate the silent response files during the GUI installation:

- On Windows: `jmsStandaloneSetup.exe -r <silent response installation file name>` and `jmsStandaloneConfig.exe -r <silent response configuration file name>`
- On Linux: `jmsStandaloneSetup.bin -r <silent response installation file name>` and `jmsStandaloneConfig.exe -r <silent response configuration file name>`

These commands run the installation program interactively and records the information during the real-time installation.

After the silent response installation file and silent response configuration file are generated, update the files with the appropriate values, and then install Independent Java Method Server using the command line as described in “[Installing and configuring Independent Java Method Server using the command line](#)” on page 172.

2.17.1.4.2 Installing and configuring Independent Java Method Server using the command line

You must use this procedure cautiously and only if you cannot use the GUI installation procedure as described in “[Installing Independent Java Method Server using GUI](#)” on page 168. You can use `jms_install.properties`, the silent response file template packaged with Independent Java Method Server.

1. Provide the appropriate values as mentioned in `jms_install.properties` or created silent response installation file and then run the silent installation command as follows:
 - On Windows: `jmsStandaloneSetup.exe -f <path of jms_install.properties or silent response installation file>`
 - On Linux: `./jmsStandaloneSetup.bin -f <path of jms_install.properties or silent response installation file>`



Note: For silent installation of Independent Java Method Server in Oracle Linux or Red Hat Enterprise Linux, you must export the `JMS_INSTALL_DIR` and `DOCUMENTUM` environment variables. In addition, make sure that you do not export the `DISPLAY` environment variable.

2. Provide the appropriate values in the `jms_config.properties` file located in the Independent Java Method Server installation directory (`<JMS_INSTALL_DIR>`) and then run the silent configuration as follows:
 - On Windows: `jmsStandaloneConfig.exe -f <path of jms_config.properties or silent response configuration file>`
 - On Linux: `./jmsStandaloneConfig.bin LAX_VM "<JMS_INSTALL_DIR>/Java64/<JDK_VERSION>/jre/bin/java" -f <path of jms_config.properties or silent response configuration file>`

 **Note:** On Linux, you must explicitly specify the configuration to use the supported version of Oracle JDK/OpenJDK with Independent Java Method Server.
3. Restart the repositories.

2.17.1.5 Enabling privileged DFC for custom web applications deployed in Independent Java Method Server

There are some methods which need trusted connection to Documentum CM Server. If trusted authentication is not enabled, the `DM_SESSION_E_AUTH_FAIL` error occurs in the Tomcat application server log file. By default, Independent Java Method Server deploys the `ACS.war`, `documentum-bocs-ws.war` and `ServerApps.zip` files. Independent Java Method Server installer takes care of privilege approval of these web applications. If you want to deploy Foundation Java API based custom WAR file inside the Independent Java Method Server method server, you need to manually approve the generated Foundation Java API keystore file.

Perform the following steps:

1. Navigate to the location of the `dfc.properties` file and check if `dfc.keystore` is available.
2. Run the `keytool -list -keystore dfc.keystore -v` command with the keystore password as `dfc`.

For example:

```
C:\Documentum\config>keytool -list -keystore dfc.keystore -v
Enter keystore password:

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: dfc
Creation date: Jan 19, 2018
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=dfc_c05b0UEM8RSfxHcitET32tfGqPwa, O=TEST, OU=Documentum
Issuer: CN=dfc_c05b0UEM8RSfxHcitET32tfGqPwa, O=TEST, OU=Documentum
Serial number: -38b39e6d32ad3651c5a7cea4f80499c4
Valid from: Fri Jan 19 14:24:45 IST 2018 until: Mon Jan 17 14:29:45 IST 2028
Certificate fingerprints:
```

```
MD5:  
E5:7A:2C:55:53:B8:D0:B3:E4:7B:D5:CC:36:B7:4B:2E  
SHA1:  
E8:6A:D2:47:B7:0D:13:2C:FC:4A:1D:8B:3D:77:1F:C2:9B:CD:A3:C7  
SHA256:  
7A:55:8E:C1:67:B1:AD:E8:F3:A2:22:02:F9:72:79:15:62:5C:EA:F3:  
13:A7:33:F9:7B:B6:84:8C:57:7C:C5:B5  
Signature algorithm name: SHA256withRSA  
Version: 1
```

3. Use CN dfc_c05b0UEM8RSfXHcitET32tfGqPwa to view and approve the Foundation Java API instance through Documentum Administrator.



Notes

- There are scenarios where dfc.keystore is deleted from ACS\WEB-INF\classes or DmMethods\WEB-INF\classes. In addition, the Independent Java Method Server installer might not be able to find dfc.keystore and you might find a keystore is not found warning in the install.log file. In these cases too, follow the instructions as described in [“Enabling privileged DFC for custom web applications deployed in Independent Java Method Server” on page 173](#) to manually approve the respective Foundation Java API instance.
- In Tomcat, to avoid JAR conflicts while deploying custom applications, OpenText recommends you to package the required JAR files inside the custom applications. For example: C:\Documentum\tomcat\webapps\serverapps\WEB-INF\lib. If this method JAR requires two WAR files, then you must package the locations of both the \WEB-INF\lib folders.

2.17.1.6 Configuring for SSL communication

To configure for SSL communication, perform the same steps as described in [“Configuring Java Method Server for certificate-based SSL communication” on page 99](#).

2.17.2 Uninstalling Independent Java Method Server

2.17.2.1 Using GUI

1. Run the Independent Java Method Server uninstallation program (`Uninstall.exe` or `Uninstall.bin`) located in <JMS_INSTALL_DIR>/uninstall/jms/.
2. Click **Next**.
3. In the configuration page, provide the primary Documentum CM Server installation owner information and click **Uninstall**.
 - If HashiCorp Vault is enabled, then type the primary Documentum CM Server installation owner user name. However, the primary Documentum CM Server installation owner password is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, then type the primary Documentum CM Server installation owner user name and password.

4. After the successful uninstallation, click **Done**.
5. Delete the Independent Java Method Server installation directory folder.
6. Restart the repositories.

2.17.2.2 Using the command line

1. Write a response file similar to `jms_uninstall.properties` using the repository superuser user name credentials that contains the following:

```
INSTALLER_UI=silent
JMS_DOCBASES_ADMIN_USER_NAME=
<user name of the administrator who installed Independent Java Method Server>
JMS_DOCBASES_ADMIN_USER_PASSWORD=
<Password of the repository superuser>
```

- If HashiCorp Vault is enabled, then the repository administrator user password is retrieved from the HashiCorp Vault server.
- If HashiCorp Vault is not enabled, then type the repository administrator user password.
2. Run the following command format:

```
Uninstall.exe -f <path of jms_uninstall.properties>
```
3. Restart the repositories.

Chapter 3

Distributed Content

This documentation describes the Documentum CM Server features that support the distributed configurations. It contains the information to help you determine which features and configurations best meet the needs of your business. Also, it provides procedures to implement and manage those features and configurations.

The information is intended for system administrators or superusers who are responsible for the implementation and maintenance of a distributed environment.

3.1 Distributed Configuration components

This section describes the Documentum CM Server features that are the building blocks for the common distributed configurations.

3.1.1 Building blocks

This section describes the features that are the building blocks for implementing distributed configuration models.

3.1.2 Network locations

Network locations are a basic building block of a single-repository distributed environment for web-based clients. Network locations represent a place or area on network's topography. A network location can represent:

- A branch office
- A group working in the same geographical area
- Any set of users you select to aggregate as a network location

Network locations typically identify one or more IP addresses or address ranges. The addresses generally represent machines that are close to each other. For example, all machines in one branch office could be defined as one network location. Another network location could represent all users in the Western United States or in Eastern Europe. The administrator who configures the system determines the geographic size of a network location.

3.1.2.1 Benefits and best use

Network locations are useful only when configuring a single-repository distributed environment for web-based clients. Desktop-based clients do not use network locations.

Users connecting to a repository through a web-based client are automatically connected to the closest server for content requests by using network locations. Network locations enhance performance when the users access documents and any other object with content.

3.1.2.2 Configuration requirements

To use network locations, fulfill the following configuration requirements:

- Designate one repository as the global registry, and you must store the network location definitions in that repository.
- Specify a Documentum CM Server or Accelerated Content Services server proximity value for each network location.

A Documentum CM Server or Accelerated Content Services server proximity value defines the network location's proximity to those servers. This information is used to ensure that the server closest to the user manages content requests.

3.1.3 Accessing Accelerated Content Services, Branch Office Caching Services, and Messaging Service URLs in an IPv6 environment

To access the Accelerated Content Services, Branch Office Caching Services, and Messaging Service URLs in an IPv6 environment, you must provide the IPv6 address within the square brackets in the URL.

3.1.4 Accelerated Content Services servers

The Accelerated Content Services server is a lightweight server that handles read and write content operations for web-based client applications. There is one Accelerated Content Services server for each Documentum CM Server host installation. It communicates with one Documentum CM Server for each repository in the host installation. The Accelerated Content Services server uses HTTP or HTTPS for all communications.

The Accelerated Content Services server is installed automatically when the first repository on the installation is configured. If you add repositories to the installation, the Accelerated Content Services server configuration information is updated so that the server can communicate with the primary Documentum CM Server for the new repository.

If you install a remote site with a remote Content Server (RCS), then the installation at that site also has its own Accelerated Content Services server.



Note: Accelerated Content Services does not use nor require user authentication. Furthermore, if user authentication (OTDS SSO) is implemented in the OpenText Documentum CM system, then as long as a user successfully authenticates through their Web client, the appropriate Accelerated Content Services server seamlessly returns the requested content.

3.1.4.1 Benefits and best use

The Accelerated Content Services server serves users who are accessing the content through web-based client applications.



Note: You cannot use an Accelerated Content Services server to handle content read and write requests for users on desktop client applications.

3.1.4.2 Configuration requirements

Configure Accelerated Content Services servers as follows:

- At least one valid Accelerated Content Services configuration object for the Accelerated Content Services server in each repository served by that Accelerated Content Services server must exist.
- Configure the `acs.properties` file for the Accelerated Content Services correctly. The installation process automatically configures an initial `acs.properties` file.
- To enable asynchronous write operations for the server, set the `acs_write_mode` property in the content transfer configuration object in the repositories correctly.
- Make sure that the Accelerated Content Services server ports are open if the network locations that the server is servicing are outside a firewall.
- If the Documentum CM Server and application server clocks are out of sync by more than six hours (the default), URLs expire. When URLs expire, content transfer for uploading and downloading of content fails. Therefore, synchronize the Documentum CM Server and application server clocks with a time server. You set the time interval after which URLs expire in the `acs.properties` file on the Accelerated Content Services host.

3.1.4.3 Accelerated Content Services caching

You can configure the Accelerated Content Services server to perform content caching. To set up content caching, modify properties as mentioned.

- Windows:

```
%DM_JMS_HOME%\webapps\ACS\WEB-INF\classes\config\acs.properties
```

- Linux:

```
$DM_JMS_HOME/webapps/ACS/WEB-INF/classes/config/acs.properties
```

To set up Accelerated Content Services caching:

1. Set the `acs.cache.enabled` property to `true`. This property enables content caching. By default content caching is disabled.
2. Set the `cache.store.root` property to the location where you want to store the cached content, for example, `C:\\\\Documentum\\\\acsCache`. This property sets the root directory for the cache that holds the content.
3. Set the `cache.store.quota` property to `1000M`. This property determines the quota size of the cache.

3.1.5 Branch Office Caching Services servers

A Branch Office Caching Services server is a caching server. It is a separate, optional product with its own installer. It is not installed with Documentum CM Server.

Branch Office Caching Services servers cache content locally. When a Branch Office Caching Services server handles content requests from users, it caches the requested content locally. Branch Office Caching Services servers can pre-cache content through a pre-caching job or programmatically. Caching content allows users to obtain frequently accessed content quickly. You can configure the amount of content that is cached and the length of time for holding the content.

When users save content to the repository through a Branch Office Caching Services server, the underlying client application determines whether the write operation is asynchronous or synchronous. This application might default to one or the other or offer users a choice between synchronous or asynchronous write operation. If the content is written asynchronously, it is cached on a Branch Office Caching Services server until an internal job runs to save the content to the repository.

Branch Office Caching Services servers communicate with Accelerated Content Services and Messaging Service servers instead of directly with Documentum CM Servers. You can configure the repositories that a Branch Office Caching Services server serves.

Additionally, you can configure a Branch Office Caching Services server in either push or pull mode to obtain messages from a Messaging Service server. In push mode, a Branch Office Caching Services server accepts messages sent to it from a Messaging Service server. In pull mode, a Branch Office Caching Services server contacts the Messaging Service server and takes messages from the Messaging Service server's message queue. If there is a firewall between the Messaging Service server and the Branch Office Caching Services server, the Branch Office Caching Services server is typically configured in the pull mode. If there is no firewall between the two servers, the Branch Office Caching Services server is typically configured in push mode.

Branch Office Caching Services does not use or require user authentication. In addition, if user authentication (OTDS SSO) is implemented in the OpenText Documentum CM system, then as long as a user successfully authenticates through

their Web client, the appropriate Branch Office Caching Services server seamlessly returns the requested content.



Note: If your SSO implementation requires cookie authentication in order to connect to the Branch Office Caching Services host, then all hosts (Web client, UCF client/application server, Branch Office Caching Services, and Documentum CM Server/Accelerated Content Services) must use the same SSO server because the UCF client uses the Web client's authentication cookie, which is provided by an SSO server, when connecting to the Branch Office Caching Services host.

3.1.5.1 Benefits and best use

Use Branch Office Caching Services servers to provide web-based clients the fastest possible access to frequently used content without having to install a distributed storage area remote site.

3.1.5.2 Limitations

Access to the file systems on the Branch Office Caching Services server host can be less secure than access to content storage areas in the repository.

3.1.5.3 Branch Office Caching Services encryption

Branch Office Caching Services servers support content encryption. Content encryption enhances the security of confidential content by preventing the unauthorized access and viewing of content. You can select from the following encryption options:

- You can allow the Branch Office Caching Services server to decide whether to encrypt content.
- You can configure the Branch Office Caching Services server to encrypt content always (also known as “unconditional encryption”)
- You can disable content encryption for the Branch Office Caching Services server.

3.1.5.4 Partial download of content

Branch Office Caching Services servers support partial download of content. Branch Office Caching Services servers can serve content as it arrives without having to wait for the remainder of the content file to arrive. This practice improves performance because users can start viewing content sooner than waiting for the entire content file to download.

For more information, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

3.1.5.5 Configuration requirements

Configure a Branch Office Caching Services server as follows:

- A Branch Office Caching Services server configuration object representing the Branch Office Caching Services server must reside in the global registry.
- Configure an `acs.properties` file correctly for each Branch Office Caching Services server. The installation process configures an `acs.properties` file automatically.
- The ports on which Branch Office Caching Services server serves content to users must be open.
- If the Branch Office Caching Services server is configured in pull mode, the server must have access to the URL defined in the `message_consume_url` property of the Messaging Service configuration object.
- If the Branch Office Caching Services server is configured in push mode, the Messaging Service server must have HTTP access to the Branch Office Caching Services server.
- To use the pre-caching feature, enable pre-caching in the content transfer configuration object in each repository for which you want the feature enabled.



Note: The pre-caching feature is enabled by default.

3.1.6 Messaging Service servers

A Messaging Service server is a server that routes messages between Branch Office Caching Services and Documentum CM Server instances. A Messaging Service server routes the following messages:

- Content pre-caching
- Asynchronous write operations

For asynchronous write operations to work, 23.4 or later versions of Messaging Service requires 23.4 or later versions of Web Development Kit based applications (for example, Documentum Administrator) or OpenText Documentum Content Management (CM) Foundation SOAP API.

In other combinations of versions, the asynchronous write operations parks the content in Branch Office Caching Services server. The message from Documentum Administrator or Foundation SOAP API to Messaging Service server which is intended to move the parked content from Branch Office Caching Services server to the Accelerated Content Services server fails. This behavior is seen only for 23.4 and later versions. To resolve this issue and to make other combinations of versions to work, configure the `dm_AsyncronousWrite` job (set to Active) to periodically run so that the parked content is moved to the Accelerated Content Services server.

The scenario examples are as follows:

- Documentum CM Server 23.4 or later with Documentum Administrator/Foundation SOAP API 23.2 or earlier
- Documentum Administrator/Foundation SOAP API 23.4 or later with Documentum CM Server 23.2 or earlier



Note: In all other versions prior to 23.4, the asynchronous write operations work for all combinations.

You can deploy Messaging Service servers in a high-availability configuration.

3.1.6.1 Benefits and best use

The operations of a Messaging Service server are integrated into the distributed environment. The installation is simple and minimal configuration is required.

3.1.7 Pre-cached content

Pre-cached content is content that has been cached on a Branch Office Caching Services server before users request that content. You can cache the most up-to-date version of content that users frequently request before they request it again.

3.1.7.1 Benefits and best use

Pre-caching content is most useful for large content files that are frequently or regularly requested by users. It provides the fastest possible content delivery to users because the content is already cached as close as possible to the users.

3.1.7.2 Limitations

- You cannot pre-cache virtual documents or content that requires manipulation before viewing.
- Additionally, resource forks of Macintosh files are not pre-cached; only data forks are pre-cached.

3.1.7.3 Configuration requirements

Using content pre-caching has the following configuration requirements:

- Make sure that the Messaging Service server is correctly configured.
- Make sure that content pre-caching is enabled in the content transfer configuration object in the repository that contains content you want to pre-cache.

Content pre-caching is enabled through Documentum Administrator. For more information about content pre-caching, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

- Specify the content to pre-cache, either through a pre-caching job or programmatically through Foundation Java API.

3.1.8 Asynchronous write capabilities

Asynchronous write means to store content on a Branch Office Caching Services server only and then write that content to the repository later. Metadata is still written immediately to the repository. Even after content is written to the repository, the content remains in the Branch Office Caching Services server's content cache.



Note: In contrast to asynchronous write, synchronous write means to write content immediately to the repository. The underlying application can default to one type of write or allow the user to select one.

As long as the content resides on the Branch Office Caching Services server only, users who do not have access to the Branch Office Caching Services server cannot access the content for viewing or modification nor can they use administration methods, such as `MIGRATE_CONTENT`, to access the content.

Requests to write content to the repository are sent to the Branch Office Caching Services server where they are processed in first-in-first-out (FIFO) order. If the request is not executed immediately, then a user-defined job sends the request again. This job implements the `dm_AsyncronousWrite` method, which checks the relevant metadata on the Documentum CM Server to determine which content is parked on the Branch Office Caching Services server. For each piece of content that is found, a message requesting to write that content to the repository is sent to the Messaging Service server.

3.1.8.1 Benefits and best use

Asynchronous write operations ensure that a user does not wait for content to be saved to the repository when the network communication is slow. Additionally, other users in the network locations served by the Branch Office Caching Services server on which the content is stored have immediate access to the content.

Asynchronous write operations are best used in the following situations:

- The branch office and primary office have slow network connections.
- When users at the network locations served by the Branch Office Caching Services servers primarily use that content.
- The content is large.

3.1.8.2 Limitations

Using asynchronous write has the following limitation:

Content is unavailable to users who are not accessing the repository through the Branch Office Caching Services server on which the content is stored.

3.1.8.3 Configuration requirements

Using synchronous write has the following configuration requirement:

In the content transfer configuration object, configure the write mode for the Accelerated Content Services server to allow write operations.

Configure the Branch Office Caching Services server for asynchronous write as follows:

- Enable the Branch Office Caching Services server for asynchronous write.
- Enable asynchronous write in the content transfer configuration object in the global registry.
- Make sure that the Messaging Service server is configured appropriately.
- There must be an index on the `i_parked_state` and `r_object_id` properties of the `dmr_content` object in the database.
- Set the `dm_AsyncWrite` job to the active state.

This job is installed in the inactive state.

3.1.9 Distributed storage areas and configuration requirements

A distributed storage area is a single storage area made up of multiple component storage areas. They are the foundation of the single-repository distributed model for desktop clients. You can also use a distributed storage area in web-based models. All sites in a model using a distributed storage area share the same repository. However, each site has its own local storage area component to provide fast, local access to content.

The component storage areas can be file store or linked store storage areas. If you encrypt one component, encrypt all components. It is not possible to have a distributed storage area with some file-store components that are encrypted and some that are not encrypted.



Notes

- Linked store storage areas are not supported.
- A file store storage area may not use compression and de-duplication if that storage area will be a component of a distributed storage area. Distributed storage areas do not support compression and de-duplication.

You can either share content files or replicate them between component storage areas.

3.1.9.1 Benefits and best use

Using a distributed storage area solves the performance problems experienced by remote desktop users when all sites share one repository with centralized content storage. For example, if the repository and its content files are located in Nice, France, then users in Toronto, Canada, can experience delays in accessing files due to the distance between them and the repository. If you set up a distributed storage area, each site has fast access to content files in a local storage area. Users in Toronto no longer experience poor performance when opening documents.

For web-based users, distributed storage provides an alternate configuration model if the preferred model is not acceptable.

3.1.9.2 Limitations

After a repository begins using a distributed storage area, it is not possible to remove that storage area and return to a standalone, non-distributed configuration.

3.1.9.3 Configuration requirements

Configure a distributed storage area as follows:

- The host machines for the participating servers must run the same operating system.
- Your network must have high bandwidth and good reliability to support repository access.
- All the components of the distributed storage area and the containing distributed store object must have the same value in the `media_type` property. This property indicates whether the files stored in the storage area are thumbnail renditions, streaming content, or another kind of content.
- Install the index agent and index server only at the primary site.

Additional requirements depend on how you select to handle content files.

3.1.10 remote Content Servers

A remote Content Server (RCS) resides at each remote site that has a component of a distributed storage area. The remote Content Servers are automatically configured to provide maximum performance for desktop users for content-related queries. remote Content Servers do not handle metadata requests.

When a repository is configured with a distributed storage area, there is a primary site, and one or more remote sites. The RDBMS, which holds the repository's metadata, resides at the primary site. The Documentum CM Server at the primary site is configured to service all metadata requests, as well as content requests from clients local to that server. The remote Content Servers at the remote sites are

configured to handle only content requests. These servers provide content to users from their local storage area and write content to that storage area. An RCS does not handle metadata requests.

This model provides reasonably good performance for content requests because content is stored locally and accessed by a local server. It also provides good performance for metadata requests because the server closest to the RDBMS manages the requests.

3.1.10.1 Benefits and best use

The remote Content Servers provide increased performance for repository queries when a repository has a distributed storage area that is distributed across different geographical sites.

3.1.10.2 Limitations

- You cannot use an RCS as a failover server for the primary server.
- The Documentum CM Server at the primary site cannot fail over to an RCS.
- The primary Documentum CM Server and remote Content Servers must be of the same version.

3.1.10.3 Configuration requirements

Configure a distributed storage area and its remote Content Servers as follows:

- Proximity values for the Documentum CM Server at each site must identify one server as the metadata server and all others as remote Content Servers.

You can still modify the basic proximity values even though they are configured automatically when the sites are installed.

- Install the index agent and index server at the primary site.
- Set the metadata server's session timeout value to a minimum of 30 minutes.
- Configure servers at all sites to use the same authentication mechanism to authenticate all remote users and groups accessing distributed documents.
- The remote Content Servers and the data server must have compatible secure connection mode configurations. This configuration ensures that clients connecting from remote sites can access the remote Content Server and the metadata server. Alternatively, the secure connection default for the client must allow the client to request a connection on a native or secure port.

Configuring the servers to accept either secure or native connections is the easiest solution.

- It is recommended to provide unique host names in remote Content Server configuration when more than one host is used.

3.1.11 Shared content

Shared content files are files that are stored only in one component of a distributed storage area but are still accessible to users at all sites. The remote Content Servers fetch shared content files directly when needed. Documentum CM Servers can fetch the content files directly when a distributed-storage area's components are configured as shared drives.

3.1.11.1 Benefits and best use

Using shared content eliminates the need to enable surrogate get functionality or run ContentReplication jobs at each component site.

In a distributed storage area, the best documents for sharing are the ones local to one site that users from other sites do not access frequently.

3.1.11.2 Configuration requirements

Configure shared content files as follows:

- Configure each distributed-storage area's component to be a shared directory.
- Configure the installation owner (`dmadmin` or equivalent) at each site to be the same account at all sites.
- On Windows platforms:
 - All the host machines must belong to the same domain.
 - The Documentum CM Server installation owner must be a domain user and administrator in that domain.
 - Specify the UNC path for the location objects representing the shared storage areas.
 - The Documentum CM Server installation and the repository owners must have Full Control permissions for the storage locations.

3.1.12 Content replication

Content replication supports configurations with distributed storage areas. In a distributed storage area within a repository, a portion of the content files stored in each site are replicated to other sites. Because each site has a copy of the content files, servers accessing these files do not fetch them from a remote storage area.

Documentum CM Server provides automatic replication, through the ContentReplication tool, on-demand replication, using the surrogate get feature, or manual replication, using the `REPLICATE` or `IMPORT_REPLICA` administration methods.



Note: Content replication does not replicate content from one repository to another one.

3.1.12.1 Benefits and best use

Replicated content optimizes content transfer performance when users open a document because access is local. Consequently, the best candidates for replication are documents that are large or accessed frequently by users at all locations.

3.1.12.2 Configuration requirements

The ContentReplication tool and the surrogate get feature both require a homogenous server host environment. Therefore, the host machines for all of the participating servers must be all Windows or all Linux machines.

If you use the `REPLICATE` administration method to replicate content or the `IMPORT_REPLICA` method to copy content, configure the servers to connect with each other. On Linux platforms, the servers must be able to connect using NFS.

The secure connection mode setting of the target server must be compatible with the connection mode requested by the client performing the content replication.

3.1.13 Reference links

Reference links are a feature of a multi-repository configuration. If your deployment has multiple repositories, users are not limited to working only with objects in the repository into which they logged in. Users can also work with objects in other repositories. For example, a user might receive a task in their home repository inbox with an attached document that resides in a different repository. Or a user might view or update an object in a repository that is a replica of an object in a different repository.

A reference link in one repository points to an object in another repository. A Documentum CM Server creates reference links as needed when users work with objects in multiple repositories or when object replication occurs. The following operations create reference links:

- Linking a remote object to a local folder or cabinet
- Checking out a remote object
- Adding a remote object to a local virtual document
- Object replication jobs

3.1.13.1 Benefits and best use

Reference links simplify work sessions for users. Users can start one repository session and manipulate objects in any repository without starting a new session each time they want to access an object in another repository. Reference links can also provide users with local access to the properties of a remote object.

3.1.13.2 Configuration requirements

To enable the reference links feature:

- Users must have accounts in each repository they access.
- Each participating repository must project to the connection brokers of the other participating repositories.
- Configure all servers to listen on a secure and a native (unsecured) port. For information, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

To facilitate user access, you can configure both or one of the following:

- Add all of the repositories into the same federation
- Use an LDAP directory server to manage users and groups (see *OpenText Directory Services* documentation)

A federation is a OpenText Documentum CM feature that facilitates management of global users and groups and external ACLs across repositories. Any addition or deletion of a global user or group or modification of a global user or group property is made in the governing repository. The governing repository then propagates the changes to the member repositories. External ACLs are automatically replicated to member repositories as needed so that security for the global users and groups is uniform within the federation.

An LDAP directory server is a third-party product that provides a single place for maintenance of some or all users and groups in your enterprise. User and group entries are created in the directory server and those entries are propagated to all repositories that are set up to use the directory server. The property information that is propagated is defined when you set up the repository to use the LDAP directory server. This property information is not limited to the global properties of the users and groups.

Unlike a federation, the LDAP directory server does not replicate external ACLs to participating repositories. If you use a directory server without a federation, then manage ACL replication manually.

If you use a federation and an LDAP directory server, then the directory server communicates with the governing repository in the federation. It also communicates with and any other nonfederated repositories with which you want to use the directory server. The governing repository propagates the user and group changes it receives from the LDAP directory server to the member repositories. It also manages the external ACLs within the federation.

Users who are managed by an LDAP directory server do not require an operating system account. However, Documentum CM Server requires an operating system login name to create the user. This login must be unique within the system. For LDAP-managed users, you can define operating system names for them without actually creating the operating system accounts. For more information about configuring a repository to use an LDAP directory server, see *OpenText Documentum Content Management - Administrator User Guide* (EDCAC250400-UGD).

3.1.14 Object replication

Object replication supports multi-repository, distributed configurations. These configurations have multiple sites with a separate repository and relational database (RDBMS) at each site. Object replication replicates entire objects (property data and content) between repositories.

In the target repository, replicated objects are stored as replica objects with associated `dm_reference` objects. Each replica object has an associated reference object. With only a few constraints, users can manipulate replica objects much as they do source objects. Users can review, annotate, or index replicas. They can query against them or add them to a virtual document. They can also manipulate the source object through the replica.

Jobs automate replication and they are defined by the business requirements of the enterprise.

3.1.14.1 Benefits and best use

Replication can reduce network traffic because users access local replicas of the objects for most operations. Replication is of most benefit during peak usage times.

Use object replication if you want local autonomy; for example, if your wide area network (WAN) is not always reliable. If objects are replicated into a target repository, then users can continue to work even if the replica object's source repository is not available.

3.1.14.2 Configuration requirements

You enable object replication as follows:

- All participating sites must project to the connection brokers at all other participating sites.



Note: Cross-projection between all sites is only required if you want:

- To allow users to manipulate source objects through the replicas
- To allow the server at the target repositories to perform automatic replica refreshes

If your replicas are read-only, then cross-projection is not necessary. For example, if you are replicating from a source repository inside a firewall

to a target repository outside a firewall, then the target server does not need to project to the source repository's connection broker nor the source to the target.

- Because object replication uses the dump and load operations, the databases of the source and target repositories must either use the same code page, or the target database must use unicode. For example, if the source database is using Japanese and the target is using unicode, the replication operation succeeds. However, if the source is using Unicode and the target is Japanese, replication fails.
- The `secure_connect_mode` defined for the server in the target repository must be compatible with the secure connection mode requested by the content replication job that performs the object replication.

The server's secure connect mode is defined in the server configuration object. The mode that the job requests is defined in the `dfc.session.secure_connect_default` key in the `dfc.properties` file used by the job. That file is found on the host machine where the job resides. The mode defaults to `try_native_first`.

For more information about setting `secure_connect_mode` for a server or a client application, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

- If you are replicating documents created on Macintosh client machines, all participating sites must use the same Macintosh access protocol.
- Both the source and the target sites for each replication job must have enough disk space to accommodate the temporary dump files created by the job.
- Each target site must have enough disk space to accommodate the replicated content files.
- Network latency affects replication performance. For adequate performance, a ping between any two participating sites should be 200 milliseconds or less.

3.1.15 Federations

A federation is a set of two or more repositories that are bound together to facilitate the management of a multi-repository distributed configuration. One repository in the set is the governing repository. The remaining repositories are member repositories.

One enterprise can have multiple federations, but each repository can belong to only one federation.

A federation can include repositories with both trusted and non-trusted Documentum CM Servers.

3.1.15.1 Benefits and best use

In a multi-repository distributed configuration, users are typically working with objects from more than one repository in the same session. The objects can be the original objects, mirror objects, or replica objects. To maintain data consistency, object type and format definitions must be the same across all of the repositories. Similarly, to maintain consistent security, the definitions of users, groups, and ACLs must be the same across all of the repositories.

If data and security consistency is maintained manually in each repository, then keeping objects synchronized in multiple repositories can be time consuming and error prone. A repository federation automates much of the management of data and security consistency. The governing repository automatically propagates changes you make to users, groups, and external ACLs to each repository in the federation.

Federations are best used in multi-repository production environments where users share objects among the repositories. It is not recommended to create a federation that includes a mixed environment of production, development, and test repositories because development and test repositories can be different versions than the production ones, contain object type and format definitions that change frequently, and have a small subset of all of your users.

3.1.15.2 Configuration requirements

For the most consistent behavior in a repository federation or any multi-repository, distributed configuration, follow these configuration requirements:

- Object type and format definitions must be the same across all participating repositories.

Because Documentum Administrator does not propagate type or format changes in a governing repository to member repositories, perform this operation manually or by using your own applications.

- Users and groups must be the same across all participating repositories.

Documentum Administrator is used to manage all global users and groups in a repository federation. At setup, define whether users and groups are global or local. Making them all global is recommended.

The federation update jobs automatically propagate all changes to global users defined by `dm_user` objects. If you define users by subtypes of `dm_user`, they are not automatically propagated unless you identify those subtypes in the federation's properties. You can do that when you create the federation or after, as a federation modification.

Changes to global users and groups are only allowed using the Documentum Administrator and are propagated automatically to all member repositories.

- The Documentum CM Server at the governing site must project to the connection brokers at the member sites.

- The Documentum CM Servers at the member sites must project to the connection broker at the governing site.
- Make sure that either of the requirements are met:
 - The servers are configured to listen on both a secure and a native port.
 - The secure connection default for clients allows the clients to request a connection on a native or secure port.

Configuring the servers to accept either secure or native connections is the easiest solution. For more information about setting the Documentum CM Server connection mode, see *OpenText Documentum Content Management - Administrator User Guide* (EDCAC250400-UGD).



Note: When you set up a federation and use an LDAP server for accessing users, the governing repository synchronizes users and populates the user_login_domain with the name of the LDAP configuration object. For example if an LDAP configuration object is called MS_LDAP, this value is populated on the user domain. When users are then synchronized to members as part of the federation update, those same user attributes are populated. However, if on the member, the LDAP configuration name is different, authentication fails. Therefore, if you are using an LDAP configuration object to synchronize users on the governing repository, each member must have an LDAP configuration object with the same name.

3.2 Distributed Configuration models

This section provides information on configuration settings that affect Foundation CMIS API, including JVM, Linux, and application properties settings.

3.2.1 Overview of models

You can set up a distributed configuration for either a single repository or multiple repositories. In a single-repository distributed configuration, content is distributed across all sites configured to access one repository. In a multi-repository distributed configuration, objects (content plus metadata) are distributed across all participating repositories.

The most common distributed configurations are as follows:

- For single-repository distributed environments:
 - A single repository with content stored at the primary site and accessed from remote sites using the Accelerated Content Services server. Optionally, you can also use the Branch Office Caching Services servers.
 - A single repository with content stored in a distributed storage area and accessed from remote sites using remote Content Servers, Accelerated Content Services servers, and optionally, Branch Office Caching Services servers.

[“Single-repository distributed models” on page 195](#) contains more information.

- For multi-repository distributed configurations:
 - Multiple repositories that replicate objects among themselves
 - Multiple repositories organized as a federation

[“Multi-repository distributed models” on page 216](#) contains more information.

An enterprise can use one model or a combination of the models. For example, you might have one repository that contains primarily material, such as SOPs, that users only read. You might configure that repository to use Branch Office Caching Services servers for its remote users. Another repository might store business and sales proposals that are written and updated frequently. That repository uses distributed storage areas with Accelerated Content Services servers or desktop clients for remote users. You might also tie together the two repositories in a federation, to make sure that the users and groups in both repositories are the same.

3.2.2 Single-repository distributed models

In a distributed single-repository, the distributed data is content. Users in many locations want fast access to the documents, that is, content, in the repository. OpenText Documentum CM supports distributed content models for both web-based and desktop access.



Note: The figures in this section depict geographic locations, not individual machines.

3.2.2.1 Single model 1: Single repository with content persistently stored at primary site and accessed using Accelerated Content Services or Branch Office Caching Services servers

In this model, remote users connect through a web browser. The content is stored at the primary site and content operations are handled through either an Accelerated Content Services or Branch Office Caching Services server. The Accelerated Content Services server is a Documentum CM Server dedicated to handling content. It does not process metadata. It only handles read and write content requests. The Branch Office Caching Services server is a separate product. It is a caching server that communicates only with Accelerated Content Services servers. Like the Accelerated Content Services server, it does not handle metadata requests. Both the Accelerated Content Services and Branch Office Caching Services servers use HTTP or HTTPS protocol to process client content requests. Single model 1 is the preferred model when remote users are accessing repository content through a web-based application.

Two alternative configurations for this model exist. The configurations differ in how users at remote sites access the content at the primary site. In the first alternative, remote sites have a Branch Office Caching Services server installed and clients at each remote site use that Branch Office Caching Services server to access content. In

the second alternative configuration, users at remote sites have only web clients, and they access content using the Accelerated Content Services server at the primary site.

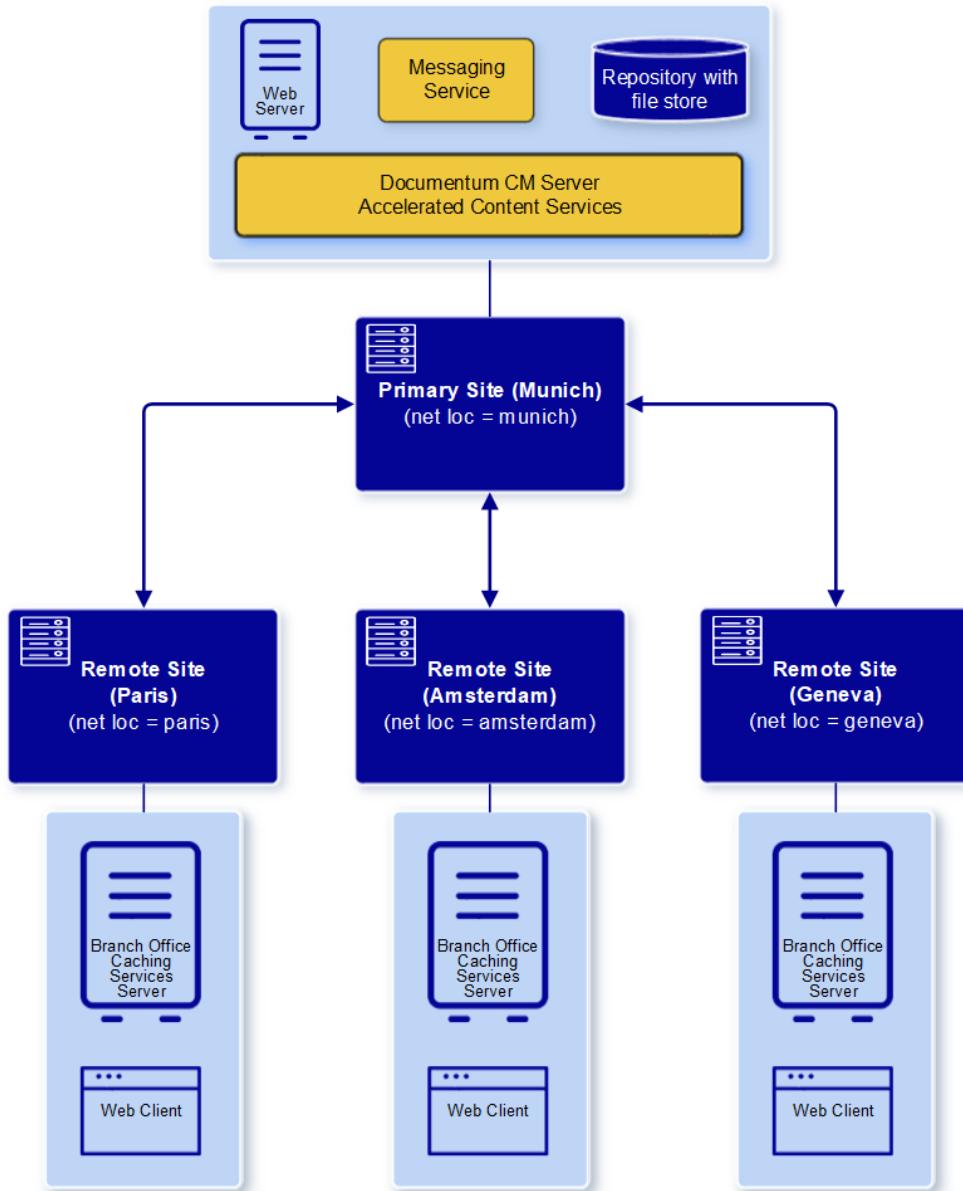


Figure 3-1: Alternative 1: Branch Office Caching Services servers at remote sites communicating with primary site

In this example of alternative 1, users at each remote site use a Branch Office Caching Services server to handle content requests. The Branch Office Caching Services server in the Paris branch office manages requests from Paris users. The

Branch Office Caching Services server in the Amsterdam branch office manages content operations for users in the Amsterdam branch office. The Branch Office Caching Services server in the Geneva branch office manages requests from Geneva users.

The Branch Office Caching Services server is a caching server. It maintains a cache of content files requested by users. You can also pre-cache content on a Branch Office Caching Services server. For example, you can cache content that users access frequently or regularly on the server before users request that content. You can perform pre-caching by a job or programmatically.

When the Branch Office Caching Services server receives a request for content, it first checks the cache that it maintains. If the content is in the cache, the Branch Office Caching Services server provides that content to the user. If the content is not in the cache, the Branch Office Caching Services server communicates with the Accelerated Content Services server at the primary site to locate the content. The Accelerated Content Services server, in turn, communicates with the web server and Documentum CM Server at the primary site.

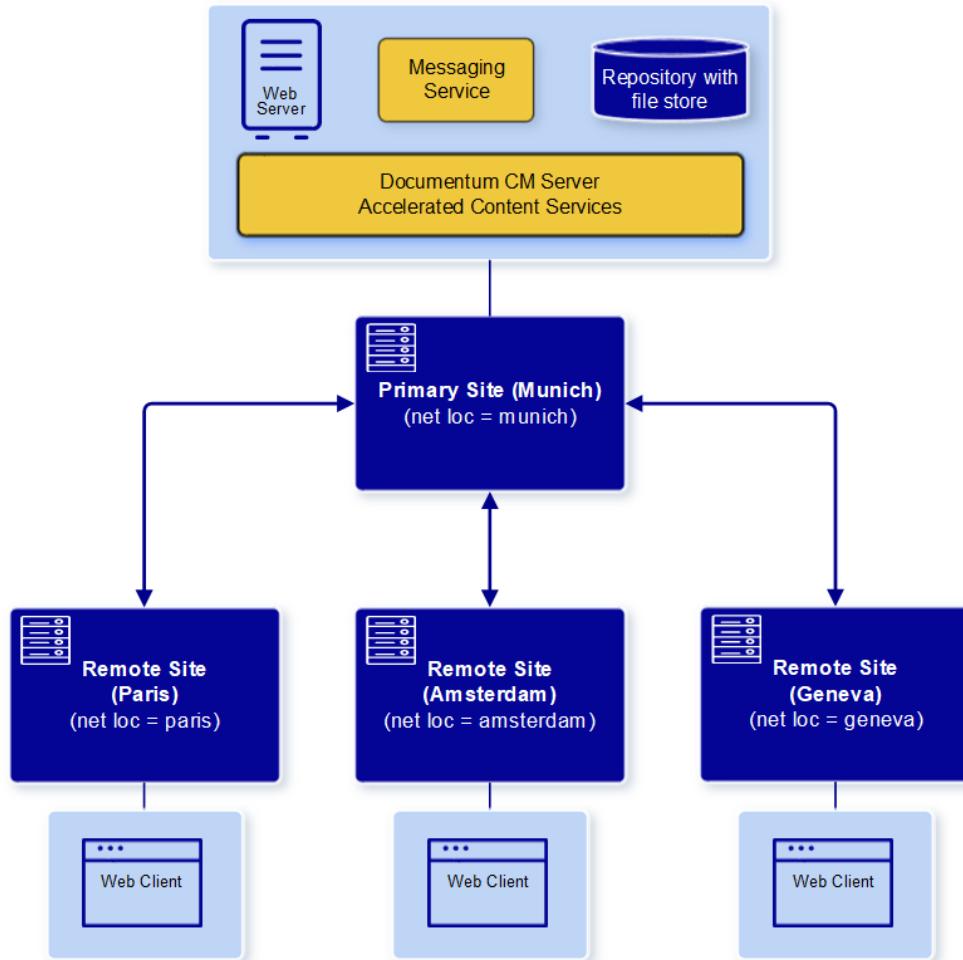


Figure 3-2: Alternative 2: Remote sites, without Branch Office Caching Services servers, using primary site's Accelerated Content Services server

In the second alternative, the Accelerated Content Services server at the primary site manages requests content operations from users at a remote site.

The illustration describes variation of the configurations that combines the two. In this example, the remote clients are telecommuters, working from their web browsers on home machines. When they request documents, a Branch Office Caching Services server installed at the branch office closest to their location manages the request. Documentum CM Server at the primary site manages all metadata requests.

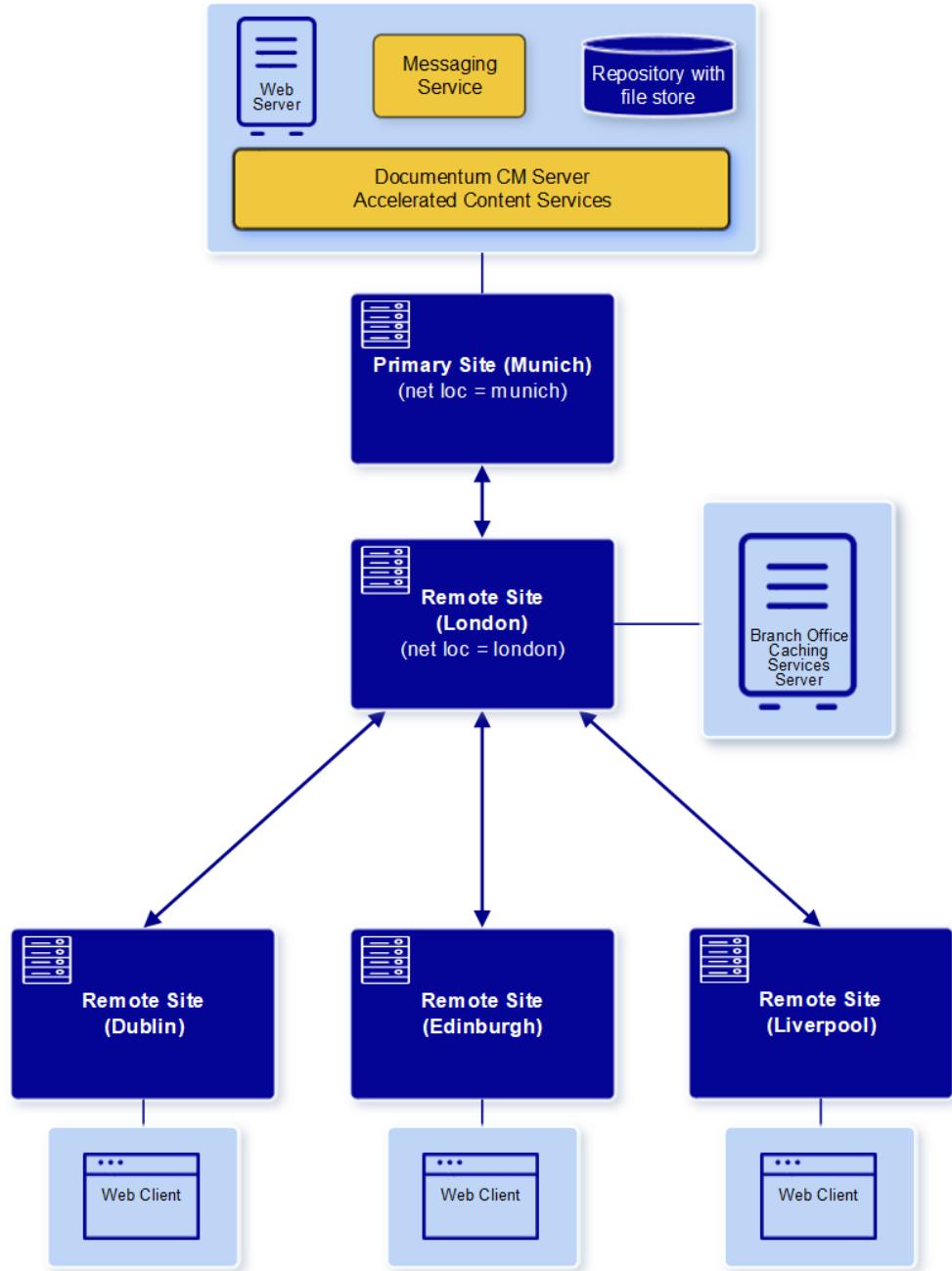


Figure 3-3: The two alternatives for single model 1 combined

In each of the configuration alternatives, each Accelerated Content Services or Branch Office Caching Services server is defined as a network location. When users begin a web-based session, the client application determines which network locations are available to the user. Users are either automatically assigned to a

network location or they can select from available locations, depending on how the client applications are configured. The network location with which the session is associated determines which Accelerated Content Services or Branch Office Caching Services server handles each user's content requests.

3.2.2.1.1 Benefits and best use

This model requires the least amount of administrative overhead. It is easy to set up and configure, and ongoing administration needs are minimal. There are no content replication jobs to define, administer, and run.

This model is not available for users on desktop clients.

3.2.2.2 Single model 2: Single repository with content in a distributed storage area

In this model, content is stored in a distributed storage area. A distributed storage area is a storage area with multiple component storage areas. One component is located in the repository's primary site. Each remote site has one of the remaining components. Each site has a full Documentum CM Server installation: a remote Content Server and an Accelerated Content Services server for the repository. Content is replicated from its source component to the remaining components by user-defined content replication jobs.

You can use this model for either web-based clients or desktop clients.

If a remote site is using web-based clients, the site must configure the use of an Accelerated Content Services server to manage content. In addition to configuring an Accelerated Content Services server, you can also configure a Branch Office Caching Services server. Also install a Messaging Service server to facilitate pre-caching for Branch Office Caching Services servers and asynchronous write operations for remote users. Desktop clients at remote sites use the Documentum CM Server at the remote site to access content. In this configuration, Documentum CM Server at the primary site manages metadata requests, and an RCS or Accelerated Content Services at the remote sites manage content operations.

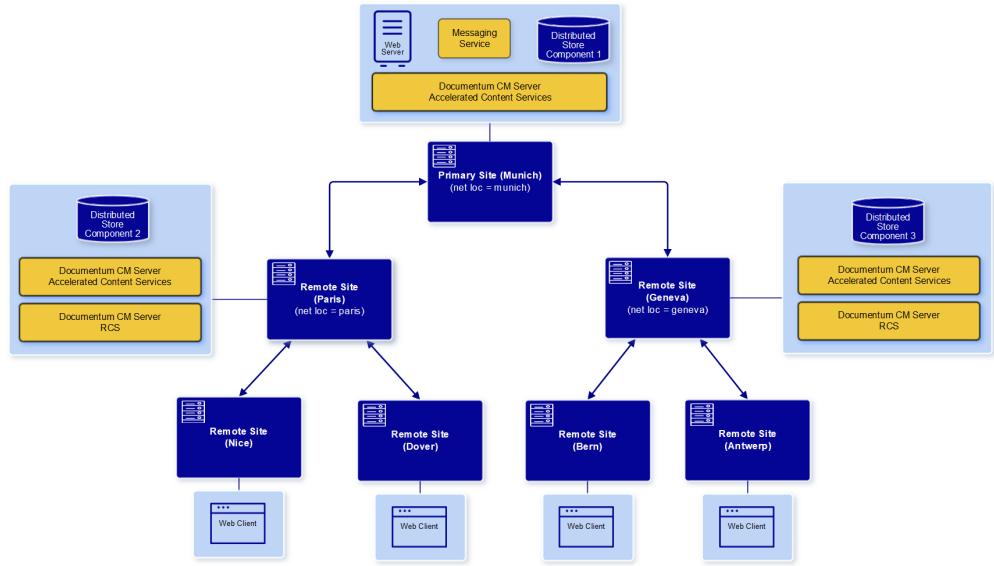


Figure 3-4: Single Model 2: Single repository with a distributed storage area

In this model, users in the branch offices in Nice and Dover access content stored in distributed storage at the larger Paris branch office. Users in the branch offices at Bern or Antwerp access content stored in the Geneva branch office. If users are logging in using a web-based client, Accelerated Content Services manages content requests at the appropriate branch office in Paris or Geneva. If users are logging in using a desktop-based client, Documentum CM Server in Paris or Geneva manages content requests. In this model, the remote sites using web-based clients could use Branch Office Caching Services servers for managing content requests, instead of web browsers.

3.2.2.2.1 Benefits and best use

For sites using web-based clients, this model is best if model 1 is not acceptable for any particular reason. For sites using desktop clients, this model is the only model available for a single-repository, distributed configuration.

3.2.3 Building block architectures for single-repository models

The distributed models for web-based clients use either an Accelerated Content Services server or a Branch Office Caching Services server (or both) to distribute content to users at remote sites. These models use the network location, Accelerated Content Services server, Branch Office Caching Services server, and proximity values building blocks.

The distributed model for desktop-based clients uses the distributed storage area, shared or replicated content, and proximity values building blocks.

This section describes how the various building blocks work to support the distributed model for a single repository. It also includes a description of the internal communications in the web-based model.

3.2.3.1 Network locations

Network locations are recorded in the repository designated as the global registry. They are stored as `dm_network_location_map` objects. The properties in the object record the following information:

- The name of the network location
- The IP addresses or address ranges are assigned to that location

Use network locations to determine an end user's proximity to a Documentum CM Server or Accelerated Content Services server. Define network locations for remote sites with Branch Office Caching Services servers or Accelerated Content Services servers. The definitions are not required if remote sites are using the Accelerated Content Services at the primary site and there are no servers installed at the remote site.

3.2.3.2 Accelerated Content Services server

Accelerated Content Services server handles content requests from web-based clients. There is one Accelerated Content Services server for each OpenText Documentum CM installation on a server host machine. The Accelerated Content Services server runs in the Java Method Server.

When you configure the first repository in an installation, the procedure installs a Documentum CM Server for the repository and an Accelerated Content Services server for the installation. If it does not exist and deploys the `ACS.war` and `documentum-bocs-ws.war` files, the procedure also configures the Java Method Server service.

If you later configure additional repositories in that installation, the procedure automatically updates the Accelerated Content Services configuration information for the existing Accelerated Content Services server. Adding additional repositories does not add additional Accelerated Content Services servers.

Two internal components enable you to configure an Accelerated Content Services server: the `acs.properties` file and Accelerated Content Services configuration objects. Each Accelerated Content Services server has one `acs.properties` file and at least one Accelerated Content Services configuration object. The file is created when the Accelerated Content Services server is installed. The Accelerated Content Services configuration object is created when a repository is configured.

The `acs.properties` file identifies the repositories with which the Accelerated Content Services server communicates and is located in the following directories:

- Windows:

`%DM_JMS_HOME%\webapps\ACS\WEB-INF\classes\config\acs.properties`

- Linux:

`$DM_JMS_HOME/webapps/ACS/WEB-INF/classes/config/acs.properties`

The Accelerated Content Services configuration object resides in a repository. Each repository served by an Accelerated Content Services server must have an Accelerated Content Services configuration object for that Accelerated Content Services server.

The properties of the Accelerated Content Services configuration object configure various behaviors for the server. They can record the proximity values that define the Accelerated Content Services server's distance from each of the locations. The Accelerated Content Services configuration object also has a property that identifies with which Documentum CM Server the Accelerated Content Services server communicates in the repository.



Note: If there are multiple servers installed on a single host machine for a single repository, the Accelerated Content Services server communicates only with one of the servers. You cannot manually configure Accelerated Content Services configuration objects for additional servers on a host machine.

The Accelerated Content Services configuration object can also identify which storage areas the server can access. These storage areas must be either standalone file store storage areas or file stores that are components of a distributed storage area. An Accelerated Content Services server cannot access content in other types of storage areas. Nor can it directly access encrypted or compressed content or virtual documents or XML documents. If an Accelerated Content Services server receives a request for a document that it cannot access, the request is forwarded to its associated Documentum CM Server. That Documentum CM Server services the request and sends the result back to the Accelerated Content Services.

3.2.3.3 Branch Office Caching Services server

The Branch Office Caching Services server configuration is recorded in an `acs.properties` file and a Branch Office Caching Services configuration object.

The `acs.properties` file for a Branch Office Caching Services server configures the location of the server's content cache. It also defines how long the server holds the content and other behavioral characteristics of the server. In a Branch Office Caching Services installation on a client host, the file is placed in a location dependent on the host operating system as follows:

- Windows:

`%DOCUMENTUM%\tomcat\webapps\bocs\WEB-INF\classes\config\acs.properties`

- Linux:

`$DOCUMENTUM/tomcat/webapps/bocs/WEB-INF/classes/config/acs.properties`

The Branch Office Caching Services configuration object identifies the network locations that the Branch Office Caching Services server serves and allows you to configure the following characteristics:

- Whether to run the Branch Office Caching Services server in push or pull mode
- The repositories that the Branch Office Caching Services server can service
- Whether the Branch Office Caching Services server can perform asynchronous write operations

Store the Branch Office Caching Services configuration object in the global registry repository and modify it by using Documentum Administrator.

3.2.3.1 Push and pull modes

A Branch Office Caching Services server can communicate with a Messaging Service server in either push or pull mode. This configuration is set in the Branch Office Caching Services configuration object. If the Branch Office Caching Services server is configured in push mode, the Messaging Service server sends messages routed to the server through Messaging Service to the Branch Office Caching Services server. In such cases, the Messaging Service server must have HTTP access to the Branch Office Caching Services server.

If the Branch Office Caching Services server is configured in pull mode, the Branch Office Caching Services server picks up messages routed to the server through Messaging Service. The Messaging Service server does not send them to the Branch Office Caching Services server. In such cases, configure the URL that the Branch Office Caching Services server uses to contact the Messaging Service server in the `acs.properties` file associated with the Branch Office Caching Services server. Additionally, the Branch Office Caching Services server must identify itself to the Messaging Service server by using the PKI credentials installed with the Branch Office Caching Services server. You can configure PKI credentials in the Branch Office Caching Services configuration object.

You can change the push or pull setting for a Branch Office Caching Services server. However, if you change from pull mode to push mode, the messages pushed to the Branch Office Caching Services server might not arrive if a firewall exists between the Messaging Service server and the Branch Office Caching Services server. Additionally, queued messages in the Messaging Service server are marked as either push or pull messages. Consequently, if you change from pull mode to push mode, the Branch Office Caching Services server does not receive any queued messages that are marked as pull messages. Pull mode, rather than push mode, is typically used for Branch Office Caching Services servers outside of a firewall.

3.2.3.3.2 Repository inclusion or exclusion

You can configure which repositories a Branch Office Caching Services server can service by defining either a repository inclusion list or a repository exclusion list. Two underlying properties record this configuration: `is_inclusion_list` and `docbase_names`. If you select to specify the repositories that the Branch Office Caching Services server can service, then the `is_inclusion_list` property is set to T. The repositories listed in `docbase_names` are the repositories that the Branch Office Caching Services server can service. If you select to specify the repositories that the Branch Office Caching Services server cannot service, `is_inclusion_list` is set to F. The names in `docbase_names` are the repositories that the Branch Office Caching Services server cannot service. However, it is recommended that you do not exclude any repositories from servicing by a Branch Office Caching Services server.

3.2.3.3.3 Asynchronous write configuration

You can enable a Branch Office Caching Services server for asynchronous write operations. If enabled and if a user or application selects to use an asynchronous write, the content file is written to the repository storage area at a later time. Whether a content file is stored on a Branch Office Caching Services server is recorded in the `i_parked_state` property of the associated content object. Applications can use the `IDfContentAvailability.getContentAvailability` method to determine whether a content file is available in the storage area. There is also a method available, `isContentTransferCapabilityEnabled`, that indicates what content transfer capabilities are enabled in the repository.

3.2.3.4 Messaging Service server

The Messaging Service server configuration is recorded in a Messaging Service configuration object and a `dms.properties` file. The Messaging Service configuration object is created manually after installing Messaging Service. The properties in the Messaging Service configuration object enable or disable the Messaging Service server and define the URLs to be used to communicate with the Messaging Service server. Create the Messaging Service configuration object in the global registry repository. Create and modify Messaging Service configuration objects only through Documentum Administrator.

The `dms.properties` file is created when Messaging Service is installed. Most of the keys in the file are set during installation and are not modifiable. Modify those keys that are modifiable through Documentum Administrator.

3.2.3.5 Content pre-caching

You can pre-cache content on a Branch Office Caching Services server either programmatically or through the execution of a content pre-caching job.

To cache content programmatically, use the `IDfPrecachingOperation` package interface in Foundation Java API.

To use a job, create a pre-caching job using Documentum Administrator.

3.2.3.6 Asynchronous write

Enable asynchronous write operations in the Branch Office Caching Services server, in the Branch Office Caching Services configuration object, and in the `dm_cont_transfer_config` object. Use Documentum Administrator to enable the feature. Both the Branch Office Caching Services configuration object and the content transfer configuration object are stored in the global registry.

When an application or user requests an asynchronous write operation, the content file is stored, or parked, on the closest Branch Office Caching Services server. It is then uploaded to the repository. If it is not uploaded immediately, it is written to the repository at a later time, after an internal job, which executes the `dm_AsyncronousWrite` method, executes. That method sends the write request to Messaging Service server again, and continues sending the message at scheduled intervals, until the content is written to the repository.

While the content is parked on the Branch Office Caching Services server, the `i_parked_state` property of its content object is set to TRUE. The content is available for reading or modification by any user who can access the Branch Office Caching Services server on whose host the content is parked. Other users do not have access to the content.

Asynchronous write operations affect content only. The metadata associated with the object is written to the repository as soon as the user or application requests a write operation, synchronous or asynchronous.



Note: The `dm_AsyncronousWrite` job requires an index on the `i_parked_state` and `r_object_id` properties of the `dmr_content` object in the database. This index is created automatically for the new repositories.

The `dm_AsyncronousWrite` job is installed in the inactive state.

3.2.3.7 Communication flow descriptions

This section describes the flow of communications in a web-based environment in the following situations:

- When users request a document for viewing
- When a synchronous or asynchronous write occurs
- When a content pre-caching request occurs

3.2.3.7.1 Communication flow when a remote user requests a document for viewing

The following steps describe the basic communications that occur when an end user using a web browser requests a document for viewing:

1. The web application hosted on a web application server receives a request.
2. The application sends a Getfile request to the Documentum CM Server through the UCF facilities of the Foundation Java API on the web application server host.
3. Documentum CM Server sends back a list of the candidate content files.

For each candidate file, the list describes the following information:

- The file's location
- How far the file is from the user requesting the file
- Instructions on building the URL to access the file

Documentum CM Server digitally signs these instructions by using the private key stored in the dm_cryptographic_key object in the repository. The signature is used to ensure integrity and to enable authentication of the source of the instructions.

4. When the Foundation Java API receives the list of candidate content files from Documentum CM Server, it contacts a connection broker to determine the Accelerated Content Services servers referenced in the list that are up and running. To obtain this information, it requests a repository map from the connection broker.
5. Using the information returned by the connection broker, the Foundation Java API constructs a URL to each candidate content file that is accessible by a running Accelerated Content Services. The Foundation Java API sends the list of URLs to the UCF client on the user's host machine, sorted according to their distance from the user's network location.
6. When the UCF client receives the list of URLs, it uses the first URL to request the content from the chosen Accelerated Content Services or Branch Office Caching Services server.

Depending on the configuration, the Accelerated Content Services server can be on a remote host machine or on the primary site's host machine.

If the UCF client receives an error using the first URL, it uses the next URL in the list it received from Foundation Java API. The UCF client reports any failed

attempts to Foundation Java API, and Foundation Java API will not include the failed URLs in future requests. If all candidate URLs fail to return the content, the request fails. The client application must repeat the request; since all Accelerated Content Services and Branch Office Caching Services options failed, Foundation Java API serves the content file through the application server.

7. If the server providing the content is an Accelerated Content Services server, it validates the signature on the request, using the public key provided to the server. Then, it returns the content file to the UCF client. (There is one public key per repository. It is stored in the dm_public_key object.)

If the Accelerated Content Services is on a remote site and it determines that the storage areas it can access do not have the file, it sends a request to its associated Documentum CM Server. Documentum CM Server then uses surrogate get to obtain the file.

8. If the server is a Branch Office Caching Services server, it also validates the signature on the request, using the public key provided to the server. The server then searches its cache for the content. If the content is found, the Branch Office Caching Services server returns the content to the UCF client. If the content is not found, the Branch Office Caching Services server sends the request to the closest Accelerated Content Services server that has the content. The Accelerated Content Services server then returns the content to the Branch Office Caching Services server, which in turn, sends the content to the UCF client.



Note: If there are no remote Accelerated Content Services servers close to the Branch Office Caching Services server, the Branch Office Caching Services server requests the content from the Accelerated Content Services server at the primary site. To ensure faster performance, schedule frequent replication jobs to replicate content to the remote Accelerated Content Services server sites.

9. The UCF client writes the content to the user's local disk.



Note: When viewing and writing, UCF is installed in the client machine and you may encounter any checksum validation errors in the Web Development Kit native plug-in logs.

To disable the checksum validation in the Web Development Kit native application (for example, Webtop) where multiple Web Development Kit native application servers are used, perform the following step in server:

Add the `<disableVerifyChecksum value="true"/>` parameter in the platform `os="all"` and `platform os="windows"` sections in `ucf.installer.config.xml` located at `Tomcat\webapps\<client>\wdk\contentXfer\` or any similar location in your application for all operating systems.

For example in Webtop with `platform os="all"`:

```
<platform os="all" arch="all">
<runtime type="all" version="any">
...
...
```

```
<defaults>
<ucfHome value="$java{user.home}/Documentum/.ucf" />
<ucfInstallshome value="$java{user.home}/Documentum/.ucf" />
<disableVerifyChecksum value="true"/>
```

For example in Webtop with platform os="windows":

```
<platform os="windows" arch="all">
...
<runtime type="java"
...
<defaults>
<ucfHome value="$java{user.home}/Documentum/ucf" />
<ucfInstallshome value="$java{user.home}/Documentum/ucf" />
<disableVerifyChecksum value="true"/>
```



Caution

The <disableVerifyChecksum value="true"> parameter must be used ONLY when using multiple Web Development Kit native applications (for example, Webtop) servers.

3.2.3.7.2 Communication flow when a remote user updates a content file

The following steps describe the basic communications that occur when an end user at a remote site updates a content file. The steps differ depending on whether a Branch Office Caching Services server is configured for the user's network location. It also depends on whether a Branch Office Caching Services server is configured for synchronous or asynchronous writing.

Asynchronous write with Branch Office Caching Services server

1. A user at a remote site uses a Web application to perform an operation that includes a new or updated content file.
2. The Web application uses the Foundation Java API on the web application server host to write the metadata associated with the new or update content file to the repository at the central location.
3. The Web application determines the network location of the remote user and identifies the Branch Office Caching Services server for that network location. Using information from the Branch Office Caching Services configuration object for that Branch Office Caching Services server, the application generates a URL for writing the content file to the Branch Office Caching Services server cache. The application then passes the URL to the UCF client application running on the user's browser machine.
4. The UCF client application writes the content file to the cache of the local Branch Office Caching Services server, using the generated URL.
5. The web application adds an asynchronous write request to the queue for the Messaging Service server.

From the original user's point of view, the operation is complete. The web application returns control of the browser to the user. The content file is available to all users in the same network location. However, the file is not

available to users at other network locations because other Branch Office Caching Services servers serve these network locations. Users at other locations can view and edit the metadata, but not the content file.

6. If the Branch Office Caching Services server is configured for push mode, the Messaging Service server retrieves the asynchronous write request from its queue. It then passes the write request to the Branch Office Caching Services server. If the Branch Office Caching Services server is configured for pull mode, the Branch Office Caching Services server contacts the Messaging Service server and retrieves the write request.
7. The Branch Office Caching Services server copies the content to the Accelerated Content Services server. The Accelerated Content Services server writes a copy of the content file into a file store at the central location, without removing the copy in the Branch Office Caching Services cache. The content is now available to all users.

Synchronous write with Branch Office Caching Services server

1. A user at a remote site uses a web application to perform an operation that includes a new or updated content file.
2. The web application uses the Foundation Java API on the web application server host to write the metadata associated with the new or update content file to the repository at the central location.
3. The web application determines the network location of the remote user and identifies the Branch Office Caching Services server for that network location. Using information from the Branch Office Caching Services configuration object for that Branch Office Caching Services server, the application generates a URL for writing the content file to the Branch Office Caching Services server cache. It then passes the URL to the UCF client application running on the user's browser machine.
4. The UCF client application writes the content file to the cache of the local Branch Office Caching Services server, using the generated URL.
5. The web application posts a write request to the Branch Office Caching Services server.
6. The Branch Office Caching Services server copies the content to the Accelerated Content Services server. The Accelerated Content Services server writes a copy of the content file into a file store at the central location, without removing the copy in the Branch Office Caching Services cache. The content is now available to all users.

Synchronous write with Accelerated Content Services server

1. A user at a remote site uses a web application to perform an operation that includes a new or updated content file.
2. The web application writes the metadata associated with the new or update content file to the repository at the central location.

3. The web application determines the network location of the remote user. Using information from the Accelerated Content Services configuration object, the application generates a URL for writing the content file to the Accelerated Content Services server. It then passes the URL to the UCF client application running on the user's browser machine.
4. The UCF client application writes the content file to the Accelerated Content Services server, using the generated URL.
5. The Accelerated Content Services server writes a copy of the content file into a file store. The content is now available to all users.

3.2.3.7.3 Communication flow when a content pre-caching request occurs

The following sequence describe the basic communications that occur when a pre-caching content job runs. The communication flow is similar when an application requests pre-caching programmatically using the IDfPrecachingOperation package interface in Foundation Java API.

1. The method server on the Documentum CM Server host runs the pre-caching content job. The job definition identifies which objects have their content files pre-cached and to which network locations the files are pre-cached.
2. Using information from Branch Office Caching Services configuration objects in the global registry, the job determines which Branch Office Caching Services servers manage each of the selected network locations.
3. The job posts one request for each copy operation into the Messaging Service queue. The copy operations are necessary to pre-cache the selected objects' content files to the selected Branch Office Caching Services caches. The Branch Office Caching Services servers at each remote site perform the next two steps.
4. If the Branch Office Caching Services server at a remote site is configured for push mode, the Messaging Service server retrieves the pre-caching request from its queue and passes the request to the Branch Office Caching Services server. If the Branch Office Caching Services server is configured for pull mode, the Branch Office Caching Services server contacts the Messaging Service server and retrieves the pre-caching request.
5. The Branch Office Caching Services server sends a request to the closest Accelerated Content Services server that has the content. The Accelerated Content Services server returns the content to the Branch Office Caching Services server, which stores it in its cache.

3.2.3.8 Implementation of distributed storage areas

A distributed store object that points to the component storage areas defines a distributed storage area. Each component storage area consists of a location and storage object.

For example, if you define a distributed storage area with two file store components, it is represented in the repository by one distributed store object, two file store objects, and the two location objects associated with the file store objects.

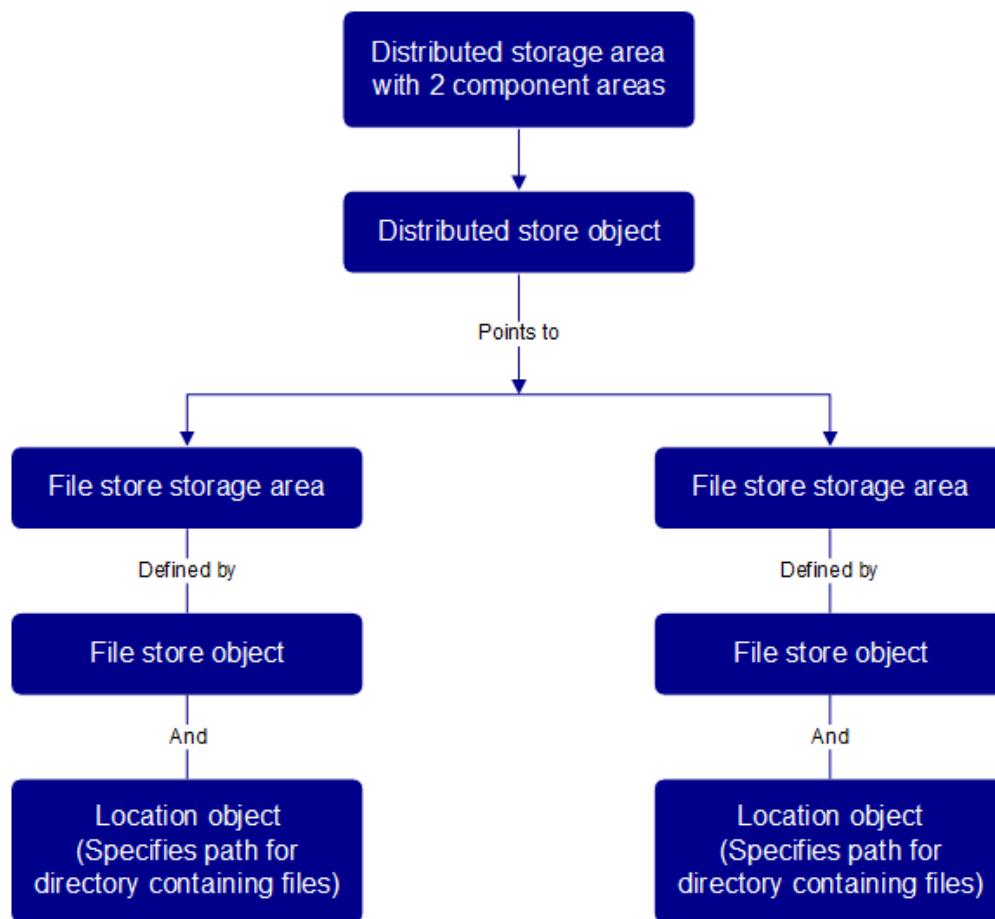


Figure 3-5: Simple example of distributed architecture

You can have as many component areas as needed. In one distributed storage area, the components can be all file stores or all linked stores, or they can be a mixture of the two types. However, all the components must have the same value in media_type property of their defining storage object. The media_type property indicates the formats of the content files in the storage area.

You can encrypt or compress a file store, or use content duplication checking and prevention, or use any combination of these options.

3.2.3.9 Proximity values

Proximity values are used to define:

- A Documentum CM Server's proximity to a connection broker
- An Accelerated Content Services server's proximity to a connection broker
- A network location's proximity to an Accelerated Content Services server
- Setting up multiple Accelerated Content Services servers for load balancing and failover

3.2.3.9.1 Use by Documentum CM Servers

For Documentum CM Servers, proximity values represent the server's distance from the connection brokers. The initial values are set when a Documentum CM Server is installed. You can modify the values.

The values are set in the server configuration object or in the server.ini file. If a server is projecting to multiple connection brokers, the proximity values sent to each connection broker must reflect the server's relative distance from each connection broker. For example, assume that a server is projecting to connection brokers, A and B, and the server is farther from connection broker B than connection broker A. In this case, the proximity value projected to connection broker B is higher than the value projected to connection broker A. The proximity values reflects the topology of the network.

When a client requests connection information, the connection broker returns the proximity value of each known server in addition to other connection information to the client. The client uses these proximity values to distinguish between the data server and remote Content Servers.

To select a server for data requests, the client looks for servers with proximity values of 0 to 999. From that group of values the client selects the server with the lowest proximity value. If servers with a proximity value from 0 to 999 do not exist, the client looks at the servers with proximity values of 9000 to 9999. The client then selects the server from that group with the lowest proximity value as the data server.

To select a server for content requests, the client looks at all servers, compares the first three digits of each proximity value, and selects the lowest value. The digit in the fourth place (0 or 9) is disregarded when choosing a remote Content Server. Only the first three digits, counting left from the ones position, are considered. These first three digits are called a server's content proximity.

For example, assume that a server is projecting a value of 9032 to a connection broker. In this case, the server's proximity value is 9032 but its content proximity value is 032.

3.2.3.9.2 Use by Accelerated Content Services servers

When an Accelerated Content Services server projects to a connection broker, the information tells the connection broker that the Accelerated Content Services server is running. This information is used to determine which Accelerated Content Services server to use when handling content request for a particular user. When asked by Foundation Java API, Documentum CM Server presents list of candidate content files. The Foundation Java API also asks the connection broker for a list of the Accelerated Content Services servers that are running. Using both sets of information, the Foundation Java API selects the file to return to the user.

3.2.3.9.3 Use by network locations

For network locations, the proximity value identifies the location's distance from an Accelerated Content Services server. You can record the proximity values in the Accelerated Content Services configuration objects representing the Accelerated Content Services servers that service the location. Or, the Accelerated Content Services server can use proximity values from its associated Documentum CM Server. Foundation Java API uses the Accelerated Content Services server's proximity value to determine the closest Foundation Java API to the user.

3.2.3.9.4 Setting up an Accelerated Content Services server for load balancing and failover

When you use a OpenText Documentum CM web client for content management and the Accelerated Content Services fails, the UCF client reports to the UCF server that a content transfer failure has occurred. The UCF server reports this information to the Foundation Java API, which then stops using the failed Accelerated Content Services until it projects again. The content is transferred through the application server until Accelerated Content Services starts projecting. This leads to regression in content transfer performance.

To avoid regression in content transfer performance, you can configure multiple cooperating Accelerated Content Services servers for failover. This configuration requires multiple network locations and multiple Accelerated Content Services servers. These Accelerated Content Services servers can either be in different geographic locations or in the same location. In either case, to set up the Accelerated Content Services servers for load balancing and failover, you specify different network locations and different proximity values for each Accelerated Content Services. When one Accelerated Content Services fails, another Accelerated Content Services can process the failed Accelerated Content Services's traffic. Make sure that all the cooperating Accelerated Content Services servers are serving a common repository and are allowed to read/write to the filestore where content is stored. After the configuration, the UCF client receives URLs corresponding to all cooperating Accelerated Content Services servers in the order of configured network proximity values. The UCF client attempts to transfer content through all URLs in the order as mentioned previously until one content transfer succeeds. If content transfer through all URLs fail, the system falls back to content transfer through the application server which may or may not be seamless to the end user, depending on the operations in progress.



Note: Failover or load balancing is orchestrated by the client. If you are using a OpenText Documentum CM web client that does not use a UCF client for content transfer, this functionality may not exist. There is no dedicated failover or secondary Accelerated Content Services to another Accelerated Content Services. High availability deployment of Accelerated Content Services is not supported.

Use the instructions in the *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UCD)* to define network locations and proximity values for the Accelerated Content Services servers. If the Accelerated Content Services servers are in different locations, users use the Accelerated Content Services server located closest (with the lowest proximity value) to their network location. For example, users located in network A with Accelerated Content Services server 1 can use network A with Accelerated Content Services server 1. Users in network B with Accelerated Content Services server 2 can use network B with Accelerated Content Services server 2. If Accelerated Content Services 1 fails for an user of network A, UCF client can use Accelerated Content Services 2 for content transfer if it also can serve that content. Similarly, if Accelerated Content Services 2 fails for an user of network B, UCF client can use Accelerated Content Services 1 for content transfer if it also can serve that content.

3.2.3.10 Shared content

Shared content is managed through shared directories. All sites must share the distributed storage area component's directory at each site.

The feature also uses the settings of the far_stores property and the only_fetch_close property to make sure that content file reads and writes are handled correctly.

The far_stores property is a repeating property. It is defined for the server configuration object type, that contains the names of all storage areas that are considered far for the server. A server cannot save files into a far storage area.

For example, assume that a distributed storage area has component storage areas in Nice, Toronto, and Los Angeles. If the component storage areas in Toronto and Los Angeles are defined as far for the server in Nice, that server can only save files into the component storage area at its own site, Nice. Similarly, for the server in Toronto, if the component storage areas in Nice and Los Angeles are defined as far, the server can only save files into the component area at its site, Toronto.

The only_fetch_close property is defined for the distributed store object type. This property is FALSE by default, which allows servers to read from far storage areas. Setting the property to TRUE means that servers can only read from storage areas not named in the far_stores property of their server configuration objects.

3.2.3.11 Content replication

Content replication is used when a repository has a distributed storage area. There are several ways to replicate content to components of the distributed storage area:

- ContentReplication tool

The ContentReplication tool provides automatic replication on a regular schedule. This tool is implemented as a job. After you define the parameters of the job, the agent exec process executes it automatically on the schedule you defined.

- Surrogate get

The surrogate get feature provides replication on demand. If you use surrogate get, when users request a content file that is not in their local storage area, the server automatically searches for the file in the component storage areas and replicates it into the user's local storage area.

For surrogate get, you can use the system-defined `SurrogateGet` method or you can write your own program.

- `REPLICATE` or `IMPORT_REPLICA`

The two administration methods, `REPLICATE` and `IMPORT_REPLICA`, let you replicate content manually. On Linux, these two methods require that the servers be able to connect using NFS.

3.2.4 Multi-repository distributed models

In multiple repository distributed models, entire objects, both content and metadata, are distributed between repositories. The distribution can occur through user-defined object replication jobs, or internally, when a user manipulates objects from multiple repositories in one repository session. For example, an internal replication occurs when a user starts a repository session and opens an email-attached document sent from another repository.

This section discusses the basic multi-repository replication model and the federation model. Both models are based on object replication. The federation model provides system-defined jobs that automate much of the administration work required to make sure that object replication works correctly.

3.2.4.1 Multiple repositories using object replication

Object replication replicates objects, both content and metadata, between repositories. Object replication jobs are user-defined. In object replication, there is a source and target repository. A replication job replicates objects from the source repository to the target repository. Which objects are replicated and how often the job runs is part of the job's definition. In the target repository, the replicated objects are marked as replica objects.

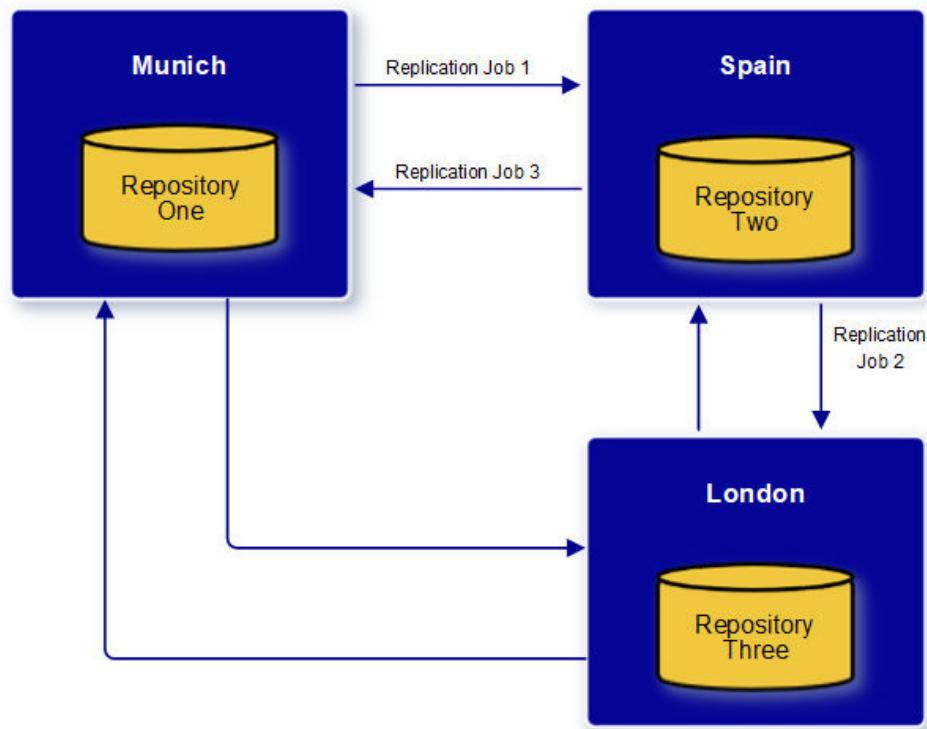


Figure 3-6: Object replication model architecture

3.2.4.2 Multiple repositories working as a federation

When sites have multiple repositories and users are accessing objects from those repositories in one session, maintaining consistent security is imperative. The definitions of users, groups, and ACLs must be the same across the repositories. Keeping users, groups, and ACLs synchronized in multiple repositories can be time-consuming and error prone when the work is done manually in each repository. Placing the repositories in a federation automates much of the process.

In a federation, one repository is the governing repository. The remaining repositories are member repositories. Changes to global users and groups and

external ACLs in the governing repository are propagated from the governing repository to all the member repositories automatically.

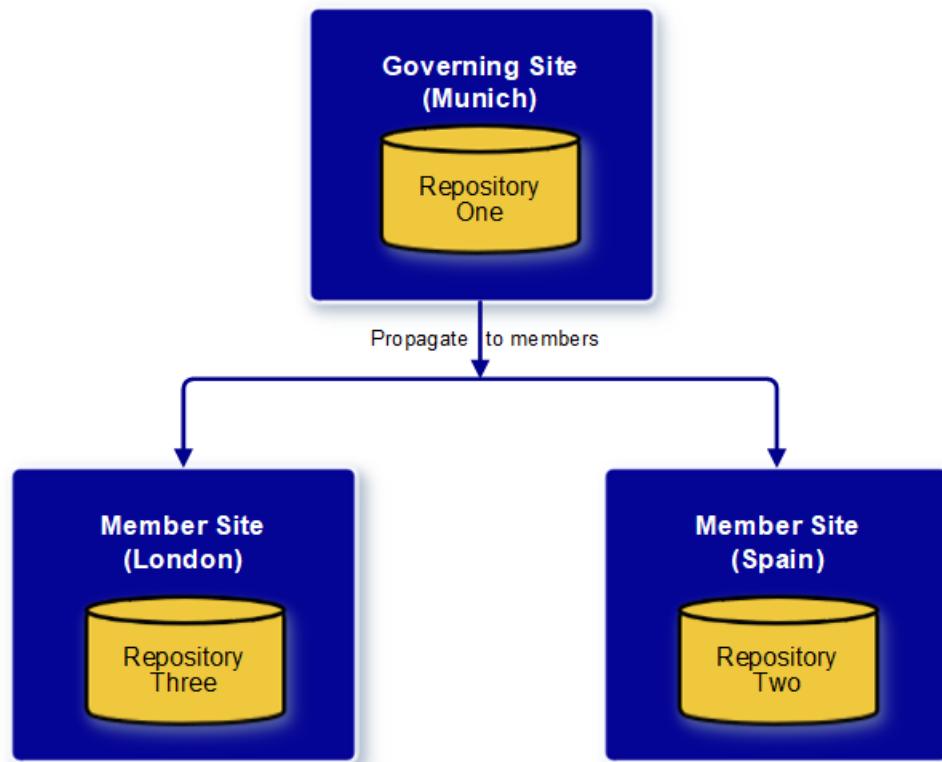


Figure 3-7: Federation Model

3.2.5 Building block architectures for multi-repository models

This section describes the features that implement a multi-repository distributed object environment.

3.2.5.1 Reference links

This section describes the object types that support reference links and how reference links are handled in cross-repository applications.

3.2.5.1.1 Object type implementation

A reference link in a repository represents an object in another repository. Each reference link comprises a pair of objects: a dm_reference object and either a mirror object or a replica object.

3.2.5.1.1.1 Mirror objects

A mirror object is an object in one repository that mirrors an object in another repository. The term mirror object describes the object's function. It is not a type name. For example, if a user logs on to repository A and checks out a document in repository B, the server creates a document in repository A that mirrors the document in repository B. The mirror object in repository A is an object of type dm_document.

Mirror objects only include the original object's property data. Any content associated with a remote object is not copied to the local repository when a mirror object is created.

Only a limited number of operations can affect mirror objects. For example, users can link mirror objects to local folders or cabinet or retrieve their property values. Most operations affect the remote objects represented by the mirror object.

3.2.5.1.1.2 Replica objects

Replica objects are created with an object replication job is run to replicate objects from one repository to another. The term replica object represents the object's function. It is not an object type name. When an object replication job runs, it creates replicas in the target repository of the objects it is replicating. The replicas have the same object type as the source objects that were replicated.

3.2.5.1.1.3 Reference objects

Every mirror object has an associated dm_reference object. A dm_reference object is the internal link between the mirror object and the source object in the remote repository. Reference objects are persistent. They are stored in the same repository as the mirror object, and Documentum CM Server manages them. Users never see reference objects.

3.2.5.1.1.4 Valid object types for reference links

Foundation Java API supports reference links only for objects of the following types:

- cabinet
- docbase config
- document
- folder
- note
- procedure
- query
- script
- server config
- smart list
- SysObject

Foundation Java API does not support creating reference links for other object types.

3.2.5.1.2 Reference link binding

By default, the operation that creates the reference link also defines which version of the source object is bound to the reference link. For example, when users check out a document, they identify which version they want. If they are checking out a remote document, the specified version is bound to the reference link.

If the version is not identified as part of the operation, the server automatically binds the CURRENT version of the source object to the reference link.

You can change the default binding.

3.2.5.1.3 Reference link storage

When linking operation creates a reference link, the mirror object is stored in the requested location. For example, if a user links a document from repository B to Folder A in repository A, the mirror object is stored in Folder A.

When other operations create a reference link, the mirror objects are stored in folders in the local repository under /System/Distributed References. For example, mirror objects created when users check out remote objects are stored in /System/Distributed References/Checkout.

3.2.5.1.4 Type-specific behavior of reference links

Foundation Java API does not allow users to link local objects to a cabinet or folder that is a reference link. For example, suppose a folder from the Marketing repository is linked to a cabinet in the Engineering repository. Users in the Engineering repository cannot add objects to the folder.

If users run scripts or procedures that are reference links, Foundation Java API fetches them from the remote repository. It then runs them against the local repository session. The scripts or procedures are not run against the remote repository.

3.2.5.1.5 Reference link updates

A reference link is most useful when its mirror or replica object reflects the actual source object. When you change the source object, reflect those changes in the mirror or replica object. Keeping the reference link synchronized with the source occurs automatically in most cases.

The dm_DistOperations job automatically refreshes replicas.

Mirror objects are refreshed using an IDfPersistentObject.refresh method.

3.2.5.1.6 Operations on replica objects

When an object is replicated into a repository, the replica is given a local object ID. The operation also creates a reference object for the replica that points to the replica's source object.

When users perform operations on a replica object, the operation can affect the replica or it can affect the source object. Which is affected depends on the operation and how the method's redirection policy is set. Each Foundation Java API method that can operate on a replica has a redirection policy defined for it in an annotation. That policy determines whether the method operates on the replica or is redirected to the source object.

The redirection policies, which the com.documentum.fc.client.distributed.replica.ReplicaRediretPolicy class defines, are:

- NEVER

This policy is always run against the replica object. This policy is the default behavior and is appropriate for methods that access only local properties or are always called by methods whose redirection policy has been set.

- ALWAYS

This policy redirects the method to the source object. This policy is used for methods that change global properties.

- ON_GLOBAL_CHANGE

This policy directs the method against the source if a global property for the replica has been changed. This policy is used for operations that read global properties.

- RUNTIME_TEST

This policy redirects the method to the source object based on the results of a specified runtime test.

- ON_GLOBAL_CHANGE_AND_RUNTIME_TEST

This policy redirects the method to the source object if a global property for the replica has been changed and a runtime test indicates that redirecting the method is appropriate. This policy is for internal use only.

The annotations are implemented using the com.documentum.fc.client.distributed.replica.ReplicaMethodBehavior annotation.

To perform an operation on a replica that affects the source object, the repository in which the source object resides must be active. That is, the source repository must have a server that is running.

You cannot assign replica objects to a retention policy.

3.2.5.1.7 Reference link security

This section describes how security is applied to reference links.

3.2.5.1.7.1 Mirror objects

Documentum CM Server for the source repository controls security for the source object, regardless whether the update is made directly to the source or through a reference link.

Documentum CM Server for the repository that contains the mirror object controls security on the mirror objects. The ACL applied to the mirror object is derived from source object's ACL using the following rules:

- If the source object's ACL is a private ACL, the server creates a copy of the ACL and assigns the copy to the mirror object.
- If the source object's ACL is a system ACL and a system ACL is in the local repository with the same name, the server assigns the local system ACL to the mirror object.

In this situation, the access control entries in the system ACLs are not required to match. Only the ACL names are matched. It is possible in these cases for users to have different permissions on the source object and the reference link.

- If the source object's ACL is a system ACL and no system ACL with a matching name is in the local repository, the server creates a local copy of the ACL. It then assigns the copy to the mirror object. The copy is a private ACL.

3.2.5.1.7.2 Replicas

The local Documentum CM Server manages replica security. Each replica is assigned an ACL when the object replication job creates the replica. The job's replication mode determines how the ACL is selected.

3.2.5.2 Object replication

Documentum CM Server's replication services allow you to replicate objects (property data and content) between repositories. Replication is automated, using jobs. Using parameters you define, the jobs dump a set of objects from one repository, called the source repository, and load them into another repository, called the target repository. You can initiate jobs by:

- The source repository
- The target repository
- A third repository that starts the job but is neither the source nor the target repository.

A repository that performs this function is called a *mediator* repository.

A replicated object in the target repository is called a replica and is designated by a replica icon.

3.2.5.2.1 Replication jobs

A replication job is defined as a job object in the repository that initiates the job. The properties of the job object define the parameters of the replication job. They include how often the job is executed, the source and target repositories, what is replicated, and the security assigned to the replicas. Additional properties provide status information about the job. They can tell you the time at which it last executed, the time of the next expected execution, and whether the job is active or suspended.

Jobs are created using Documentum Administrator by users with superuser privileges.

3.2.5.2.2 Multi-dump file replication jobs

A replication job puts the objects to be replicated in a dump file. That file is then used to load the target repository. By default, all objects replicated by a single job are placed in a single dump file. If the job replicates a large number of objects, the size of the generated dump file can be large because replication jobs replicate the specified objects and associated objects as needed.

A large dump file can be problematic. For example, the size can be a problem if the job must transfer the file to a target repository over an unreliable network. Size can also be a problem when either the source or target repository has limited disk space for the file.

To avoid problems arising from a large dump file, you can use the -objects_per_transfer argument to direct the replication method to break down the

job into a series of smaller dump and load operations. Each of the smaller operations dumps and loads a specified portion of the objects to be replicated. For example, suppose you want to replicate 100,000 objects in a single job. You can define the job to use a single dump file, containing all 100,000 objects and associated objects. Or you could set the argument to 10,000, which causes the job to use a series of smaller dump and load operations, each replicating 10,000 objects (plus associated objects).

Replication jobs defined as a series of smaller dump and load operations continue until all objects are replicated. The job dumps a specified number of objects and associated objects, loads those objects into the target. It then dumps the next set of objects and loads those objects. No manual intervention is necessary.

3.2.5.2.2.1 Best use of multiple dump file replication

Defining a replication job to use multiple dump and load operations is beneficial if:

- If all objects are put into a single file, then the disk space on the source or target sites cannot accommodate a dump file of the size that would be generated.
- You want to avoid long, continuous remote dump or load operations
Such operations require an operational network connection for the duration of the operation, which can require special configuration of firewalls.
- An unreliable network makes it difficult to support an automatic transfer of large files.
- If the replication job encounters an error, then you want to limit the amount of lost work.

3.2.5.2.2.2 Conditions of use

Defining a replication job to use multiple dump and load operations is subject to the following conditions:

- You must define the job as a fast replication job.
- You cannot define the job as a manual transfer job.



Note: If any of the preceding conditions are true , the -objects_per_transfer argument is automatically set to unlimited, so that all objects are replicated in one operation.

3.2.5.2.3 Replication modes

There are two replication modes: nonfederated and federated. The modes determine how the owner_name, acl_name, acl_domain, and a_storage_type properties of each replicated object are handled.

3.2.5.2.3.1 Nonfederated replication mode

In the nonfederated replication mode, the owner_name, acl_name, acl_domain, and a_storage_type properties are considered local properties. This mode is typically used when the source and target repositories are not members of the same federation.

Nonfederated replication mode provides two choices for handling the owner_name, acl_name, and acl_domain properties in the target repository. You can:

- Remap the properties to default values provided in the replication job's definition

If you select this option, all replicas that the job creates have the same ACL. They are stored in the same storage area in the target repository.

The ACL entries can be as open or restrictive as needed. This means that a user who has access to a document in one repository may not necessarily have access to the document's replicas in other repositories. The ACL assigned to the replica in that repository controls access in each repository.

If you select this option and fail to provide a default ACL, the server creates a default ACL. This ACL gives Relate permission to the world and group levels and Delete permission to the owner.

- Preserve current security and storage settings when possible

If you select this option, the owner_name property is reset to the default value. However, the server does not automatically assign default values to the security and storage properties. Instead:

- If the source object has a public ACL, then the server looks for a matching ACL in the target repository.

An ACL is a match if the acl_name and acl_domain values match those values of the source object's ACL. If the server finds a matching ACL, that ACL is assigned to the replica. Otherwise, the replica's acl_name and acl_domain properties are assigned the default values specified in the job definition.

- The server looks for a matching storage area in the target repository.

If the storage area has the same name as the storage area defined in the source object's a_storage_type property, then the storage area is a match. If the server does not find a matching storage area in the target repository, the a_storage_type property in the replica is set to the storage area specified in the job definition.



Note: When a repository is configured, a storage area named `replica_filestore_01` is created. This area can be specified in the job definition as the storage area.

If there is no storage area whose name matches the source storage area and a storage area is not defined in the job, the content is placed in the default storage area defined for the `dm_sysobject` type.

- The `group_name` property of the replica object is always reset to the default group of the user specified in the replica's `owner_name` property.

3.2.5.2.3.2 Federated replication mode

In federated mode, global security controls the replica objects. The source repository retains control of the ownership and security of the replicas. The `owner_name`, `group_name`, `acl_name`, and `acl_domain` properties are not mapped to local values in the target repository. Their values can only be changed from the source repository.

This means that the `owner_name` and `group_name` properties in the replicas can contain values representing users or groups that do not exist in the target repository.

Federated replication handles ACLs in the following way:

- If the target repository contains an ACL that matches the ACL of a source object, the object's replica is assigned the local matching ACL.

An ACL is a match if its `acl_name` and `acl_domain` property values match those values of the source object's ACL. The entries in the ACL are not required to match those entries in the source object ACL.

If the source ACL is a public ACL, the `acl_domain` value is reset to `dbo` when it is written into the replication dump file. When it is loaded into the target repository, `dbo` is considered a match for the owner of the target repository.

- If there is no matching ACL in the target repository, you have two options. You can select to:

- Preserve security

The source ACLs are replicated into the target repository. The `acl_name` and `acl_domain` values are retained.

- Remap security

If the source ACL is an external ACL, the replica's `acl_domain` is reset to `dbo`. The `acl_name` is reset to the default value specified in the replication job definition.

In federated mode, the `a_storage_type` property is a local property. If the target repository contains a storage area of the same name, the replica content is placed in that storage area. If no storage area with the same name exists, the replica's `a_storage_type` property is reset to the storage area specified in the replication job definition. The replica's content is placed in that storage area.



Note: When a repository is configured, a storage area named `replica_filestore_01` is created. This area can be specified in the job definition as the storage area.

If there is no storage area whose names matches the source storage area and a storage area is not defined in the job, the content is placed in the default storage area defined for the `dm_sysobject` type.

3.2.5.2.4 What is replicated

The first time an object replication job runs, the job replicates all objects of type `dm_sysobject`. Its subtypes are linked to the source folder or cabinet (and any subfolders) specified in the job definition. The entire version tree for each object is replicated.



Note: It is up to you to determine which documents and folders to link to the source folder or cabinet and to make sure that the linking is done. The server does not select objects for replication.

The folder hierarchy within the source cabinet or folder is recreated in the target cabinet or folder. (The target cabinet or folder is defined in the job object.) The source cabinet or folder itself is not recreated in the target repository.

If a remote folder is linked to the source folder, the replication job replicates the reference link representing the remote folder. The job does not replicate the remote folder or any objects in the remote folder.

Additionally, the first run of a replication job also replicates certain objects that are related to the objects in the source folder or cabinet:

- All annotations attached to a replicated object
- All renditions of the replicated object
- If the replicated object is a parent in any user-defined relationships in the source repository, the child object and the associated relation and relation type object are replicated.
- If the replicated object is a virtual document, all of its components are replicated.



Note: If a component is a remote object and a virtual document or a folder, the replication does not traverse the remote object to replicate its children (in the case of a virtual document) or any objects it contains (in the case of a folder).

- If the replicated object has an associated assembly, all of the objects in the assembly are replicated.



Note: It is possible to create an assembly for a virtual document and attach the assembly to a simple document. If such a simple document is replicated, the components of the assembly attached to it are also replicated.

- If the object is attached to a lifecycle, the policy object representing the lifecycle is replicated.



Note: The replica lifecycle does not function as a lifecycle in the target repository. Documents cannot be attached to replica lifecycles.

- If a replicated object has properties that reference other SysObjects by their object IDs, those referenced SysObjects are also replicated with one exception.

The exception is retainer objects. If a replicated object is assigned to a retainer object (a retention policy), the retainer object is not replicated to the target repository. The `i_retainer_id` property in the replica is not set.

- If the replication mode is federated, any events associated with the replicated objects are also replicated.

What is replicated on subsequent runs of the job depends on whether the job is defined as fast replication.

If the job is not defined as fast replication, each time it runs, the job replicates all changed objects in the source folder. The job also examines all related objects. It replicates any objects that have changed, even if their parent object (that is, the SysObject to which they are related) is unchanged. The job also replicates any new objects in the source folder and their related objects.

In fast replication, except for annotations, the job does not consider related objects for replication unless their parent object is changed or has not been previously replicated. If an object in the source folder has not changed since the last replication job, it is not replicated. Only its annotations are considered for inclusion. New or changed annotations are replicated, but the object's other related objects are not included in the job.

If an object in the source folder has changed, it is replicated. Changed related objects are not replicated unless they are also stored in or linked to the source folder.

A fast replication job also replicates any new objects in the source folder and their related objects.

For example, suppose you have a source folder named `Source_TargetB` with ten documents in it, including a virtual document called `MyVirtualDoc`. `MyVirtualDoc` has two direct components, `X` and `Y`, and `X` has one child itself, `X_1`. The document `X_1` is stored outside the source folder.

Now, suppose you run fast replication jobs with `Source_TargetB` as the source folder to repository B. The first time the job runs, the job replicates all the documents in `Source_TargetB` and their related objects. After the first run, one document, `Mydoc`, is added to the source folder and the document `X_1` is versioned. On the second job run, the job replicates `Mydoc` and all its related objects. However, `MyVirtualDoc` has not changed. Consequently, it is not replicated. Its component, `X_1` is not replicated either because it is not stored in or linked to the source folder.

In fast replication, if you want to replicate related objects regardless of changes in the parent objects, link the related objects to the source folder.

3.2.5.2.5 Aspect modules and replication

Object replication does not replicate aspects associated with a replicated object. If you have aspects associated with specific instances of an object type, create those aspects in the target repository. If you have default aspects defined for an object type and you replicate instances of that type, create the default aspects manually in the target repository. The aspects must be created in the target repository before performing the object replication.

3.2.5.2.6 Format objects and replication

It is recommended that you manually ensure that all format objects for a particular format across participating repositories be the same.

If the format of a replicated SysObject is not present in the target repository, Documentum CM Server creates the format object in the target repository. However, if you later change the format object in the source, the changes are not reflected in the target repository. The changed format object is not re-created in the target the next time the job runs. To make sure that format definitions are the same across participating repositories, create and maintain them manually.

3.2.5.2.7 Data dictionary information and replication

With one exception, data dictionary information is not replicated for object types or properties. The exception is the dm_policy object type. If the replication job replicates a dm_policy object, the data dictionary information for the policy object type and its properties is replicated also.

In the target repository, data dictionary information for any object types or properties created by the replication process is available after the next execution of the data dictionary publishing job.

3.2.5.2.8 Display configuration information and replication

Display configuration information is information used by client applications to determine how to display one or more properties when they appear in a dialog box. Display configuration information is stored in the repository in scope configuration and display configuration objects. When you install Documentum CM Server, a script sets up default display configurations for properties in dm_sysobject, dm_document, and dm_folder.

Object replication does not replicate scope configuration and display objects associated with any object type. If you have defined display configuration information for types other than dm_sysobject, dm_document, or dm_folder, or have modified the information for these types, explicitly set it in the target repository.

If you have defined such information in the source repository and you want the same display configuration for the properties in the target repository, set it up explicitly in the target repository.

Client applications display configuration information.

3.2.5.2.9 Full refreshes

It is possible to run a replication job as a full refresh job. Such a run behaves as a first-time execution of the job and update all replicas, regardless of whether the source has changed. However, full refreshes do not change the r_object_id property of the replicas in the target repository.

3.2.5.2.10 Related objects storage

Those objects that are related to objects in the source folder and are replicated because of their association with the source objects are stored in folders called Related Objects in the target repository. There is one Related Objects folder for each replication job that targets the repository. The Related Objects folder is located under a job's target folder in the target repository.

3.2.5.2.11 How the replication process works

Replication services use the agent exec process that is part of Documentum CM Server. This process runs continuously and, at intervals, scans the repository for jobs that are in need of execution. The process uses properties in the job definition to determine whether to launch the job. These properties define how often the job is run and when.

When the agent exec program finds a job that must be run, it executes the method that runs the program associated with the job. This method is also specified in the job definition. For replication, the method is the replicate_folder method, which is created as part of the replication installation and configuration process. The replicate_folder method runs the replicate_folder program, which is provided as part of Documentum CM Server.

The replicate_folder program performs the actual work of replication. In the source repository, the program:

1. Dumps the source cabinet or folder to a file.
2. Performs delete synchronization.
3. Transfers the dump file to the target repository.
4. Filters the file.
5. Loads the filtered file into the target repository.

Delete synchronization makes sure that any objects that have been deleted from the source repository are also deleted from the target repository.

3.2.5.3 Federations

Federations are created using Documentum Administrator. Define a repository as the governing repository in a federation, add member repositories, and activate the jobs that keep global users, groups, and external ACLs synchronized.

3.2.5.3.1 Supporting architecture

Internally, federations are supported by:

- A dm_federation object in each participating repository
- A property in the associated repository configuration objects
- The globally_managed property in the user, group, and ACL object types
- The replicate_temp_store storage area

The federation object in each repository is named for the federation in which the repository is participating. The properties in a federation object list the federation's members and the governing repository. They also provide information about the federation's management, such as when each federation member was last updated. For more information about the list of properties defined for the federation type, see *OpenText Documentum Content Management - Server System Object Reference Guide (EDCCS250400-ORD)*.

In addition, each repository configuration object has a property called r_federation_name that contains the name of the federation to which the repository belongs, if any. This property is provided as insurance because it is possible for a repository to have multiple federation objects. In such instances, the server uses the federation object whose name matches that in the r_federation_name property in the repository configuration object.

When a repository becomes a member of a federation, the governing repository's name and r_federation_name value are projected to all its connection broker targets immediately. This action reflects the repository's new membership.

If the globally_managed property is set for a user, group, or ACL, that object is managed through the governing repository. Changes to globally managed objects in the governing repository are propagated to all member repositories. Documentum Administrator provides an easy tool for changing globally managed objects.

The replicate_temp_store storage area is used as a temporary storage place for the content of replication jobs. Object replication jobs create dump files that must be transferred from the source repository to the target repository. These files are held in the replicate_temp_store storage area until the replication job completes. Then Documentum CM Server removes the files.

The directory location of the replicate_temp_store storage area for a particular repository is %DOCUMENTUM%\data\replicate_temp_store\<repository_id> (Windows) or \$DOCUMENTUM/data/replicate_temp_store/<repository_id>, where <repository_id> is the hex value.

3.2.5.3.2 Jobs and methods

After a federation is created, global users, groups, and external ACLs are synchronized automatically using jobs and methods. Documentum Administrator manages any changes, additions, and deletions to users and groups using a change record file. The changes recorded in the file are propagated to all members by jobs. The Documentum Administrator manages external ACLs using replication jobs.

Documentum Administrator activates the jobs that perform these operations.

3.2.6 Distributed environments and the secure connection defaults

When users or applications connect to a server, they request either a secure or native (unsecured) connection. By default, all connection requests default to a request for a native connection and all servers listen on a native port. However, a server can be configured to listen on a secure port or on both a secure and a native port. If a server is listening on a secure port, then a client can request a secure connection with that server.

The building blocks that support single-repository, distributed environments rely on connections between servers within a repository. The building blocks that support multi-repository distributed environments rely on connections between repositories. Configure compatible secure connection defaults for the servers and clients to ensure successful connection requests.

There are two ways to configure the secure connection default settings to avoid connection failures:

- Set the `secure_connect_mode` property for the servers to `dual`.
`secure_connect_mode` is a server configuration property. If you set the property to `dual` for a server, the server listens on both a native and a secure port. Consequently, a client requesting a connection through that server can request either a native or a secure port with success. For more information about the `secure_connect_mode` property and its settings, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.
- Set the `dfc.session.secure_connect_default` key in the client's `dfc.properties` file to either `try_native_first` or `try_secure_first`.
`try_native_first`, the default setting for this key, directs Foundation Java API to try to obtain a native connection first and if that fails, to request a secure connection. `try_secure_first` directs Foundation Java API to try to obtain a secure connection first and if that fails, to request a native connection. In either case, the target server's default setting does not affect the success of the connection request.

The jobs and features provided with Documentum CM Server that support single- or multi-repository environments use a `dfc.properties` file. This file is located on the host machine on which the job resides. The `dfc.properties` file used by client products is on client application's host machine or in a network location determined that you configure. For more information about the default settings for the client, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

3.2.7 Distributed messaging

Distributed messaging is a feature of Documentum CM Server that supports multi-repository configurations. The server copies events from one repository to another to facilitate distributed workflows and distributed event notification.

Distributed workflow occurs when an application or user assigns an activity in a workflow definition in a repository to a user in another repository. When the activity is started, the server creates a distributed task (queue item) in the local repository that is copied to the user's home repository. It appears in the user's home repository inbox.

Distributed event notification occurs when users register for event notifications in a repository that is not their home repository. When the event occurs, the server in the repository that contains the registration creates a distributed event notification (queue item). This queue item is copied to the user's home repository. The event notification appears in the user's home repository inbox.

Distributed messaging is implemented using five properties in the `dmi_queue_item` object type. This table briefly describes these properties.

Table 3-1: Properties implementing distributed messaging in `dmi_queue_item`

Property	Description
<code>source_docbase</code>	The repository in which the event or task originates.
<code>target_docbase</code>	The home repository of the user receiving the event or task.
<code>remote_pending</code>	TRUE if the <code>dmi_queue_item</code> must be copied to the target repository. FALSE if it has already been copied or is not a distributed queue item.
<code>source_event_id</code>	Object ID of the source <code>dmi_queue_item</code> object that generated this queue item in the target repository. This property is only set in the target repository.

Property	Description
source_event_stamp	The i_vstamp value of the source dmi_queue_item object that generated this queue item in the target repository. This property is only set in the target repository.

3.3 Installing Branch Office Caching Services servers

3.3.1 Overview

This section contains the instructions for installing, uninstalling, and starting and stopping the component.



Note: The Branch Office Caching Services server is a component of a distributed configuration. For information about installing other components, such as Documentum CM Server, see “[Documentum CM Server](#)” on page 9.

The Branch Office Caching Services server is supported on the same operating systems as Documentum CM Server and you can install in a heterogeneous environment. The database client is not required on the Branch Office Caching Services server host because the Branch Office Caching Services server does not interact with the database.

Branch Office Caching Services servers are supported in heterogeneous environments. For example, a Branch Office Caching Services server installed on a Windows host might serve as a repository and Documentum CM Servers running on Linux, or a Branch Office Caching Services server installed on a Linux host might serve as repository and Documentum CM Server on a Windows host.

3.3.2 Prerequisites

- Download and install the supported version of JDK from Oracle JDK/OpenJDK website. For more information, see “[Installing JDK](#)” on page 35.
- Set the JAVA_HOME environment variable to point to the installed Oracle JDK/OpenJDK version.
- Set the PATH environment variable to JAVA_HOME\bin.
- The host environment meets the system requirements.
- If you do not want to accept the defaults, select directories and port numbers available for use during installation.

Default directory locations are suggested for the installation directory, the content cache, and the parked content cache. Default port numbers are suggested for the Branch Office Caching Services instances. The port 8086 is used for the instance server.

- You know the host name of the Messaging Service installation and port number to use to communicate with the Messaging Service server if the Branch Office Caching Services server is configured to operate in pull mode.
- Select the user name and password for the user who administers the Branch Office Caching Services server's `acs.properties` file through the resource agent in Documentum Administrator.
- If you are installing on Linux, ensure that the `DOCUMENTUM` and `DOCUMENTUM_SHARED` environment variables are set.
 - Set `$DOCUMENTUM` to the installation directory.
 - Set `$DOCUMENTUM_SHARED` to the directory in which you want to install Foundation Java API on the Branch Office Caching Services host. This directory is also the top-level directory under which the scripts used to remove the components are stored.



Note: The installation owner must have read, write, and execute permissions on these directories and their subdirectories.

- Select a password for the `bocsAdmin` user.

The `bocsAdmin` user is the user who administers the Branch Office Caching Services server's `acs.properties` file through the resource agent in Documentum Administrator.

- The host name where Branch Office Caching Services is deployed must not contain any special characters as a trailing period.

3.3.2.1 Installing JDK

Download and install the supported version of Oracle JDK/OpenJDK version from Oracle JDK/OpenJDK website. The *Oracle JDK* and *OpenJDK* documentation contains detailed information.

3.3.2.1.1 Using SSL communication

For information, see “[Using SSL communication](#)” on page 36.

3.3.3 Installing Branch Office Caching Services server

1. Download and extract the compressed distribution file.
2. If you want to use HashiCorp Vault, perform all the tasks as described in “[Configuring Documentum Secret Integration Service](#)” on page 49 and ensure that DSIS is running.
3. Run the installation program.
 - Windows: `bocsSetup.exe`
 - Linux: `bocsSetup.bin`

If an error occurs during the setup for installation, that error is recorded in the `setuperror.log` file. During the installation process, any messages relating to the process are saved to a file named `install.log`. Both of these files are stored in the same directory where the downloaded files were decompressed.

4. Accept the license agreement and click **Next**.
5. If you want to configure the HashiCorp Vault secrets, on the **Vault configuration** page, select the **Enable Vault** check box.
6. If you select the **Enable Vault** check box, provide the following information and click **Next**:
 - a. **DSIS URL:** Provide a value in the following format:
`http://localhost:<port mentioned in application.properties>/dsis`
 - b. **DSIS Token:** Provide the `dsis.dctm.token` token value as described in “Configuring Documentum Secret Integration Service” on page 49.

! **Important**

If you enabled the HashiCorp Vault configuration, the installer retrieves all the password information automatically from the HashiCorp Vault server. Ensure that you have stored the following secrets in the HashiCorp Vault server:

Table 3-2: Secret and key name

Secret name	Key name	Description
INSTALL_OWNER_PASSWORD	<hostname>	Branch Office Caching Services installation owner password. For example, <code>INSTALL_OWNER_PASSWORD/host1.abclab.net</code> .
DCTM_SERVER_JMX_PASSWORD	<hostname>	Tomcat administrator owner password. For example, <code>DCTM_SERVER_JMX_PASSWORD/host1.abclab.net</code> .
DCTM_BOCS_KEYSTORE_PASSWORD	<hostname>	Keystore password. For example, <code>DCTM_BOCS_KEYSTORE_PASSWORD/host1.abclab.net</code> .



Note: The host name must be a fully qualified domain name.

7. Specify the destination directory. For example, `C:\Documentum` and click **Next**.
8. Provide the password information and click **Next**.

- If HashiCorp Vault is enabled, the installation owner password is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, type the installation owner password.
9. Provide the administrator user password and click **Next**. The default port is 8086.
- If HashiCorp Vault is enabled, the administrator user password is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, type the administrator user password.
-  **Note:** Ensure that you follow the password complexity rules. For more information, see “[Password complexity rules](#)” on page 52.
10. On the **Specify BOCS options** page, specify the location and maximum sizes of the content caches.
- To select a directory for the cache directory that is different from the default, use **Browse**.
 - To select a maximum size for the cache directory that is different from the default, type the new value in megabytes.
 - To select a directory for the Prime Store directory that is different from the default, use **Browse**.

The Prime Store directory is the location where parked content is preserved.

- To select a maximum size for the Prime Store directory that is different from the default, type the new value in megabytes.

11. On the **Pull Mode** page, enable or disable pull mode.

- a. To enable push mode, clear **Enable Pull Mode**.
- b. To enable pull mode, select **Enable Pull Mode**.
- c. In **DMS URL**, type the URL that the Branch Office Caching Services server uses to connect to Messaging Service. If you are connecting to multiple Messaging Service servers in the High availability (HA) mode, type multiple, comma-separated Messaging Service URLs.

In pull mode, Branch Office Caching Services servers retrieve messages from Messaging Service. The URL must be in the following format:

```
http://<host>:<port>
```

where *<host>* is the host of the Messaging Service server and *<port>* is the port on which the Messaging Service server is listening.

- d. In **Config Object Name**, type the name of the Branch Office Caching Services configuration object.

Record the name you select for later use when actually creating the Branch Office Caching Services configuration object. The name is recorded in the `acs.properties` file by the installation program. If you use a different name when creating the Branch Office Caching Services configuration object, edit the `acs.properties` file to change the name.

- e. In the **Keystore Password** box, type the keystore password for Branch Office Caching Services. The length of the password must be greater than or equal to 7.
 - If HashiCorp Vault is enabled, the keystore password for Branch Office Caching Services is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, type the keystore password for Branch Office Caching Services.
 - f. Record the **Path to Certificate** value that is displayed.
This directory path is required later, when the Branch Office Caching Services configuration object is created.
12. On the **Proxy Settings** page, indicate whether a proxy server is to be installed between Branch Office Caching Services and Documentum CM Server.
- a. If you do not want to install a proxy server, select **No**.
 - b. If you want to install a proxy server, select **Yes** and specify the proxy server's host name and the port used to contact the proxy server.
13. Click **Install**.
14. Using Documentum Administrator, create a Branch Office Caching Services configuration object in the global registry.



Note: Go to `$DOCUMENTUM\patch` and open the `patch-info.xml` to view the version details. The `patch-info.xml` file displays the installed component version (base or patch or both). It is not meant for patch release versions only.

3.4 Post-installation task

3.4.1 Licensing OpenText Documentum CM

OpenText Documentum CM uses OpenText Directory Services (OTDS) to apply licenses for all the OpenText Documentum CM components. For more information about procuring the license file and configuring OTDS and license, see “[Licensing OpenText Documentum CM](#)” on page 86.

3.5 Configuring Branch Office Caching Services servers

This section describes how to configure Branch Office Caching Services.

3.5.1 Configuration requirements

To use Branch Office Caching Services servers:

- A Branch Office Caching Services configuration object representing the Branch Office Caching Services server must reside in the global registry.
- Configure the `acs.properties` file correctly for each Branch Office Caching Services server. The installation process configures an `acs.properties` file automatically.
- The ports on which Branch Office Caching Services server serves content to users must be open.
- If the Branch Office Caching Services server is configured in pull mode, the Branch Office Caching Services server must have access to the URL defined in the `dms.pulling.url` property in the `acs.properties` file.
- If the Branch Office Caching Services server is configured in push mode, the Messaging Service server must have HTTP access to the Branch Office Caching Services server.
- To use the pre-caching feature, the capability must be enabled in the content transfer configuration object in a repository that owns the content.



Note: This feature is enabled by default.

- If Branch Office Caching Services is configured to use the Asynchronous Write mode, ensure that the `dm_AsyncWrite` job is enabled in Documentum CM Server. Use Documentum Administrator to enable the `dm_AsyncWrite` job.

3.5.2 Messaging Service and global registry compatibility requirements

Pre-caching jobs and asynchronous content transfers require Foundation Java API to send certain information to Branch Office Caching Services. Foundation Java API sends this information to Messaging Service, and Messaging Service delivers this information to Branch Office Caching Services later. Foundation Java API uses the `DmsClient` service-based business object (SBO) to communicate to Messaging Service. This SBO is deployed in a global registry repository and is downloaded by using Business Object Framework (BOF). For the `DmsClient` SBO to correspond to Messaging Service, Messaging Service and the global registry must be of the same version. This situation is not considered to be an issue because there is only one Messaging Service and global registry per installation.

3.5.3 acs.properties file

This section provides basic information about the file.

3.5.3.1 acs.properties file location

The acs.properties file is located as follows:

- Windows: %DOCUMENTUM%\tomcat\webapps\bocs\WEB-INF\classes\config\acs.properties
- Linux: \$DOCUMENTUM/tomcat/webapps/bocs/WEB-INF/classes/config/acs.properties



Note: The acsfull.properties file, which is in the same directory as acs.properties is located, describes all parameters and their default values.

3.5.3.2 File administration

The file is administered using Java Management Extensions (JMX), accessed through the Branch Office Caching Services resource agent in Documentum Administrator. The resource agent is accessed through the **Resource Management** node. For more information about accessing the resource agent, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)* or *OpenText Documentum Content Management - Server Administration and Configuration Guide (EDCCS250400-AGD)*. The URL for the JMX resource agent is as follows:

```
service:jmx:rmi:///jndi/rmi://<host_name>:<port>/bocs
```

where

- <host_name> is the name or IP address for the host computer.
- <port> is the assigned port. The default port is 8087.

If you want to connect to the Branch Office Caching Services administration resource agent from outside of a firewall, configure the firewall settings to allow the Remote Method Invocation (RMI) protocol for the port.

3.5.3.3 Changing the JMX user password

When you installed Branch Office Caching Services, you provided a user name and password for the JMX server. You can change the password for that user.

To change the JMX user password:

1. Log on to Documentum Administrator as the installation owner.
2. Go to the Branch Office Caching Services resource agent.
3. Double-click the resource agent to display the Mbeans in the agent.

4. Right-click the JmxUserManagementMBean and click **Operations**.
5. Click **changePassword**.
6. Type the current user name and the new password.
7. Click **Start Operation**.

3.5.4 Default settings

The Branch Office Caching Services installation program explicitly sets a number of configuration keys in the `acs.properties` file for the Branch Office Caching Services server.

3.5.4.1 Configuration keys that cannot be changed

Do not change the `mode.cachestoreonly=false` configuration key, which the installation program sets.

The following keys define the server associated with the file as a Branch Office Caching Services server:

- `jms.url`
- `java.naming.factory.url.pkgs`
- `jms.connection.factory`
- `jms.queue.name`
- `jndi.factory`

3.5.4.2 Configuration keys that to be modified only as a part of Branch Office Caching Services reconfiguration

Modify the following keys only as part of Branch Office Caching Services reconfiguration:

- `bocs.keystore`
- `bocs.configuration.name`
- `bocs.pulling.mode.enabled`
- `dms.pulling.url`
- `dms.server.base.urls`
- `bocs.pulling.interval`
- `proxy.host`
- `proxy.port`
- `policy`
- `policy.1`

3.5.4.3 Configuration keys that you can change

You can change the following configuration keys that the installation program sets:

- `cache.store.root`

This key identifies the root cache directory. The Branch Office Caching Services server stores cached content in this directory. The default name of the directory created by the installation program is `acsCache`.

- `cache.store.quota`

This key specifies the size of the root cache directory. The default value is 1 GB.

- `primary.content.store.root`
- `primary.content.store.quota`
- `tracing.enabled`
- `mode.debug`

3.5.5 Adding or modifying a root cache directory

The Branch Office Caching Services server caches content requested by users in the root cache directories. Root cache directories are identified in the `cache.store.root` key or keys in the file.

You can add additional root cache directory locations or change the size of a root cache. The key that identifies a root cache directory is `cache.store.root`. The key that defines the size of a root cache directory is `cache.store.quota`.

If you add additional cache directories, be sure to define their sizes also. The size specified in the `cache.store.quota` key is applied to the directory identified in the matching `cache.store.root` key. For example, if you add a directory specified in `cache.store.root.1`, the size of that directory is set in `cache.store.quota.1`.

3.5.6 Defining the parked content directory

If the Branch Office Caching Services server uses asynchronous write operations, define the location and maximum size of the directory that stores the parked content. This directory is not configured by default when the Branch Office Caching Services server is installed.

To define the directory's location, set the following key in the Branch Office Caching Services server's `acs.properties` file:

```
primary.content.store.root=<fully_qualified_path>
```

where `<fully_qualified_path>` is the path to the directory location where you want to store parked content. This directory must not be a subdirectory of the directory specified in the `cache.store.root` key.

The maximum size of the directory is defined in the following key:

```
primary.content.store.quota
```

3.5.7 Configuring cache housekeeping

The housekeeper runs once a day by default. Additionally, if the amount of content in the cache reaches 80% of the cache's configured capacity, the housekeeper runs outside of the scheduled execution to purge content to free space in the cache.

There are several keys in the `acs.properties` file that control housekeeping for the Branch Office Caching Services server. The values in these keys are applied to the content in all cache root directories. The following table describes the keys:

Table 3-3: `acs.properties` keys controlling Branch Office Caching Services cache housekeeping

Parameter	Description
<code>cache.schedule.housekeeper</code>	Controls how often the housekeeping process runs by default to find and remove obsolete files from the cache. The value is interpreted as seconds. The default value is 24*3600 (once a day).
<code>cache.schedule.housekeeper.start_delay</code>	Defines the interval between the startup of the Branch Office Caching Services server and the first run of the housekeeper. The value is interpreted in seconds. The default value is 5*60 (5 minutes)
<code>cache.schedule.housekeeper.retire_interval</code>	Defines how long a cached, content that has not been accessed may be in the cache before being considered obsolete. The value is interpreted as seconds and represents an interval counted from the last time the cached content was accessed by a user. If the specified number of seconds passes and the content has not been accessed within that interval, the housekeeper considers the content as obsolete and removes it from the cache. The default value is 30*24*3600 (30 days)

3.5.8 Configuring consistency checks

The system periodically validates the consistency of content in the cache. There are two keys that control consistency checks. The following table describes the keys:

Table 3-4: acs.properties keys controlling cache consistency checking

Parameter	Description
cache.schedule.consistency	Controls how often the consistency of the cache is validated. The value is interpreted as seconds. The default value is 24*3600 (once a day).
cache.schedule.consistency.retire_interval	Defines the interval between the startup of the Branch Office Caching Services server and the first run consistency check. The value is interpreted in seconds. The default value is 5*60 (5 minutes).

3.5.9 Configuring cache write intervals

The Branch Office Caching Services server keeps in memory information about available on Branch Office Caching Services content. By default, the Branch Office Caching Services server saves this information on a disk at specified intervals. The default interval is every 5 minutes. To change that interval, specify a number of seconds in the following key:

```
cache.schedule.flush_directory
```

3.5.10 Configuring use of content retrieval URLs

There are two keys that control how a Branch Office Caching Services server handles the URLs used to retrieve content files. The following table describes the parameters:

Table 3-5: acs.properties keys controlling URL use

Parameter	Description
repository.validation.delta	<p>Defines the length of time for which a URL for content retrieval is valid. The interval is counted from the time the URL was generated.</p> <p>The value is interpreted as minutes. The default value is 360 minutes (6 hours).</p> <p>Each Branch Office Caching Services server is loaded differently. Some Branch Office Caching Services server may need to process more URLs for which URL expiration time needs to be longer than 360 minutes (6 hours). Some Branch Office Caching Services server may have less URLs for which there is no need to have longer expiration time since it can process the URLs within that time.</p> <p>Branch Office Caching Services defines the URL expiration time on its own since defining the URL expiration time by Accelerated Content Services of Documentum CM Server globally affects all the Branch Office Caching Services servers connected to it. This parameter is implemented for Branch Office Caching Services, irrespective of whether it is a pre-cache job or any other jobs or Branch Office Caching Services acting as a client or server.</p>
validate.certificate	<p>Controls whether the Branch Office Caching Services server validates the public key certificate used for URL validation against the list of trusted certificates.</p> <p>The default value is <code>false</code>, meaning that the public key certificate is not validated.</p>

3.5.11 Log files

The following log files are maintained in a Branch Office Caching Services installation:

- An application server log file

The application server log file (`catalina<timestamp>.log`) records application server errors and is located as follows:

- Windows: `%DOCUMENTUM%\tomcat\logs`
- Linux: `$DOCUMENTUM/tomcat/logs`

- Branch Office Caching Services specific Foundation Java API log file

The Branch Office Caching Services specific log file records Foundation Java API information operations related to the Branch Office Caching Services server.

The default name and location of the file is:

- Windows: %DOCUMENTUM%\tomcat\logs\bocs_trace.log
- Linux: \$DOCUMENTUM/tomcat/logs/bocs_trace.log



Note: The name and location of the file are defined in `log4j2.properties`.

- A log file for all Foundation Java API log messages

By default, the name of the log file that records all Foundation Java API log messages is `log4j2.log`. The default location of the file is:

- Windows: %DOCUMENTUM%\tomcat\logs\bocs_trace.log
- Linux: \$DOCUMENTUM/tomcat/logs/bocs_trace.log



Note: The name and location of the file are defined in the `log4j2.properties` file. The default size for this file is 100 KB.

- A log file recording all messages sent and received by Branch Office Caching Services

The `access.log` file records information about messages sent, received, and pulled by a Branch Office Caching Services server. The entries in the file have the following format:

```
128.222.102.204 - - [04/Jun/2007:18:31:11 -0400]
"POST /bocs/servlet/ACS HTTP/1.1" 200 296 128.222.102.204
- - [04/Jun/2007:18:31:32 -0400]
"GET /bocs/servlet/Handshake HTTP/1.1" 200 19 128.222.102.204
- - [04/Jun/2007:18:31:32 -0400]
"POST /bocs/servlet/ACS HTTP/1.1" 200 60
```



Note: Entries are line-wrapped in the documentation only. They appear on one line in the file.

Each entry records the following:

- The IP address that originated the request
- The date on which the request was processed
- The request itself
- The return status
- The length of the reply, in bytes.

The return status is either 200, representing a good status, or 500, representing an error status.

3.5.11.1 Log file size and backups

The maximum size of the application server and Foundation Java API log files is defined in the `log4j2.properties` file, in the `MaxFileSize` entry for each log file. You can set the size by setting the `MaxFileSize` key for the log file.

Log files are not backed up by default. Instead, when the file reaches its maximum size, the system overwrites the file. If you want to save backups of the log files, set the `MaxBackupIndex` key in the `log4j2.properties` file for the specific file. If that is set, when a log file reaches its specified maximum file size, it is backed up and another file is started. The name of the backup file is `<log_file_name>.log.N`, where `<N>` starts with 0 and increments by 1 with each backup. For example, `AcsServer.log.0`, `AcsServer.log.1`, and so on. The first backup of the log file is numbered 0, the second is numbered 1, the third is numbered 2, and so on. This means that the lower the number on the backup file, the older the file is.

The value set in `MaxBackupIndex` determines the number of saved backup files.

3.5.12 Reconfiguring a push Branch Office Caching Services server to pull mode

1. Log on to Documentum Administrator and connect to the global registry.
2. Access the `acs.properties` file, through the Branch Office Caching Services resource agent, to:
 - a. Check the file to make sure that the Branch Office Caching Services configuration name is set:

```
bocs.configuration.name=<name_of_bocs_config_object>
```

The certificate required for pull mode is associated with the Branch Office Caching Services server through the name of the Branch Office Caching Services configuration object. The name in `bocs.configuration.name` must match the name identified in the certificate.

- b. Set the mode to pull.

```
bocs.pulling.mode=TRUE
```

- c. Define the Messaging Service server's consume URLs as follows:

```
dms.pulling.url=BOCS,<DMS_URL_1>,<DMS_URL_2> dms.server.base.urls=BOCS,<DMS_URL_1>,<DMS_URL_2>
```

Enable Messaging Service high-availability by specifying multiple Messaging Service server URLs in a comma-delimited format (that is, `<DMS_URL_1>` and `<DMS_URL_2>`).

3. Run the following script to create the Branch Office Caching Services certificate:
 - Windows:


```
<user_directory>\bocs\bin\generateCert.bat <certificate_file_path>
```
 - Linux:

```
$DOCUMENTUM/bocs/bin/generateCert.sh <certificate_file_path>
```

where *<certificate_file_path>* is the location where you want to store the certificate.

4. Restart the Branch Office Caching Services server.
5. Use Documentum Administrator to modify the Branch Office Caching Services configuration object identified in `bocs.configuration.name` in the `acs.properties` file.
 - a. Change the mode from push to pull.
This setting is on the **Security** tab of the **BOCS Server Configuration Properties** page.
 - b. Upload the Branch Office Caching Services certificate you generated to the Branch Office Caching Services configuration object.
6. Ensure that the global registry connection for the Messaging Service is correctly set.

3.5.13 Enabling Accelerated Content Services/Branch Office Caching Services/Messaging Service with forward and reverse proxy

Proxy servers are integrated into the Distributed Content architecture to provide additional security to deployments. They can also provide caching benefits and traffic monitoring.

There are two types of proxy servers:

- Forward proxy: Acts as an intermediary between browsers in a client subnet and the outside world. HTTP requests from the clients are redirected through the forward proxy so that outside of the client subnet, all requests are seen as coming from the proxy.
- Reverse proxy: Provides protection to the backend environments. They work in parallel with firewall configuration so that only the reverse proxy host has access to the backend environment. HTTP requests from the outside world are intercepted by the reverse proxy server and passed to the backend servers.

The configuration information described in this section is for reference purposes only.

If you are using the Apache Web Server, then the Apache Web Server acts as the forward and reverse proxy server. Refer to the appropriate third-party vendor documentation for additional information.

Forward proxy

For example, consider an environment that has one client machine accessing the Distributed Content applications and a Branch Office Caching Services server in

the branch office using Apache Web Server as a forward proxy server. Branch Office Caching Services and the client machine that consumes Distributed Content need to be configured to send requests to the backend servers using the forward proxy server.

1. Configure Apache Web Server as a forward proxy server.

Modify the `httpd.conf` file or any similar configuration file to update the information for the client machine that consumes Distributed Content to use the forward proxy as follows:

```
ProxyRequests On
ProxyVia On
<Proxy *>
Order deny,allow
Deny from all
Allow from <Host name or IP address of the client machine accessing the Branch
Office Caching Services server>
</Proxy>
```

2. Restart the Apache Server.

3. Configure the client machine that consumes Distributed Content.

Set the forward proxy server as the proxy server in the client machine that consumes Distributed Content.

4. Configure Branch Office Caching Services to use the forward proxy server.

If Messaging Service can directly access Branch Office Caching Services, push or pull mode is used. In this example, Branch Office Caching Services is configured in pull mode. In pull mode, Branch Office Caching Services poll the Messaging Service server periodically for any messages.

Specify the forward proxy server host and port while installing Branch Office Caching Services. The forward proxy server information is stored in the `acs.properties` file as follows:

```
proxy.host=<forward proxy host>
proxy.port=<forward proxy port>
```

Reverse proxy

For example, consider an environment that has Documentum CM Server/Accelerated Content Services, Messaging Service, and client that consumes Distributed Content. Install Messaging Service in a location with direct access to the primary Documentum CM Server/Accelerated Content Services.

To configure Accelerated Content Services, Messaging Service, and client that consumes Distributed Content with supported reverse proxy servers so that client requests are redirected through the proxy server to the Accelerated Content Services, Messaging Service, or application servers of clients, do the following:

1. Configure the reverse proxy server for redirection.

Configure Apache Web Server as a reverse proxy to Accelerated Content Services and Messaging Service.

- a. Modify the `httpd.conf` file or any similar configuration file to update the information for the client that consumes Distributed Content to use the reverse proxy redirect rules as follows:
 - Enable all appropriate `LoadModule` directives for proxies.
 - Setup proxy rules.
- b. Add the directives for the Messaging Service server.

Add the following directives to map `http://<reverse proxy host>/dms-ws` to `http://<Messaging Service server host>:<port>/dms-ws` as follows:

```
ProxyPass /dms-ws http://<host name>:<port>/dms-ws
ProxyPassReverse /dms-ws http://<host name>:<port>/dms-ws
```

- c. Configure the Branch Office Caching Services server to access the Messaging Service server with reverse proxy in pull mode.

Update the following entries in the `acs.properties` file of the Branch Office Caching Services server:

```
dms.pulling.url=http\://<reverse proxy host name>\:<port>
dms.server.base.url=http\://<reverse proxy host name>\:<port>
```

2. Configure the Accelerated Content Services server.

- a. In an on-premises environment with multiple Accelerated Content Services servers, configure the `acs config` object with reverse proxy URL as follows:

```
#acs config object1:
http://<reverse proxy host name>:<port>/ACS1

#acs config object2:
http://<reverse proxy host name>:<port>/ACS2
```

Modify the `httpd.conf` file or any similar configuration file to map ACS1 URL and ACS2 URL as follows:

```
ProxyPass /ACS1 http://<ACS1 server host name>:<port>/ACS
ProxyReversePass /ACS1 http://<ACS1 server host name>:<port>/ACS
ProxyPass /ACS2 http://<ACS2 server host name>:<port>/ACS
ProxyReversePass /ACS2 http://<ACS2 server host name>:<port>/ACS
```

- b. In a cloud environment, configure all the `acs config` objects with reverse proxy URL as follows:

```
#acs config object1:
http://<reverse proxy host name>:<port>/ACS

#acs config object2:
http://<reverse proxy host name>:<port>/ACS
```

Modify the `httpd.conf` file or any similar configuration file to map ACS URL as follows:

```
ProxyPass /ACS https://<host name of ACS ingress>/ACS
ProxyReversePass /ACS https://<host name of ACS ingress>/ACS
```

- c. For all the `acs config` objects in both on-premises and cloud environments, add a new URL with `s3` as protocol with `http` only including `localhost` as follows:

```
http://localhost:9080/ACS/servlet/ACS
```

3. Configure the Messaging Service server.

Update the following entry in the `dms.properties` file of the Messaging Service server:

```
dms.webservice.update.url=http://<reverse proxy host name>\:<port>
```

3.5.14 Enabling Branch Office Caching Services access from behind a firewall

For enabling Branch Office Caching Services access from behind a firewall, the following are the assumptions:

- Branch Office Caching Services is on a host named `dctm_bocs01` and the port 8086 is used.
- An Apache HTTP server is used as the reverse proxy and the host name is `proxy01`. HTTP is used as the communication protocol.

Add the following directives to `httpd.conf` of the Apache HTTP server to map `http://proxy01/bocs` to `http://dctm_bocs01:8086/bocs` and `http://proxy01/bocs-ws` to `http://dctm_bocs01:8086/bocs-ws`:

```
ProxyPass /bocs http://dctm_bocs01:8086/bocs ProxyPassReverse
/bocs http://dctm_bocs01:8086/ bocs ProxyPass
/bocs-ws http://dctm_bocs01:8086
/bocs-ws ProxyPassReverse
/bocs-ws http://dctm_bocs01:8086/bocs-ws
```

Set the Branch Office Caching Services URL to `http://proxy01/bocs/servlet/ACS`.



Notes

- If you access Branch Office Caching Services through a firewall and reverse proxy, some additional configuration is required.
- In this example, an Apache HTTP server is used and the instructions can only be used as a reference if other reverse servers are used.

3.5.15 Enabling or disabling the write mode

You can configure a Branch Office Caching Services server to use both asynchronous and synchronous write or only synchronous write or neither mode.

Use Documentum Administrator to set the properties in the Branch Office Caching Services configuration object and content transfer configuration object and enable messaging in Messaging Service.

To set write modes for a Branch Office Caching Services server:

1. In the Branch Office Caching Services server properties page, modify the content access to specify the content write access you want to allow for the Branch Office Caching Services server.
In the repository, this information is recorded in the Branch Office Caching Services configuration object, in the `rw_capability` property.
2. In the **Distributed Transfer Settings Properties** page, set Accelerated Content Services Write to allow read and write capabilities.
In the repository, this information is recorded in the content transfer configuration object, in the `acs_write_mode` property.
3. In the **Messaging Server Configuration Properties** page, enable messaging.
4. Update the `app.xml` file to enable Branch Office Caching Services write functionality as follows:

```
<accelerated-write>
    <!-- when set to be "true", enables ACS, and, when optimal, BOCS write operations
        -->
    <enabled>true</enabled>
        <!-- when set to:
            1. "prohibit-async" - application doesn't allow BOCS write asynchronously
            2. "default-sync" - application should see content written from BOCS to ACS synchronously by default
            3. "default-async" - application should see content written from BOCS to ACS asynchronously by default
        -->
        <bocs-write-mode>default-async</bocs-write-mode>
        <!-- when set to "true", allows override of default BOCS write mode by users
            in general, it should be scoped with <filter> tag, e.g. role-based
        -->
        <allow-override-bocs-write-mode>true</allow-override-bocs-write-mode>
</accelerated-write>
```

5. Restart the application server.

3.5.16 Domain Name System requirement for web-based client hosts in distributed environment

In a configuration that includes Accelerated Content Services or Branch Office Caching Services servers, the machines hosting the web browsers must be able to resolve the base URLs defined for the Accelerated Content Services or Branch Office Caching Services servers using Domain Name System (DNS). The base URLs for the servers are recorded in the Accelerated Content Services configuration object for the server. You can modify it using Documentum Administrator. Alternatively, you can use the Documentum Query Language to modify the base URLs after they are set.

3.5.17 Creating a Branch Office Caching Services configuration object

After you install Branch Office Caching Services or upgrade an existing installation, create a Branch Office Caching Services configuration object for the Branch Office Caching Services server. For more information about using Documentum Administrator to create the Branch Office Caching Services configuration object, see *OpenText Documentum Content Management - Administrator User Guide* (EDCAC250400-UGD).

3.5.18 Configuring security for Branch Office Caching Services servers in pull mode

When you install or upgrade Branch Office Caching Services, the installation program generates a public key. A Branch Office Caching Services server in pull mode uses that key to generate a digital signature. It identifies itself with this signature when it contacts a Messaging Service server to pull messages from the Messaging Service server's queue.

If you enable pull mode during Branch Office Caching Services installation, the installation program provides the path to the public key certificate. When you create the Branch Office Caching Services configuration object, provide the file path for that key. The location of the public key is recorded in the Branch Office Caching Services configuration object.

3.5.19 Configuring Branch Office Caching Services server to view files in Smart View

To view files in the Branch Office Caching Services enabled Smart View, perform the following tasks:

1. Update the existing `web.xml` file in the Branch Office Caching Services installation machine (for example, `C:\Documentum\tomcat\webapps\bocs\WEB-INF`) with the following entries as follows:

```
<filter>
<filter-name>CORSFilter</filter-name>
<filter-class>com.documentum.osgi.filter.CORSFilter</filter-class>
<init-param>
<param-name>CORSAllowed</param-name>
<param-value>true</param-value>
</init-param>
<init-param>
<param-name>AllowedHeaders</param-name>
<param-value>documentum-custom-unauth-scheme,x-d2-client-type,x-d2-network-
location</param-value>
</init-param>
</filter>
```

2. Save the file.
3. Start the Branch Office Caching Services server in the console mode.

3.5.20 Starting and stopping Branch Office Caching Services server

- Windows: Branch Office Caching Services servers are installed as Windows services. To start and stop Branch Office Caching Services servers, use the **Services** dialog box. The service name is Documentum Branch Office Caching Services.
- Linux: The internal database must be running before you start Branch Office Caching Services. However, if the database is not running, no error is generated. Use scripts to start and stop Branch Office Caching Services and to start and stop the internal database. If you stop Branch Office Caching Services, you do not need to stop the internal database.
 - To start, run the \$DOCUMENTUM/tomcat/bin/startBOCS.sh command.
 - To stop, run the \$DOCUMENTUM/tomcat/bin/stopBOCS.sh command.

3.6 Uninstalling Branch Office Caching Services server

To uninstall Branch Office Caching Services, remove the **Documentum Branch Office Caching Services** installed service.

To uninstall from Windows:

1. Log on to the Branch Office Caching Services server host as the installation owner.
2. Use the Windows **Service** to stop the Branch Office Caching Services server.
3. Use the **Control Panel** to uninstall the Branch Office Caching Services service.

To uninstall from Linux:

1. Log on to the Branch Office Caching Services server host as the installation owner.
2. Run the \$DOCUMENTUM/uninstall/bocs/Uninstall command.

3.7 Upgrading Branch Office Caching Services server

For information, see *OpenText Documentum Content Management - On-Premises Upgrade and Migration Guide (EDCCS250400-UMD)*.

3.8 Installing and configuring Messaging Service servers

3.8.1 Overview

You deploy a Messaging Service server to an application server. You can deploy a Messaging Service server to either a Documentum CM Server host or by itself on a different one. The Messaging Service database is configured in the global repository's database when a new global repository is configured or old global repository is upgraded to 7.x. The Messaging Service database consists of additional tables for storing messages. To set up high availability for Messaging Service, you install multiple Messaging Service servers on separate hosts (including VMs) and specify their URLs from Branch Office Caching Services servers.

3.8.2 Prerequisites

- You must have created a global registry repository and the Messaging Service server must be able to access the global registry database and the port must be open.



Note: The Messaging Service database user account is configured as follows:

– User name: dms

– Password: same as the global repository user password

- On the host that you want to install a Messaging Service server, you must have already installed the Documentum CM Server binaries. You do not have to create a repository, that is, when running the Documentum CM Server installer, do not select to install a repository.
- A Messaging Service server and Documentum CM Server instance must be the same version.
- A Messaging Service server must have HTTP access to all Branch Office Caching Services servers in push mode (servers in push mode receive messages from the Messaging Service server).
- To set up high availability properly, it is recommended that you install only one Messaging Service server per host.
- A mixed mode where some components are in anonymous and some in certificate-based is not supported.

- Set the value of the `VALIDATE_SIGNATURE` environment variable in the Messaging Service machine to enable or disable the signature validation for all the requests. By default, the value is `false`.

3.8.3 Installing Messaging Service server

1. If you want to use HashiCorp Vault, perform all the tasks as described in “Configuring Documentum Secret Integration Service” on page 49 and make sure that DSIS is running.
2. Start the Documentum CM Server configuration program as follows:
 - Windows: Select **Start > All Programs > Documentum > Documentum Server Manager** and on the **Utilities** tab, click **Server Configuration**, and then select **Documentum Messaging Service (DMS)**.
 - Linux: Run `$DM_HOME/install/Server_Configuration_Program.sh` and then select **Documentum Messaging Service (DMS)**.
3. If you want to configure the HashiCorp Vault secrets, on the **Vault configuration** page, select the **Enable Vault** check box.
4. If you select the **Enable Vault** check box, provide the following information and click **Next**:
 - a. **DSIS URL:** Provide a value in the following format:
`http://localhost:<port mentioned in application.properties>/dsis`
 - b. **DSIS Token:** Provide the `dsis.dctm.token` token value as described in “Configuring Documentum Secret Integration Service” on page 49.

! **Important**

If you enabled the HashiCorp Vault configuration, the installer retrieves all the password information automatically from the HashiCorp Vault server. Make sure that you have stored the following secrets in the HashiCorp Vault server:

Table 3-6: Secret and key name

Secret name	Key name	Description
<code>DOCBASE_PASSWORD</code>	<code><database user name></code>	Repository password. For example, <code>DOCBASE_PASSWORD/databasetestuser</code> .
<code>DCTM_DMS_JMX_PASSWORD</code>	<code><hostname></code>	Tomcat administrator owner password. For example, <code>DCTM_DMS_JMX_PASSWORD/host1.abclab.net</code> .



Note: The host name must be a fully qualified domain name.

5. Provide the installation owner password and click **Next**.
 - If HashiCorp Vault is enabled, then the installation owner password is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, then type the installation owner password.
6. Provide the administrator user password and click **Next**. The default port is 8086.
 - If HashiCorp Vault is enabled, the administrator user password is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, type the administrator user password.
7. Select **Use certificates** if Documentum CM Server and global repository is already configured with SSL certificates, provide the Foundation Java API trust store information, and then click **Next**.
 - **TrustStore:** Location of the Foundation Java API trust store. For example, \$DOCUMENTUM\dba\secure\dfc.keystore.
 - **Password:** Password of the trust store file.
 - If HashiCorp Vault is enabled, then the trust store password is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, then type the trust store password.
 - Select **Use Default Java TrustStore** if you want to use the default Foundation Java API Java trust store.
8. Select the global registries for the Messaging Service client deployment and click **Next**.
9. Provide the global registry login name and password and click **Next**.
 - If HashiCorp Vault is enabled, then global registry password is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, then type the global registry password.
10. Provide the database information and click **Next**.
 - **Database type:** Database type used for the Messaging Service installation from the list.
 - **Database host name:** Database host name on which the global registry database system resides.

- **Database listener port:** Port at which the global registry database system listens for requests. The default port is 1433 for Microsoft SQL Server, 1521 for Oracle, and 5432 for PostgreSQL.
- **DMS user name:** Global registry database user name.
- **DMS password:** Global registry database owner password.
 - If HashiCorp Vault is enabled, then global registry database owner password is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, then type the global registry database owner password.
- **Database name:** (Only for SQL Server and PostgreSQL) Name of the database created for the global registry repository. The name is specified in the *database_name* property in the *server.ini* file.
- **Database instance name:** Name of the instance if the database is installed with a named instance.
- **Repository owner name:** Repository owner name of the database table. The name is specified in the *server.ini* file.
- **Oracle SID:** (Only for Oracle) Provide the Oracle system ID.

Table 3-7: Messaging Service default directories

Directory	Description	Default directory
Installation directory	Root directory of the Messaging Service web application.	Windows: %DOCUMENTUM%\dms_tomcat\webapps Linux: \$DOCUMENTUM/dms_tomcat/webapps
Properties files directory	The installation process configures <i>dms.properties</i> automatically. See <i>dms-full.properties</i> for a description of the properties in <i>dms.properties</i> . You use Documentum Administrator to configure a Messaging Service server (these configuration changes are automatically made to <i>dms.properties</i>). <i>log4j2.properties</i> contains logging properties for Messaging Service.	Windows: %DOCUMENTUM%\dms_tomcat\webapps\dms-ws\WEB-INF\classes Linux: \$DOCUMENTUM/dms_tomcat/webapps/dms-ws/WEB-INF/classes

11. Click **Finish**.

3.8.4 Post-installation task

3.8.4.1 Licensing OpenText Documentum CM

OpenText Documentum CM uses OpenText Directory Services (OTDS) to apply licenses for all the OpenText Documentum CM components. For more information about procuring the license file and configuring OTDS and license, see “[Licensing OpenText Documentum CM](#)” on page 86.

3.8.5 Configuring Messaging Service server

Use Documentum Administrator to:

- Add and delete Messaging Service server configurations for a repository.
- Enable and disable messaging.
- Configure Branch Office Caching Services server message routing.



Note: Defaults are set for the timing of the pushing of messages to Branch Office Caching Services servers. Change the timing in `dms.properties`. You cannot use Documentum Administrator to change the timing.

- Access the resource agent. The file is administered using Java Management Extensions (JMX), accessed through the Messaging Service resource agent in Documentum Administrator. The resource agent is accessed through the Resource Management node. The URL for the JMX resource agent is as follows:

```
service:jmx:rmi:///jndi/rmi://<host_name>:<port>/dms
```

where:

- `<host_name>` is the name or IP address for the host computer.
- `<port>` is the assigned port. The default port is 8490.

If you want to connect to the Messaging Service administration resource agent from outside of a firewall, configure the firewall settings to allow the RMI protocol for the port.

- Register Messaging Service as a privileged client if you need multiple repository support to use digital signature validation.

For more information, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)* or *OpenText Documentum Content Management - Server System Object Reference Guide (EDCCS250400-ORD)*.

3.8.5.1 Timing the pushing of messages to Branch Office Caching Services servers

If a Branch Office Caching Services server is unreachable when messages are pushed to it, then the Messaging Service server waits for a specific amount of time before resending messages. The `dms.properties` file's `dms.job.push.scheduler.delay.milliseconds` property specifies this wait time. The default value is 10 minutes.

Messaging Service pushes messages to a Branch Office Caching Services server at regular intervals. The `dms.properties` file's `dms.job.push.scheduler.delay.milliseconds` property specifies this interval. The default value is 1 minute.

3.8.5.2 Best practices

Quality improvements have been made in the area of Messaging Service 7.0 high-availability (HA). For example, the CXF framework is used to deliver messages. The CXF framework is faster, simpler, and more scalable than prior methods used in delivering messages. To make Messaging Service and Messaging Service HA perform better, apply the best practices described in the following sections.



Note: The best practices and/or test results are derived or obtained after testing the product in our testing environment. Every effort is made to simulate common customer usage scenarios during performance testing, but actual performance results will vary due to differences in hardware and software configurations, data, and other variables.

3.8.5.2.1 Database settings for the Oracle-based Messaging Service database

- For the Messaging Service database, follow the general database guidelines provided in the *OpenText Documentum Content Management - Server Administration and Configuration Guide* (EDCCS250400-AGD).
- For large or rapid volumes of asynchronous writes and pre-caching activities, it is recommended that you monitor the database logs and statistics reports, so that the database does not become a Messaging Service performance bottleneck.

3.8.5.2.2 Messaging Service configuration

- The performance of Messaging Service/Branch Office Caching Services message processing is dependent on the number of messages and is not related to the size of the cached data.
- For information about settings related to performance, see `dms-full.properties`.
- For rapid ingestions, set the push delay to 1 second. This setting can help push messages to Branch Office Caching Services as soon as possible. For example:

```
dms.job.push.scheduler.delay.milliseconds = 1000
```

However, you can also set the delay for longer periods such as 1 minutes or 2 minutes because the delivery is not sensitive to time.

- For a single Messaging Service server, the maximum delivery rate can reach 50 – 70 messages per second when no messages are inserted. If messages are inserted, the delivery rate can drop to 25 message per second.
- To increase the delivery rate, adjust the following properties according to the `dms-full.properties` file:

```
# queue.size must be equal to or more than
dms.message.fetch.batch.size
# task must be less than queue size,
this is parallel task that consume the queue data.
dms.message.fetch.batch.size=100
#The task-size is the number of thread
dms.job.push.queue.size=100
dms.job.push.task.size=10
```

When a Messaging Service server is started, it creates a number of threads based on the `dms.push.task.count` parameter for pushing the messages to Branch Office Caching Services. However, one has to be careful in changing these values. For instance, setting the batch size to a higher value can prolong the database lock that is obtained for processing each batch.

- To improve Messaging Service message processing, the expired messages must be cleaned up from the database periodically. To delete messages, one can use the following queries:

```
- delete from dms_destination_message where processed_date
<time_in_milliseconds(currentdate - 180) and status_no = 2

- delete from dms_message where processed_date
<time_in_milliseconds(currentdate - 180) and status_no = 2 and
m.message_id in ( select message_id from dms_destination_message
where processed_date < time_in_milliseconds(currentdate - 180)
and status_no = 2)
```

- When pre-caching or asynchronous write jobs are scheduled very frequently and/or concurrently, Messaging Service can become a bottleneck because a flood of message queues can come in at a faster rate (if there are not enough Messaging Service nodes and Branch Office Caching Services servers to handle the load). If possible, run these jobs at times of low user activity.
- By default, only the primary node in Messaging Service HA configuration can receive messages from Messaging Service clients such as Java Method Server or Branch Office Caching Services. When the primary node goes down, activities fail over without loss of messages from the primary node to other nodes. However, in Messaging Service HA, multiple nodes can be active in delivering messages to Branch Office Caching Services, which increases the message-delivery throughput to Branch Office Caching Services.

3.8.5.2.3 Branch Office Caching Services configuration

- To allow the transfer of a high volume of documents, set the `cache.paging.page_size` or `thecache.paging.max_count` parameter to a large value. For example:

```
cache.paging.page_size=10000
```
- A Branch Office Caching Services server has limitations when processing very large volumes of requests for content transfer. The unavailability of Branch Office Caching Services can make content failover to Accelerated Content Services, transparent to the user. Even 10 requests per second can cause such a failover to Accelerated Content Services. Therefore, it is recommended to add more Branch Office Caching Services servers in rapid ingestion scenarios.
- To improve Branch Office Caching Services processing messages, you can increase the number of threads that write parked content to Accelerated Content Services (if the server can create request threads). The default value of the `bocs.asynch.executor.max.thread.count` parameter is 2.

3.8.6 Starting and stopping Messaging Service server

Use these instructions to start and stop the Messaging Service server, the database, and the application server.

Table 3-8: Starting and Stopping Messaging Service servers

Operating system	Starting Messaging Service	Stopping Messaging Service
Windows	In the Services dialog box, right-click Documentum Messaging Server and select Start .	In the Services dialog box, right-click Documentum Messaging Server and select Stop .
Linux	Run <code>\$DOCUMENTUM/dms_tomcat/bin/startDMS.sh</code>	Run <code>\$DOCUMENTUM/dms_tomcat/bin/stopDMS.sh</code>



Note: Warning messages are displayed when Messaging Service is started. For example: `WARN [org.jboss.as.ee] (MSC service thread 1-7) JBAS011006: Not installing optional component org.apache.cxf.transport.http.Servlet3ContinuationProvider$Servlet3Continuation due to exception.` However, you can ignore these warnings.

3.8.7 Uninstalling Messaging Service server

Make sure that you remove the corresponding Messaging Service server configuration from all repositories.

1. (Windows only) In the **Services** dialog box, right-click **Documentum Messaging Server** and select **Stop**.
2. Start the Documentum CM Server configuration program as follows:
 - Windows: Select **Start > All Programs > Documentum > Documentum Server Manager** and on the **Utilities** tab, click **Server Configuration**, and then select **Documentum Messaging Service (DMS)**.
 - Linux: Run `$DM_HOME/install/dm_launch_server_config_program.sh` and then select **Documentum Messaging Service (DMS)**.
3. Select **Delete DMS instance**.

3.9 Configuring Accelerated Content Services, Branch Office Caching Services, Messaging Service, and Tomcat for SSL connections

To configure SSL, perform the following steps:

1. Generate security certificates for all distributed components (Accelerated Content Services, Branch Office Caching Services, Messaging Service), Web Development Kit, and every Tomcat server on which the Accelerated Content Services, Branch Office Caching Services, and Messaging Service applications run.
2. Import those certificates into each of their keystores or cacerts of Java.
3. Enable SSL on all of their Tomcat application servers and Documentum CM Server.



Notes

- These instructions use self-signed security certificates as examples. Self-signed certificates are adequate for testing or when all applications and users reside within a secure environment. However, in a production or a non-secure environment, you should use certificates that have been verified by a certification authority (CA). The CA might have instructions on generating and verifying security certificates, which might also involve using the Java keytool.exe utility.
- To create a self-signed security certificate and keystore for each Accelerated Content Services, Branch Office Caching Services, Messaging Service, and Tomcat installation, you use the Java keytool.exe utility. The *Oracle* documentation contains more information about keytool.exe. keytool.exe is installed with Java (for example, `C:\Documentum\java64`).

- The *Apache* documentation contains more instructions about configuring SSL on Tomcat.

3.9.1 Generating and importing security certificates

- Create a key and a local certificate keystore for all Accelerated Content Services, Branch Office Caching Services, Messaging Service, and Web Development Kit Web applications as well as every Tomcat server on which the Accelerated Content Services, Branch Office Caching Services, and Messaging Service application run.

For example:

```
keytool -genkey -alias tomcat -keyalg RSA -keystore tomcat.keystore
```

or

```
keytool -genkeypair -alias myalias -keyalg RSA -keysize 2048 -keystore mykeystore.jks -storepass changeit -validity 365
```

You are prompted for additional information. These instructions use the following sample values:

Prompt/Parameter	Application	Sample value	More information
-alias	Accelerated Content Services	acskey	N/A
	Branch Office Caching Services	bocskey	
	Messaging Service	dmskey	
	Web Development Kit	wdkkey	
	Tomcat server	tomcat	
-keystore	Accelerated Content Services	acs.keystore	N/A
	Branch Office Caching Services	bocs.keystore	
	Messaging Service	dms.keystore	
	Web Development Kit	wdk.keystore	
	Tomcat server	tomcat.keystore	
What is your first and last name?	Accelerated Content Services Tomcat server	acs.machine. dnsname.com	Also known as the CN or common name. Specify the complete host name of the application server as referenced by your browser or web service consumer.
	Branch Office Caching Services Tomcat server	bocs.machine. dnsname.com	

Prompt/Parameter	Application	Sample value	More information
	Messaging Service Tomcat server	dmsmachine. dnsname.com	
	Web Development Kit	wdkmachine. dnsname.com	
Enter keystore password	All	changeit	N/A



Notes

- The keystores and certificates can be created from any supported Java (OpenText packaged or external).
- You should import the Messaging Service certificate in `dfc.keystore` of Documentum CM Server. For example, `%DOCUMENTUM%\dba\secure`.

2. Copy `tomcat.keystore` to `%DOCUMENTUM%\tomcat\conf`.

where `tomcat` is the name of the Accelerated Content Services, Branch Office Caching Services, or Messaging Service Web application.

3. Generate every Web application's certificate by exporting each one's keystore file. For example:

```
keytool -export -alias acskey -file acs.cer -keystore acs.keystore
```

These instructions use the following sample values:

Prompt/Parameter	Application	Sample value	More Information
<code>-file</code>	Accelerated Content Services	acs.cer	N/A
	Branch Office Caching Services	bocs.cer	
	Messaging Service	dms.cer	
	Web Development Kit	wdk.cer	

4. Now import each Web application's certificate into all running Java programs on the hosts with which the Web application needs to connect as follows:

Application certificate	Application machine
Accelerated Content Services	Accelerated Content Services, Branch Office Caching Services
Branch Office Caching Services	Branch Office Caching Services, Messaging Service
Messaging Service	Branch Office Caching Services, Messaging Service, Web Development Kit

For Branch Office Caching Services pre-caching job, you should import the Messaging Service certificate on Documentum CM Server. For example:

```
keytool -import -noprompt -trustcacerts -alias acscert -file  
"C:\certificate\dms.cer" -keystore  
"%JAVA_HOME%\lib\security\cacerts"
```

Notes

- On the Web Development Kit client machine, import this certificate to the Java installation where the Web Development Kit application is running.

For example:

```
keytool -import -noprompt -trustcacerts -alias acscert -file  
"C:\certificate\acs.cer" -keystore  
"%JAVA_HOME%\lib\security\cacerts"
```

- The default password for the cacerts Java trust store is the following:
`changeit`
- On the Accelerated Content Services and Branch Office Caching Services machines, the default location of the Java programs is \$DOCUMENTUM\java64.
- On a Web Development Kit client machine, the Java installation is located with the client application.

5. The Accelerated Content Services, Branch Office Caching Services and Web Development Kit client certificates—but not the Messaging Service certificate—must also be imported into the UCF client's Java and browser's Java program (where the Web Development Kit application URL is accessed) because if the UCF client's Java version is later than the existing Java on the host, then the UCF client's Java is installed during content transfer.



Note: The default location for UCF client's Java is JAVA_HOME, where the JAVA_HOME environment variable is pointing to the supported version of Java.

Use keytool.exe to import the Accelerated Content Services, Branch Office Caching Services, Web Development Kit certificates into the UCF client's Java and browser's Java. For example:

```
keytool -import -noprompt -trustcacerts -alias acscert -file  
"C:\certificate\acs.cer" -keystore  
"%JAVA_HOME%\lib\security\cacerts"
```

3.9.2 Configuring web applications for SSL

This section describes how to configure web applications for SSL.

3.9.2.1 Accelerated Content Services

1. Modify server.xml in \$CATALINA_BASE/conf/.

For example:

```
<Connector
    protocol="org.apache.coyote.http11.Http11NioProtocol"
    port="9082"
    maxThreads="150"
    SSLEnabled="true">
    <SSLHostConfig>
        <Certificate
            certificateKeystoreFile="<path of keystore file>/acs.keystore"
            certificateKeystorePassword="changeit"
            type="RSA"
        />
    </SSLHostConfig>
</Connector>
```

Provide the keystore file path for certificateKeystoreFile and password for certificateKeystorePassword which you have set for keystore.

2. Restart Accelerated Content Services.
3. To validate the SSL configuration, go to the following URL:

<https://acsmachine.dnsname.com:9082/ACS/servlet/ACS>

3.9.2.2 Branch Office Caching Services

1. Modify the server.xml in %DOCUMENTUM%\tomcat\conf\server.xml.

For example:

```
<Connector
    protocol="org.apache.coyote.http11.Http11NioProtocol"
    port="8088"
    maxThreads="150"
    SSLEnabled="true">
    <SSLHostConfig>
        <Certificate
            certificateKeystoreFile="<path of keystore file>/bocs.keystore"
            certificateKeystorePassword="changeit"
            type="RSA"
        />
    </SSLHostConfig>
</Connector>
```

Provide the keystore file path for certificateKeystoreFile and password for certificateKeystorePassword which you have set for keystore.

2. Restart Branch Office Caching Services.
3. To validate the SSL configuration, navigate to the following URL:

<https://bocsmachine.dnsname.com:8088/bocs/servlet/ACS>

3.9.2.3 Messaging Service

1. Modify the server.xml in %DOCUMENTUM%\dms_tomcat\conf\server.xml.

For example:

```
<Connector  
protocol="org.apache.coyote.http11.Http11NioProtocol"  
port="8491"  
maxThreads="150"  
SSLEnabled="true">  
<SSLHostConfig>  
<Certificate  
certificateKeystoreFile=<path of keystore file>/dms.keystore"  
certificateKeystorePassword="changeit"  
type="RSA"  
/>  
</SSLHostConfig>  
</Connector>
```

Provide the keystore file path for certificateKeystoreFile and password for certificateKeystorePassword which you have set for keystore.

2. In %DOCUMENTUM%\dms_tomcat\webapps\dms-ws\WEB-INF\classes\dms.properties, change the value of dms.webservice.update.url to https://dmsmachine.dnsname.com:8491.
3. Restart Messaging Service.
4. To validate the SSL configuration, navigate to the following URL:

<https://dmsmachine.dnsname.com:8491/dms-ws>

3.9.2.4 Web Development Kit client application server

1. Modify server.xml on the application server to enable SSL and the server to use the https protocol. Check for SSL configuration lines in the server.xml file, remove the comments and provide the appropriate details for the keystore file path, password, port, and so on.

For example, on Tomcat, remove the comment from the following lines in <app_server_home>\conf\server.xml:

```
<Connector  
protocol="org.apache.coyote.http11.Http11NioProtocol"  
port="8443"  
maxThreads="150"  
SSLEnabled="true">  
<SSLHostConfig>  
<Certificate  
certificateKeystoreFile=<path of keystore file>/wdk.keystore"  
certificateKeystorePassword="changeit"  
type="RSA"  
/>  
</SSLHostConfig>  
</Connector>
```

Provide the keystore file path for certificateKeystoreFile and password for certificateKeystorePassword which you have set for keystore.

2. Restart the application server.

3. To validate the SSL configuration, navigate to the following URL:

```
https://wdkmachine.dnsname.com:8443/da
```

where da represents the Documentum Administrator Web application.



Note: When performing an asynchronous write operation, Foundation Java API on the clients' (for example, Webtop) application server sends a message to Messaging Service which is routed to the appropriate Branch Office Caching Services to inform how to upload the document. If Messaging Service is configured in SSL mode, you should import the Messaging Service certificate in the clients' application servers' keystore.

3.9.2.5 Additional configurations for Accelerated Content Services/ Branch Office Caching Services/Messaging Service/Web Development Kit client

This section describes the additional configurations that are required only for asynchronous write operations.

Update dfc.properties for Accelerated Content Services, Branch Office Caching Services, Messaging Service, and Web Development Kit client as follows:

```
set dfc.security.ssl.use_existing_truststore=true
```

If you encounter any errors after setting the value of dfc.security.ssl.use_existing_truststore to true, then add the Documentum CM Server certificates (broker crt and server crt) to cacerts in Accelerated Content Services, Branch Office Caching Services, Messaging Service, and client machines:

```
keytool.exe -import -noprompt -trustcacerts -alias <new connection broker name> -file "C:\Documentum\dba\secure\broker.crt.der" -keystore "JAVA_HOME\lib\security\cacerts"

keytool.exe -import -noprompt -trustcacerts -alias <new server name> -file "C:\Documentum\dba\secure\server.crt.der" -keystore "JAVA_HOME\lib\security\cacerts"
```

3.9.3 Modify or configure Accelerated Content Services/ Branch Office Caching Services/Messaging Service configs

This section discusses modifying or configuring Accelerated Content Services/ Branch Office Caching Services/Messaging Service configs in the repository.

1. For the Accelerated Content Services server connection, specify the following attributes:

Attribute	Sample value
Protocol	https

Attribute	Sample value
Base URL	https://acsMachine.dnsname.com:9082/ACS/servlet/ACS

These attributes are available in the Accelerated Content Services Server config.

- For the Branch Office Caching Services server connection, specify the following attributes:

Attribute	Sample value
Protocol	https
Base URL	https://bocsMachine.dnsname.com:8088/bocs/servlet/ACS

These attributes are available in the Branch Office Caching Services Server config.

- For the Messaging Service server's Branch Office Caching Services message routing, specify the following attributes:

Attribute	Sample value
Post URL	https://dmsMachine.dnsname.com:8491
Consume URL	https://dmsMachine.dnsname.com:8491

These attributes are available in the Messaging Service Server config.

3.9.4 Troubleshooting SSL configuration

- Check the Accelerated Content Services, Branch Office Caching Services and Messaging Service server logs to make sure that no SSL-related errors or exceptions have occurred.
- Verify that you can access the Accelerated Content Services, Branch Office Caching Services, Messaging Service, and Web Development Kit client application URLs from every other host by using their full host names with the HTTPS protocol and SSL port. These URLs must be accessible from every host.
- Use keytool.exe to verify that each certificate has been imported into the trusted store. For example:

```
keytool -list -keystore "%JAVA_HOME%\lib\security\cacerts"
```

- If you encounter any SSL errors, then enable the SSL debug logs. Add the following in catalina.bat or any other configuration file depending on your operating system and web application server as follows:

```
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.debug=ssl:handshake"
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.debug=all"
```

- If you encounter any handshake failures with `no common ciphers` error, then add the missing ciphers to the ciphers list in the `server.xml` file of Accelerated Content Services, Branch Office Caching Services, Messaging Service, or Web Development Kit client web application server.



Note: For pre-caching and asynchronous jobs to work, make sure that you modify Java Method Server URL from HTTP to HTTPS in the `do_method`, `XMLStoreService`, and `do_mail` methods.

3.10 Installing and configuring remote Content Server in distributed or load-balanced configurations

This section provides instructions for installing and configuring remote Content Servers in distributed or load-balanced content configurations.

If you are creating a new, single repository in a distributed or load-balanced content configuration, a configuration program separate from the Documentum CM Server configuration program is used for installing remote Content Servers and creating the storage areas on the remote hosts and related location objects.

3.10.1 Prerequisites

The following requirements and limitations exist when you are installing remote Content Server in distributed or load-balanced configurations:

- The remote host must meet the same pre-installation requirements as the primary Documentum CM Server host.
- The remote Content Server must be installed in the same installation directory as the primary Documentum CM Server.
- The database client software must be installed on remote Content Server hosts. The remote Content Server configuration program must connect to the database to properly create the following objects for the remote server:
 - `server config`
 - `acs config`
 - `file store storage`
 - `location`
- When a remote Content Server is created for a distributed or load-balanced content environment, the `server.ini` file from the primary Documentum CM Server host is copied from the primary host to the remote host. The values used on the primary and remote hosts for database connectivity must be identical to make sure that the value of the `database_conn` key on the primary Documentum CM Server host is valid on the remote hosts. For example, if the database is SQL Server, make sure that the DSN name for the SQL Server instance's ODBC data source is the same on all hosts.

- All hosts in a distributed or load-balanced configuration must be set to the same UTC time.
- A repository that uses a distributed or load-balanced storage area with encrypted file stores as components cannot use shared content.



Caution

After a repository has been configured to use distributed or load-balanced storage, it is not possible to revert to using non-distributed storage.

- The following requirements must be met when the Documentum CM Server file store is assigned to a shared folder on the network with a UNC path:
 - Documentum CM Server and the file store must be on the same domain.
 - The installation user account of Documentum CM Server must be available on the domain.
 - The installation user account must have full access control for the file store.

3.10.2 Installing and configuring RCS

1. Copy the Documentum CM Server installation files from the installation media to the correct directories on the host.
This step is identical to the process used to copy files onto the primary Documentum CM Server host.
2. Install the Documentum CM Server software. At the end of the installation, do not select the **Configure Now** option and exit the installer.



Note: When installing remote Content Server, if the remote connection broker is configured for certificate-based mode, then manually copy all the certificate files to the remote Content Server machine.

3. Run one of the following files:
 - Windows: %DM_HOME%\install\cfsConfigurationProgram.exe
The configuration program starts automatically following a reboot of the host. However, if it does not start automatically or if you have to delete the remote Content Server and must reconfigure the remote Content Server, simply rerun it.
 - Linux: \$DM_HOME/install/dm_launch_cfs_server_config_program.sh
4. If you want to configure the HashiCorp Vault secrets, on the **Vault configuration** page, select the **Enable Vault** check box.
5. If you select the **Enable Vault** check box, provide the following information and click **Next:**

- a. **DSIS URL:** Provide a value in the following format:

```
http://localhost:<port mentioned in application.properties>/dsis
```

- b. **DSIS Token:** Provide the `dsis.dctm.token` token value as described in “Configuring Documentum Secret Integration Service” on page 49.

! **Important**

If you enabled the HashiCorp Vault configuration, the installer retrieves all the password information automatically from the HashiCorp Vault server. Make sure that you have stored all the required secrets as described in “Configuring Documentum CM Server to use HashiCorp Vault” on page 38.

6. (Only Windows) Provide the installation owner password and click **Next**.
 - If HashiCorp Vault is enabled, then the installation owner password is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, then type the installation owner password.

7. Specify the host name of the primary connection broker for the repository and verify or type the port number on which the connection broker listens.

The port defaults to 1489. If you are using the default port number, make sure that the next port number (1490) is available for use because two ports must be reserved for the connection broker.

Select **Use certificates** if primary Documentum CM Server and repository are configured with SSL certificates.

Provide the Foundation Java API trust store information:

- **TrustStore:** The location of the Foundation Java API trust store. Before that you need to copy the Foundation Java API trust store from primary Documentum CM Server to remote Content Server location.
- **Password:** The password of the trust store file.
 - If HashiCorp Vault is enabled, then the trust store password is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, then type the trust store password.

Select **Use Default Java TrustStore** if you want to use the default Foundation Java API Java trust store.

Click **Next**.

8. Specify the fully qualified domain name (FQDN) of the remote Content Server host.
9. Select the repository for which you are installing the remote Content Server and enter the primary Documentum CM Server installation owner and domain information.

The repository list is populated with repositories known to the connection broker for which you provided information in [step 8](#).

10. Type the name and port number for the connection broker on the current host, indicate whether connection broker startup following a system restart is automatic or manual.

The default values are Docbroker and 1489. If you are using the default port number, make sure that the next port number (1490) is available for use because the connection broker requires that two ports be reserved. The connection broker is started.

11. Accept the default location of the data directory or browse to a different location.

The data directory is where content files are stored in the repository.

12. Accept the default location of the share directory or browse to a different location.

The share directory is where clients, example code, and required libraries are stored.

13. Accept the default service name for the new remote Content Server or type a different name.

14. Provide the global registry login name and password.

- If HashiCorp Vault is enabled, then the global registry password is retrieved from the HashiCorp Vault server.
- If HashiCorp Vault is not enabled, then type the global registry password.



Note: If the primary server uses an LDAP Server to authenticate, copy the file Documentum\dba\config\\ldap_xxxxxxxxxxxxxxx.cnt from the primary to the remote server (xxxxxxxxxxxxxx represents the r_object_ID of the LDAP config object).

15. To start the application server instance that is running the Java Method Server and Documentum CM Server, perform one of the following actions:

- On Windows, restart after the installation.
- On distributed or load-balanced Linux configurations, use Documentum Administrator to set the Get method for each component of the distributed or load-balanced store to Surrogate Get.

16. If required, modify the dm_server_config object to specify only the app_server_name and app_server_uri entries that are relevant to the remote Content Server.



Notes

- Because the remote Content Server installation clones a copy of the dm_server_config object from the original repository, unnecessary attribute

values might have been copied over. For example, if an index agent and OpenText™ Documentum™ Content Management Process Engine are in the original repository, you might have entries for them that point to the original host machine on the remote host. Remove any of these attributes if they are not applicable to the new remote Content Server installation.

- Make sure that you enable the Trusted Server Privilege and Trusted Login for all the Documentum CM Server related DFC privileged clients using Documentum Administrator to avoid authentication issues in multi-Documentum CM Server environments.

3.10.3 Upgrading a distributed or load-balanced configuration

For information, see *OpenText Documentum Content Management - On-Premises Upgrade and Migration Guide (EDCCS250400-UMD)*.

3.10.4 Deleting remote Content Server

Use these instructions to delete a remote Content Server and its software installation in a distributed or load-balanced content environment. These instructions delete only the remote Content Server. They do not delete the repository or affect the primary Documentum CM Server for the repository.



Note: On Windows, do not use the Server Manager program to uninstall the server because the Server Manager program launches the configuration program for primary Documentum CM Servers (not remote Content Servers).

Before deleting the software installation, delete any connection brokers on the host.

1. Log in to the host as the Documentum CM Server installation owner.
2. Run the remote Content Server configuration program in one of the following locations:
 - Windows: %DM_HOME%\install\cfsConfigurationProgram.exe
 - Linux: \$DM_HOME/install/dm_launch_cfs_server_config_program.sh
3. (Only for Windows) Provide the installation owner password.
 - If HashiCorp Vault is enabled, then the installation owner password is retrieved from the HashiCorp Vault server.
 - If HashiCorp Vault is not enabled, then type the installation owner password.
4. Select the option to delete the remote Content Server.
5. To delete its server software installation, run the server uninstaller in one of the following locations:
 - Windows: %Documentum%_uninst\Server\uninstall.exe

- Linux: \$DOCUMENTUM/uninstall/server/uninstall/uninstall.bin

3.11 Implementing single-repository models

This section describes how to implement the building blocks most commonly used to create single-repository distributed configurations.

3.11.1 Implementing a distributed repository without a distributed storage area

Use the instructions in this section to implement the single-repository distributed model. This model is the preferred model for single-repository distributed configurations if users are accessing the repository using web-based clients.

To set up a distributed repository without a distributed storage area:

1. If not already installed, install the primary site.



Note: One repository in the installation must be designated as the global registry if you installing Branch Office Caching Services at the remote sites.

2. Install Messaging Service on a separate host at the primary site if you are installing Branch Office Caching Services at the remote sites.
3. To use a Branch Office Caching Services server at each remote site, install Branch Office Caching Services.
4. Define the network locations for the remote sites.

If the remote sites are accessing content through the Accelerated Content Services server at the primary site and you are not installing Branch Office Caching Services servers at the remote sites, defining network locations is not required.

Use Documentum Administrator to define network locations. You must be connected to the repository designated as the global registry to define network locations.

3.11.2 Installing with distributed storage areas

The information in this section is useful when setting up distributed configurations that use a distributed storage area and no OpenText Documentum CM installation is present. This section does not provide instructions for converting an existing installation. Following this procedure results in a single repository that uses a distributed storage area distributed across multiple geographical locations.



Caution

After a repository is set up with a distributed storage area and is using it, you cannot remove it and return to a standalone, non-distributed configuration.

There are three stages in the process of setting up a distributed storage area:

1. Planning
2. Set up the primary site (the site where the RDBMS resides)
3. Set up the remote sites

3.11.2.1 Planning

Before you begin installing the Documentum CM Servers at any site:

- Decide whether to share distributed content, replicate distributed content, or use a combination of both.
- Read and follow the guidelines in the next section to make sure that your environment is set up correctly.
- Use our guidelines for estimating the required disk space to make sure that you have enough disk space at the site.

3.11.2.1.1 Guidelines

To make sure that your distributed architecture works properly, follow these guidelines:

- The Documentum CM Server at each site (primary and remote) must be able to authenticate the user, using the same mechanism.

When a remote user logs on to a repository, the client sends two connection requests, one to the remote Content Server and one to the data server. Each server must be able to authenticate the user using the same authentication mechanism.

- If you intend to share content files among the component storage areas, the installation owner for all servers accessing the repository must be the same account at all sites.

Having the same installation owner at each site allows the Documentum CM Server at each site to access content files at the other sites. On Windows platforms, to meet this requirement, have a global domain for all sites and establish a global `dmadmin` account (or equivalent) in that domain. At each site, log on to the global `dmadmin` account when you install and configure the Documentum CM Server.

- On Windows, all machines running a Documentum CM Server for a particular repository have to be in the same domain.
- If the sites are not connected using NFS, method objects must be resolvable at all server sites .

In a distributed installation, the method commands defined by the `method_verb` property of the `method` object must exist at each server site.



Note: Some method objects might not have a full file system path defined (in the `method_verb` property) for the program they represent. For such programs to work correctly, the command executable must be found in the PATH definition for the user who is executing the command. If the `run_as_server` property for the method object is set to TRUE, the user executing the command is the installation owner. If `run_as_server` is set to FALSE, the user executing the command is the user who has issued the EXECUTE statement or the DO_METHOD administration method. (`run_as_server` is set to FALSE by default in the methods defined in the `headstart.ebs` file.)

3.11.2.1.2 Estimating disk space

Before you begin installing distributed sites, estimate how much disk space is required for content storage at each site.

The amount of space required at each distributed site depends on the following factors:

- Total number of bytes per document
- Total number of documents in the distributed repository
- Number of distributed sites
- Number of versions of each document that you intend to keep online
- Whether you intend to keep renditions of the documents
- Whether you intend to replicate each document to all sites

The formula for estimating disk space is:

$$\begin{aligned} & (\text{total \# of documents}) \times (\text{\# of sites}) \times (\text{bytes/document}) \times (\text{\# of versions}) \\ & = \text{total amount of disk space} \end{aligned}$$

3.11.2.1.2.1 Estimating document size

To estimate the total number of bytes per document, sum the following figures:

- Number of bytes for an average document
- Number of bytes for any renditions

3.11.2.1.2.2 An example of disk space calculations

To illustrate estimating disk space, assume the following:

- Your enterprise has three distributed sites
- Site 1 has 10,000 documents
- Sites 2 and 3 has 5,000 each
- You intend to index PDFText renditions of the documents
- Each document is an average of 10K bytes.

First, estimate the total number of bytes per document. Include in the total the estimated size of the document's content and renditions. For example, assume that each document has 10K of content and PDF and PDFText renditions. For a 10K document, the PDF rendition is approximately 8K and the PDFText rendition is approximately 6K. Sum these estimates to arrive at the estimated total number of bytes per document. In our example, the sum is 24K. Use this total in the disk space formula to determine the disk space needed at each site.

In this example, the three sites have a total of 20,000 documents at 24K per document. Three versions of each are kept online, with each replicated at each site:

20,000 docs x three sites x 24K/doc x three versions is approximately 4.68 gigabytes

This calculation indicates that you need a total of 4.68 gigabytes of disk space at each distributed site.

3.11.2.2 Setting up the sites

The procedure in this section describes how to install a distributed storage area as part of a new installation. It does not provide instructions for adding a distributed storage area to an existing installation.

Installing a remote site installs a remote Content Server and an Accelerated Content Services server and creates a local file store storage area for the site. The name of the local file store has the following format:

`fs_rcs_<server_config_name>`

where `<server_config_name>` is the name of the server configuration object for the site's remote Content Server. If the full name, including the `fs_rcs_` prefix is longer than 32 characters, the prefix is truncated to 32 or fewer characters.

The installation procedure also runs a script to install all the administration methods needed for that storage area.

To implement distributed storage:

1. Make sure that you have read and are complying with the guidelines in the previous section.
2. Decide how much disk space you need at each distributed site to store content files.
Read the instructions on estimating disk space requirements.
3. At the primary site (where the RDBMS is located), install Documentum CM Server and configure the repository.
4. Install the index agent and index server at the primary site.
5. Install Messaging Service on a separate host at the primary site if you are installing Branch Office Caching Services at the remote sites.

6. Start Documentum Administrator and connect to the repository as a user with superuser privileges.

7. Create the component storage area for the primary site.

Use Documentum Administrator to create the storage area.

To ensure backwards and future compatibility, do not use the default file store, `filestore_01`, as the distributed component at the primary site. Create a storage area to be the distributed component.

It is recommended that you place the content store on a different drive from the OpenText Documentum CM installation. This action allows you to separate OpenText Documentum CM programs from OpenText Documentum CM data.

8. Create the distributed storage area.

Use Documentum Administrator to create the distributed storage area.

Be sure to add the storage area you created in the previous step as a component of the distributed storage area. (Do not use `filestore_01` as the distributed component.)

If you intend to implement content sharing for any or all of the content files in the distributed storage area, do not check **Fetch Content Locally Only**.

If any of the following conditions exist, check **Fetch Content Locally Only**:

- If all the files are replicated among the distributed storage area components
- If the components are encrypted file stores
- If you use Surrogate Get

Checking **Fetch Content Locally Only** sets `only_fetch_close` to TRUE, which restricts the server's read access to its local component storage area. It is not able to fetch from the other component areas.

9. Set the default storage area for `SysObjects` to the distributed storage area.

Use type management facilities in Documentum Administrator to set the default storage area for `SysObjects`.

10. Move the objects currently in `filestore_01` to the distributed store.



Note: Objects in `filestore_01` are created by default during Documentum CM Server installation. These objects must be in the distributed store's component storage areas for OpenText Documentum CM desktop to function properly. However, you cannot make `filestore_01` a component of the distributed store to resolve this issue. Doing so does not reset the properties in the objects correctly. Additionally, it is recommended that you do not make `filestore_01` a component.

Use a DQL `UPDATE . . . OBJECTS` statement to move the objects:

```
UPDATE dm_sysobject (ALL) OBJECTS SET a_storage_type =
'<name_of_distributed_store>' WHERE a_storage_type = 'filestore_01'
```

`<name_of_distributed_store>` is the name of the distributed storage area you created.



Note: If the file store storage area has been in use already, it might have objects in it, such as installed lifecycles, that cannot be moved. The previous query fails in such circumstances. This situation can occur if you are adding a distributed storage area to an existing repository that has been in use for any time. To resolve this issue, either modify the query to exclude such objects or, in the case of lifecycles (`dm_policy` objects), make sure that they are removed first.

11. Set the timeout value for the server at the primary site to a minimum of 30 minutes.

The primary server's connection with a remote desktop client can time out while content is being transferred between the client and the remote Content Server. If a timeout occurs during a save operation on a new document, the primary server does not save the new object to the repository. To prevent this occurrence, set the `client_session_timeout` key in the [SERVER_STARTUP] section of the `server.ini` file to a minimum of 30 minutes. For more information about modifying the `server.ini` file, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

12. At each remote site, install a remote Content Server.



Note: All servers must be trusted servers or all servers must be non-trusted servers. You cannot mix trusted servers and non-trusted servers against one repository.

13. Log on to Documentum Administrator to perform the following manual steps:

- a. Update the distributed storage area to include the file store storage areas created at the remote sites.
- b. Update the **Far Stores** list in each site's server configuration object to include the component storage areas for the other sites.

For example, suppose there are three component sites, SiteX, SiteY, and SiteZ. If you are editing the server configuration object for siteX, its **Far Stores** list must include the component storage areas for SiteY and SiteZ.

- c. (Optional) Reset the proximity values for the remote Content Servers.

The installation process automatically sets a remote server's proximity to its local connection broker as 9001 and its proximity to the primary Documentum CM Server as 9010. You might want to reset the values to reflect your network's topography.

14. Make sure that servers at all sites can authenticate users and groups accessing distributed documents, using the same mechanism.

In a configuration using a distributed storage area, when a user connects to a repository, the client sends two connection requests: one to the data server and one to the remote Content Server. Each server must be able to authenticate the user's name and password using the same authentication mechanism.

15. If users at remote sites are accessing the repository using web-based client applications, perform the following additional steps for each remote site:
 - a. Define at least one network location for the Accelerated Content Services server at the site.
 - b. Add the network location to the Accelerated Content Services configuration object.

Add the location to the Accelerated Content Services configuration object for each Accelerated Content Services server that can service requests from that network location.
 - c. Define the network location's proximity to the Accelerated Content Services server or servers.
 - d. Define the Accelerated Content Services server's projection to a connection broker.
16. Make sure that all machines hosting a Documentum CM Server, remote Content Server, or Accelerated Content Services server are using UTC time and are synchronized.

When a server at a remote site (remote Content Server or Accelerated Content Services) connects to the primary site to fetch content using surrogate get, it uses a global login ticket configured to be valid for 5 minutes to connect. If the clocks in the host machines are not synchronized, the connection can fail if the host machine finds the login ticket invalid if the time difference is greater than 5 minutes.

3.11.2.3 The dm_rcs_setup.ebs script

The script `dm_rcs_setup.ebs` is executed automatically when you install a remote site. It creates the administration jobs necessary to manage and administer the remote site's file store storage area, such as Content Replication, `dmclean`, and `dmfilescan`. It also enables Surrogate get for the storage area at the remote site and creates a `sysadmin` log directory on the remote host.

You can run this script manually. (Rerunning the script in an installation in which it has already been executed is harmless.) The command line is:

```
dmbasic -f %DM_HOME%\install\admin\dm_rcs_setup.ebs -eRCSSetup -- <repository>
<server_config_name> <local_connection_broker_host> <local_connection_broker_port>
```

where:

- `<repository>` is the name of the repository.
- `<server_config_name>` is the name of the server configuration object for the Documentum CM Server at the remote site.
- `<local_connection_broker_host>` is the host name of the computer for the local connection broker.
- `<local_connection_broker_port>` is the port where the local connection broker listens.

3.11.3 Creating network locations

Use the information in this section to create network locations after you install the servers supporting a web-based, single-repository, distributed configuration.

Use Documentum Administrator to create network locations. For information about creating network locations, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.



Note: Do not create or modify a network location using DQL or the API. The Accelerated Content Services server does not recognize changes made through DQL or the API.

Network locations represent places or portions of a network's topography. Typically, each network location is defined as one or more IP addresses or range of addresses that are in that location. The addresses identified in network locations can overlap. That is, one or more network locations can include the same IP address or address range. This situation provides an opportunity for users to select their network location when they start sessions with web-based clients. The clients can be configured to present the users with a list of all network locations configured for their IP address.

However, defining an IP address or address range in a network location is optional. If a client application fails to find a network location that includes a user's IP address, the application assigns the user to a default location. The application can also present the user with a list of default locations and allows the user to select. If there are no defaults, the Accelerated Content Services at the primary site services content requests. A Boolean property in the objects that record network location definitions identify default network locations.

By default, each Accelerated Content Services server uses all of the network locations that are configured for the associated Documentum CM Server. For example, suppose a remote office in Florida has a Documentum CM Server and an Accelerated Content Services server. All IP addresses for machines in that office and the ones of the Florida-based telecommuters would be specified in one network location map object. That object is then identified in the Accelerated Content Services configuration object associated with the Accelerated Content Services server in the Florida office. An Accelerated Content Services (and Branch Office Caching Services) server can also serve multiple network locations. For example, perhaps a separate network location object is defined for each building in the Florida office and one for each telecommuter.

Network locations are recorded in the installation as `dm_network_location_map` objects that reside in the global registry. They must be stored in the global registry. The name of each location, stored in the `dm_network_location_map.netloc_ident` property, must be unique among the set of network locations in the global registry. If network locations are manually added to the Accelerated Content Services configuration object, they are identified in the `acs_network_locations` repeating property. Within a Branch Office Caching Services configuration object, they are identified in the `network_locations` property. If you want the locations to be

available to the Branch Office Caching Services server, add them to the Branch Office Caching Services configuration objects.

3.11.4 Adding network locations to an Accelerated Content Services or Branch Office Caching Services configuration object

To add a network location to an Accelerated Content Services or Branch Office Caching Services configuration object, perform the following actions:

- Identify the network location as a location serviced by the Accelerated Content Services or Branch Office Caching Services server represented by the configuration object.
- For Accelerated Content Services only, specify the network location's proximity to the server.

An Accelerated Content Services server can service multiple network locations, but it can be closer to some of those locations than it is to others. Set the network location's proximity value for each Accelerated Content Services server appropriately.

Use Documentum Administrator to update an Accelerated Content Services configuration or Branch Office Caching Services configuration object to add a network location and its proximity values.

3.11.5 Projecting an Accelerated Content Services server to connection brokers

Accelerated Content Services servers must project their presence to at least one connection broker. Documentum CM Server uses that information to determine which Accelerated Content Services servers are running. If the Accelerated Content Services configuration object represents a Branch Office Caching Services server, it is not necessary to set these properties.

The following table lists the properties used to project presence to a connection broker. These properties are repeating properties and the values in one index position across the properties represent the settings for the projection to one connection broker.

Table 3-9: Properties in Accelerated Content Services configuration objects related to connection broker projection

Property	Use
projection_enable	Determines whether the Accelerated Content Services server is projecting to the connection broker identified in the corresponding index position in projection_target. Set to T (TRUE) to project to the connection broker.  Note: If this property is set to true for a particular index position, then projection_netloc_enable at the same index position must be false.
projection_targets	List of the connection brokers to which the Accelerated Content Services server projects its presence and proximity. Set this property to the name of the connection broker.
projection_ports	List of the ports on which connection brokers are listening. The port specified at a particular index position corresponds to the connection broker identified at the corresponding index position in projection_targets.

3.11.6 Setting Accelerated Content Services proximity values for network locations

Proximity values for network locations represent the location's proximity to an Accelerated Content Services server. By default, the Accelerated Content Services server installed at the primary site has a built-in proximity of 9001. Therefore, by default, all network locations have a proximity of 9001 to the Accelerated Content Services server at the primary site. If a user at a network location cannot access other, closer Accelerated Content Services servers, the Accelerated Content Services server at the primary site is used.

Proximity values are not set in the network location object, but instead are set in the Accelerated Content Services configuration object. Alternatively, the proximity values defined in the server configuration object for the Documentum CM Server associated with the Accelerated Content Services server can be used.

Use Documentum Administrator to edit the Accelerated Content Services configuration or server configuration object. For more information, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

Table 3-10: Properties in Accelerated Content Services configuration and server configuration objects related to network proximity values

Property	Description
projection_netloc_enable (dm_server_config and dm_acs_config)	<p>Determines whether the system regards the Accelerated Content Services server as available for the network location identified in the corresponding index position in projection_netloc_ident.</p> <p>Set this property to T (TRUE) to make this Accelerated Content Services server accessible to users in the network location identified in the corresponding index position in projection_netloc_ident.</p> <p>If this property is F (FALSE), the system assumes that the Accelerated Content Services server is currently unable to handle requests originating in the corresponding network location.</p> <p> Note: If this property is true for a particular index position, then projection_enable at the same index position must be false.</p>
projection_netloc_ident (dm_acs_config) projection_netloc_id (dm_server_config)	<p>List of the network locations that the Accelerated Content Services server can service. Set this property to the name of the network location. This property is the name set in the netloc_ident property of the dm_network_location_map object.</p> <p> Note: Network locations specified in this property must also be listed in the dm_acs_config.acs_network_locations property.</p>

Property	Description
projection_proxval (dm_server_config and dm_acs_config)	<p>Proximity of the network location at the corresponding index position in the projection_netloc_ident property to the Accelerated Content Services server.</p> <p>In an Accelerated Content Services configuration object, set this property to a number from 1 through 8999. The lower the number, the closer to the Accelerated Content Services server the network location is presumed to be.</p> <p>In a server configuration object, the number might be over 9000. The system adjusts the value to use only the actual proximity, the value minus the 9000. (The number might also be being used to define a remote server's proximity.)</p>

3.11.7 Defining accessible storage areas for an Accelerated Content Services server

Which storage areas an Accelerated Content Services server can access is dependent on the value in its config_type property:

- If config_type is set to 1, the Accelerated Content Services server uses the storage areas listed in the server configuration object's far_store property as a list of unaccessible storage areas. That is, if a storage area is identified in the far_stores property, the Accelerated Content Services server cannot fetch content from that storage area. Instead, the Accelerated Content Services server can fetch content from any storage area visible to the server and not listed in that property.



Note: When the config_type is 1, the Accelerated Content Services Properties page in Documentum Administrator displays those storage areas listed in the Documentum CM Server's far_stores property. You cannot edit that field from the Accelerated Content Services Properties page. Edit it from the server configuration's Properties page. If the config_type is 1, **Use the projection targets, network locations, and near stores already configured for the associated server configuration object** is selected on the Accelerated Content Services Properties page.

- If the config_type property is set to 2, the Accelerated Content Services server uses the storage areas listed in the near_stores property of its Accelerated Content Services configuration object as its accessible storage areas. If the config_type is 2, **Manually enter projection targets, network locations, and near stores** is selected Accelerated Content Services Properties page.

The near_stores property is not set during the installation process.

By default, config_type is set to 1 when an Accelerated Content Services server is installed. To change the setting, and to set the near_stores property, use

Documentum Administrator to edit these values from the Accelerated Content Services server Properties page.

3.11.7.1 Accessing file stores in a distributed environment

Set the `is_public` parameter to `True` and mount the file system in a distributed environment to gain access to the file stores. Otherwise, an error message appears indicating that the current Documentum CM Server cannot access the storage area.

3.11.8 Modifying an `acs.properties` file

The `acs.properties` file contains configuration information for the Accelerated Content Services or Branch Office Caching Services server with which it is associated. The `repository.acsconfig` entry in the file identifies the server associated with the file. Use a text editor to modify the `acs.properties` file.

When the associated server is an Accelerated Content Services server, the location is as follows:

- Windows:

```
%DM_JMS_HOME%\webapps\ACS\WEB-INF\classes\config
```

- Linux:

```
$DM_JMS_HOME/webapps/ACS/WEB-INF/classes/config
```

When the associated server is a Branch Office Caching Services server, the location is as follows:

- Windows:

```
%DOCUMENTUM%\tomcat\webapps\bocs\WEB-INF\classes\config\acs.properties
```

- Linux:

```
$DOCUMENTUM/tomcat/webapps/bocs/WEB-INF/classes/config/acs.properties
```

3.11.8.1 The `mode.cachestoreonly` entry

This key indicates whether the `acs.properties` file is associated with an Accelerated Content Services or Branch Office Caching Services server. The value in this key must match the value specified in the `is_cache_acs` property of the Accelerated Content Services configuration object for the server. The values of both these items are set when the Accelerated Content Services or Branch Office Caching Services server is installed.

Do not change these values. If the values of the key and the Accelerated Content Services configuration property do not match, the server does not work. If the values are incorrect for the server, the server does not behave appropriately.

3.11.8.2 Adding entries for additional servers

If you are updating the file to allow an Accelerated Content Services server to communicate with an additional server for a repository in an installation, set the following configuration parameters for the server:

```
repository.name
# name of the repository
repository.login
# login name to be used to connect the server
repository.acsconfig
# name of the acs config object created for the server
```

Each parameter entry of a particular type is numbered after the first. For example, suppose the file currently has entries for only one server, the server for the Engineering repository. You want to add entries for an additional server for that repository. The additional entries would be:

```
repository.name.1=engr_1 repository.login.1=engr_1_login_user
repository.acsconfig.1=<acs config object name>
```

Each additional entry for each parameter is designated with a 1. If you added a third server, the number increments by one. The entries for the third server would end in 2 (The first entry for each parameter has no number.) You can add up to 99 entries for these parameters.

You must specify the password in `repository.password.#` that corresponds to the user in `repository.login.#`. The password must be encrypted. Use the Foundation Java API password encryption utility to encrypt the password; for example:

```
java -cp dfc.jar com.documentum.fc.tools.RegistryPasswordUtils
<password_to_encrypt>
```

3.11.9 Disabling access to an Accelerated Content Services server

You can disable access to an Accelerated Content Services server by particular network locations or by all network locations using that server. Use Documentum Administrator to disable access to an Accelerated Content Services server by network location.

To disable a particular network location's access to a particular Accelerated Content Services server:

- If the `config_type` property in the Accelerated Content Services configuration object is set to 2, set the `projection_netloc_enable` property in the Accelerated Content Services configuration object to F at the index position associated with that network location. For example, if you set `projection_netloc_enable[2]=F`, then access from network location identified in `projection_netloc_ident[2]` is disabled.
- If the `config_type` property in the Accelerated Content Services configuration object is set to 1, set the `projection_netloc_enable` property in the associated server configuration object to F at the index position associated with that network location. For example, if you set `projection_netloc_enable[2]=F`, then access from network location identified in `projection_netloc_id[2]` is disabled.

To disable all access to the server, from any network location, set the Accelerated Content Services configuration object's config_type property to 0.

3.11.10 Configuring shared content files

To enable content file sharing among a distributed storage area's components, you must:

- Configure each component storage area directory as a shared directory before you add it to the distributed storage area.
- Set the far_stores property for the server configuration object at each site.
- Set the only_fetch_close property of the distributed storage area to FALSE.

The far_stores property in a server configuration object defines the distributed storage area components that are not local for that server. For example, suppose a distributed site has three locations: London, New York, and Paris. At the New York site, the far_stores property for the server is set to London and Paris. In London, the property is set to New York and Paris. And finally, in Paris it is set to New York and London.



Note: If you followed the installation procedure for distributed storage areas, the far_stores property for the server at each of your distributed sites contains the names of all the other sites.

A server cannot save to a storage area defined as far. The only_fetch_close property controls whether the server can fetch from a far storage area.

If the storage areas are shared directories and the property is FALSE (the default), the server can fetch files from far storage areas directly.

When only_fetch_close is TRUE, servers cannot fetch content files from component storage areas named in the far_stores property in their server configuration objects. If a user requests a document that is only in a far storage area, the server satisfies the request using surrogate get.

3.11.11 Creating pre-caching content jobs

A pre-caching content job generates a pre-caching request that is sent to the Messaging Service server. The arguments in the job specify a query to identify the content you want to pre-cache and the network locations for which you want to cache the content. When the job executes, it executes the query and post a pre-caching request to the Messaging Service. The job does not transfer any content for caching itself. It only posts the request for pre-caching to the Messaging Service server. The Messaging Service server passes along the request to the appropriate Branch Office Caching Services servers, which in turn, initiates the pre-caching operation.

Use Documentum Administrator to create a pre-caching job. A pre-caching job executes the dm_PreCacheContent method. The following table lists the arguments supported by the method:

Table 3-11: dm_PreCacheContent method arguments

Argument	Description
docbase_name	Name of the repository that contains the content you wish to pre-cache
user_name	Installation owner user name
query_type	Name of the object type that is the target of the DQL query
query_predicate	DQL WHERE clause qualification that identifies the content to be pre-cached
network_locations	Comma-separated list of network location object IDs or the keyword dm_all_network_locations. When the keyword is specified, the method retrieves all available network location object IDs.
job_id	Object ID of the job invoking the method
cutoff_time	(Optional) Defines a cutoff time for content. Only content that satisfies the predicate and created after this time is cached. The value specified in cutoff_time is compared with the value in the set_time property of the dmr_content object. The format of the value in this argument is: mm/dd/yyyy HH:MM:SS
expiration	(Optional) Length of time, in seconds, for which the messages generated by this job are valid. The default value is 3600 seconds (1 hour).
batch_size	(Optional) Number of objects processed in each request. For example, if the predicate returns 150 objects, and the batch size is 50, the method processes the results in three batches, 50 objects in each batch. The default value is 50.

When the method executes, it finds all content files that satisfy the specified predicate and have a set_time value that is equal to or greater than the cutoff_time argument. The underlying query orders the results in ascending order, based on the set_time value. For the resulting content files, it sends messages to Messaging Service requesting pre-caching for the content.

Each time the job completes, it resets the cutoff_time argument to the set_time value of the last content object processed by the method. In this way, the job does not send multiple pre-caching requests for the same content.

If the job stops before processing all returned content files, it resets the cutoff_time argument to the set_time value of the last processed content file. When the job is restarted, it resumes with the next content file.

If a content fails to pre-cache after the Messaging Service delivers the pre-caching request to a Branch Office Caching Services server, then you can run the method manually with a custom predicate to regenerate the request for that content. Use Documentum Administrator to run the method.

3.11.12 Setting up content replication

Replicating content files among distributed storage area components ensures that users at each site have local copies of the files to access. Documentum CM Server includes several tools for content replication. You can set up replication to run automatically or you can perform the operation manually.

3.11.12.1 Deciding which tool to use

Deciding which tool to use based on how often replicated content changes and how important it is for users at any site to have access to current versions of the files.

3.11.12.1.1 Automatic Replication

There are two ways to replicate content automatically. You can use either or both in a single-repository distributed configuration. The tools are:

- ContentReplication tool
- Surrogate get feature

The ContentReplication tool greatly simplifies administration of content replication. The tool requires only enough temporary disk space to transfer the largest individual content file to be replicated. In contrast, using dump and load requires enough temporary space to hold a dump file containing all content files to be replicated. The ContentReplication tool is recommended for most situations. For information about using the ContentReplication tool, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

The surrogate get feature replicates content on demand to the user's local storage area. When a user attempts to access the content, the server recognizes that the content is not available locally and calls the dm_SurrogateGet method. The method obtains the content from another server and executes an IMPORT_REPLICA call on the local server to make the content available to the user. The dm_SurrogateGet method is a system-defined method installed with Documentum CM Server.

3.11.12.1.2 Manual replication

To replicate content files manually, you can use:

- The REPLICATE administration method
- The IMPORT_REPLICA administration method

The REPLICATE administration method copies a file from one storage area to another. The disks on which both component storage areas reside must be accessible to the server.

The IMPORT_REPLICA administration method imports a file from an external file system into a storage area. You can use IMPORT_REPLICA with or without NFS access.

You can execute either REPLICATE or IMPORT_REPLICA from Documentum Administrator or using the EXECUTE statement or the Apply method. On Linux platforms, both source and target storage areas must be available to the server through NFS.

3.11.12.2 Using the surrogate get feature

When a user tries to access a content file that is stored in a remote component of a distributed storage area, the server must either fetch the file from the remote area directly or invoke the dm_SurrogateGet method to fetch the file and replicate it to the local storage area.

Use surrogate get if your site is not using shared directories. Read the instructions if you are using shared directories.

To use surrogate get, you must:

- Set the far_stores property for the server configuration object at each site.
- Set the only_fetch_close property of the distributed storage area to TRUE.
- Make sure that the get_method properties in the storage areas are set to the name of the surrogate get method.
- Set the clocks on participating host machines to UTC time and synchronize the clocks.

Surrogate get uses a login ticket to connect to servers to obtain content files. The login tickets are valid for 5 minutes. If the machine on which the dm_SurrogateGet method is running and the host machine of the server to which it is connecting are not synchronized in time, the ticket can be invalid on the target. The ticket becomes invalid because of perceived differences in time. In such cases, the job fails.

The far_stores property in a server configuration object defines the distributed storage area components that are not local for that server. For example, suppose a distributed site has three locations: London, New York, and Paris. At the New York site, the far_stores property for the server is set to London and Paris. In London, the

property is set to New York and Paris. And finally, in Paris it is set to New York and London.



Note: If you followed the installation procedure for distributed storage areas, the far_stores property for the server at each of your distributed sites contains the names of all the other sites.

The only_fetch_close property controls whether a server can fetch content from a component storage area named in the far_stores property. When only_fetch_close is TRUE (the default), servers cannot fetch content files from component storage areas named in the far_stores property in their server configuration objects. If a user opens a document whose content file is only in a far storage area, the server invokes the dm_SurrogateGet method to retrieve the file. This method is identified in the storage area's get_method property.

3.11.12.2.1 The dm_SurrogateGet method

The dm_SurrogateGet method implements the surrogate get feature. The method is installed automatically along with the system administration tools. When you create a storage area, the get_method property of the storage area object is set to dm_SurrogateGet.

By default the dm_SurrogateGet method requires the following arguments to control whether to perform a content check:

- do_content_check, which verifies the hash values of the content files and performs a content check.
- no_content_check, which does not perform a content check.

The method uses a global login ticket to connect to remote servers. The ticket has a validity period of 5 minutes. Host machines must be set to UTC time and synchronized. If that is not done, the host machine of the process receiving the login ticket can consider the ticket invalid if the time difference is greater than 5 minutes.

All surrogate get operations can be traced. When tracing is turned on, the trace file is called sugg.trc and is found in the %DOCUMENTUM%\bin (Windows) or \$DOCUMENTUM/bin (Linux) directory.

You can also turn on tracing for the method invocation. Tracing the method invocation logs the method command line in the server log whenever the method is invoked.

3.11.12.3 Using REPLICATE

The REPLICATE administration method copies a file from one component storage area to another. The arguments determine which file or files are copied and where the copies are put. For information about the syntax, see *OpenText Documentum Content Management - Server DQL Reference Guide (EDCCS250400-DRD)*.

You cannot select a component storage area from which to copy the file or files. Select only the target area. The server searches the remaining component areas in the distributed storage area and replicates the first copy of the file it finds.

Use the EXECUTE DQL command to execute this administration method as follows

```
EXECUTE replicate WITH query = '<dql_predicate>', store = '<target_storage_area_name>'
```

3.11.12.4 Using IMPORT_REPLICA

The IMPORT_REPLICA administration method copies a file into a distributed storage component storage area. The file can be located in another component of the distributed storage area or in an external file system. If the source file is in another component of the distributed storage area, the server must have access to the disk on which the area resides. On Linux, if the disk is on another machine, the server must have access to NFS. If the file is in a directory on the same machine as the server or on a tape, diskette, or floppy, NFS is not required.

Use the EXECUTE DQL command to execute IMPORT_REPLICA as follows:

```
EXECUTE import_replica FOR <content_object_id>WITH store =
'<name_of_target_storage_area>', file = '<file_path_of_desired_file>'
```

3.11.13 Setting proximity values for Documentum CM Server projection to a connection broker

Use the information in this section to help you set appropriate proximity values for Documentum CM Servers.

3.11.13.1 Guidelines

Use the following guidelines to define proximity values:

- Define proximity values for data servers in the range of 0 to 999. It is not necessary to include leading zeros.
- Define proximity values for remote Content Servers in the range of 9000 to 9999.
- Define proximity values for the primary server (the Documentum CM Server at the primary site) so that:
 - The server always projects the lowest proximity value among all servers projecting to any connection broker.

- The server projects a content proximity value to the primary site connection broker that is lower than those values projected to the site by the remote servers.
- The server projects content proximity values to remote connection brokers that are higher than those values projected from the remote sites.
- Define proximity values for a remote Content Server so that:
 - The proximity value it projects to a connection broker is higher than the proximity valued projected by the primary server.
 - The proximity value it projects to its local connection broker is lower than any projected to that connection broker from other sites.
- If two Documentum CM Servers have the same content proximity value and a client has established a data connection to one of them, the client uses the established connection for content requests also.
- A proximity value of -1 represents a server that is not available for any type of connection.

3.11.13.2 Example of selecting proximity values

The following table illustrates the proximity value guidelines. It shows projected proximity values for the servers in a repository called Demo. The repository has three sites: A, B, and C, with corresponding servers named DemoA, DemoB, and DemoC, and connection brokers named Connection BrokerA, Connection BrokerB, and Connection BrokerC.

Table 3-12: Example proximity values for a three-site configuration

Server	Projected Proximity Values		
	To Connection BrokerA	To Connection BrokerB	To Connection BrokerC
DemoA	0001	0002	0003
DemoB	9002	9001	9003
DemoC	9003	9002	9001

3.11.13.2.1 At site A

When a user requests a connection, Connection BrokerA returns the following server and proximity value pairs:

Server	Proximity Value
DemoA	0001
DemoB	9002
DemoC	9003

For data requests, the client uses DemoA because it is the only server with a proximity value in the 0 - 999 range. (If there were more than one server with a proximity value in that range, the client selects the server with the lowest value.)

For content requests, the client also selects DemoA also because it has the lowest projected content proximity (001). Using the same server for data and content at the primary site is fine. DemoA is the closest server for users at the primary site.



Note: You can set up both a data server and a remote Content Server at the primary site. The remote Content Server at the primary site must project a proximity value to the local connection broker that is higher than the data server's proximity value and a content proximity value that is lower than any value projected by the servers at the remote sites.

3.11.13.2.2 At site B

When a user requests a connection, Connection BrokerB returns the following server and proximity value pairs:

Server	Proximity Value
DemoA	0002
DemoB	9001
DemoC	9002

For data requests, the client uses DemoA, as it is the only server with a proximity value in the 0 - 999 range. For content requests, the client selects DemoB because its content proximity value (001) is lower than the content proximity value (002) projected by DemoA or DemoC.

3.11.13.2.3 At site C

When a user requests a connection, Connection BrokerC returns the following server and proximity value pairs:

Server	Proximity Value
DemoA	0003
DemoB	9003
DemoC	9001

For data requests, the client uses DemoA, the only server with a proximity value from 0 through 999. For content requests, the client selects DemoC because its content proximity value (001) is lower than the content proximity value projected by either DemoA or DemoB (003).

3.11.13.3 First-time use

The first time a remote Content Server issues a Getfile request to retrieve a document, the response time can be slow. The server constructs some internal structures to handle Getfile requests. Subsequent calls use the structures built with the first call and are faster.

3.12 Implementing multi-repository models

This section describes how to set up the most commonly used building blocks for multi-repository distributed environments.

3.12.1 Repository configuration for distributed environment

Use the information in this section to complete the configuration of your repositories if your repositories are participating in:

- A repository federation
- Object replication
- Distributed workflow

3.12.1.1 Connection broker setup

Every server in the distributed environment must be able to find the other servers through its connection broker.

Like an end user, a Documentum CM Server also uses a connection broker to find outside repositories. The connection brokers a server uses are identified in the `dfc.properties` file on the server host machine. If the participating servers all project to the same connection broker and send to that connection broker for connection information, then no special setup is required. However, if the participating servers use different connection brokers for connection information, then additional configuration is required. In such cases, make sure that each participating server projects its connection information to the connection brokers used by all other participating servers.

Target connection brokers for server projection are defined in the server configuration object. The target information is stored in five repeating properties:

- `projection_targets`

The `projection_targets` property contains the name of the host machine on which the connection broker resides.

- `projection_port`

The `projection_ports` property contains the port number on which the connection broker is listening.

- `projection_proxval`

The `projection_proxval` property contains the proximity value that the server projects to the connection broker.

- `projection_enable`

The `projection_enable` property determines whether the server projects to the connection broker. If it is set to TRUE, the server projects to the connection broker. If it is set to FALSE, the server does not project the connection broker.

- `projection_notes`

The `projection_notes` property is a place for you to record short notes about the target. The property is 80 characters long.

Use the repository management facilities in Documentum Administrator to modify the server configuration object to set the projection targets for a server. For information, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

3.12.1.2 User setup

In the current implementation, the participating repositories must all use the same user name when establishing a connection for the internal jobs that manage distributed operations.

To configure the repository user account to use for distributed operations:

1. Create a user account that has Superuser privileges in each participating repository.

The simplest arrangement is to use the same user for all repositories, as this arrangement requires defining only a single global superuser account.



Note: If you are creating a global user through a federation's governing repository, the governing repository propagates the user to the other repositories. However, the user is created with no special privileges in the member repositories. Then connect to each member repository and set the user's privileges to Superuser.

2. In each participating repository, set the operator_name property of the server's server configuration object to the user's login name.
3. Set up the password for the user in each repository.

3.12.1.3 Password setup

The server for each repository must have a password for the user you defined. To define the password, set up a dm_operator.cnt for each repository and the appropriate number of <repository>.cnt files for each repository. The following procedures describe how to set up these files:

To set up the dm_operator.cnt file:

1. Create a file named dm_operator.cnt that contains the password for the user identified in the operator name.

The password must be the same in all the dm_operator.cnt files.

2. Place a copy of the file in each repository in one of the following directories:
 - Windows: %DOCUMENTUM%\dba\config\<repository_name>
 - Linux: \$DOCUMENTUM/dba/config/<repository_name>

To set up the repository.cnt file:

1. For each repository:
 - a. Create a file named <repository_name>.cnt, where <repository_name> is the name of the repository.
 - b. Put the user's password in that file.

The password can be the same or different for each repository.

2. In each repository, place a copy of the `<repository_name>.cnt` file for each of the other participating repositories in the following location:
 - Windows: `%DOCUMENTUM%\dba\config\<repository_name>`
 - Linux: `$DOCUMENTUM/dba/config/<repository_name>`

For example, suppose there are three participating repositories: RepositoryA, RepositoryB, and RepositoryC. You create three files:

- `repositoryA.cnt`, which contains the user's password for Repository A
- `repositoryB.cnt`, which contains the user's password for Repository B
- `repositoryC.cnt`, which contains the user's password for Repository C

Then, you place:

- Copies of `repositoryB.cnt` and `repositoryC.cnt` in Repository A
- Copies of `repositoryA.cnt` and `repositoryC.cnt` in Repository B
- Copies of `repositoryA.cnt` and `repositoryB.cnt` in Repository C

3.12.1.3.1 Object replication jobs

If you have object replication jobs that have defined a user other than the user identified in the user setup process, errors can occur when you execute the jobs. The errors can occur because object replication jobs now use the `dm_operator.cnt` or `<repository_name>.cnt` password files to retrieve the password for the remote user.

For example, if you are running a replication job from target repositoryB that connects to repositoryA as a pull replication, the repositoryB server reads the local `dm_operator.cnt` or `repositoryA.cnt` file (depending on which is present) to obtain a password to use to connect to repositoryA.

To avoid errors, either give the remote superuser defined in your jobs the same password as the user identified in the user setup process or change the remote user in the job to the user identified in user setup process.

3.12.1.4 Distributed operations job activation

To complete the configuration for distributed operations, the `dm_DistOperations` must be activated in each participating repository. This job is installed in the inactive state.

Use the Jobs management facilities in Documentum Administrator to activate the job in the participating repositories. The `dm_DistOperations` job is categorized as a replication job.

3.12.2 Setting up a federation

A federation is two or more repositories that are bound together to facilitate management of global users, groups, and ACLs in a multi-repository distributed configuration. One repository in the federation is defined as the governing repository. All changes to global users, groups, and external ACLs must be made through the governing repository.

If an enterprise includes multiple, mutually exclusive groups that do not share documents, you can set up multiple federations. However, a repository can belong to only one federation.

A federation can include repositories with trusted servers and repositories with non-trusted servers.

It is not recommended to mix production, test, and development repositories in one federation.

3.12.2.1 Choosing the governing repository

Consider creating a repository to be the governing repository. Such a repository has the following advantages:

- Small size
- Easier backups
- Supporting jobs are run in a small repository

If you are creating a federation from a group of existing repositories, select the dominant repository as the governing repository. The dominant repository is the repository in which most users are already defined as repository users.

3.12.2.2 Identifying user subtypes for propagation

By default, the federation jobs propagate all global users defined as dm_user objects in the governing repository. If you have created subtypes of the dm_user object type, global users defined with those subtypes are not propagated automatically.

To propagate users defined by dm_user subtypes, the subtypes must be present in all member repositories. The subtypes also must identify the user subtypes to propagate when you create the federation. You can add or delete from the subtypes after the federation is created also.

3.12.2.3 Creating a federation

Federations are created and managed through Documentum Administrator. For more information, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

To create a federation:

1. Decide which repositories participate in the federation and which repository is the governing repository.
2. Make sure that the governing repository is projecting to the connection brokers for all member repositories.
3. Make sure that the member repositories are projecting to the governing repository's connection broker.

Only repositories that project to the governing repository's connection broker appear in the list of possible member repositories.

Projection targets are defined in the server configuration object, which can be modified using Documentum Administrator. For information about adding connection broker projection targets to servers, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

4. Create the federation using Documentum Administrator. For information, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

Guidelines:

- Selecting the check box to make all current users and groups in the repository global activates the dm_FederationUpdate job. This job is responsible for starting the methods that keep the global objects synchronized across the federation. The job is inactive when installed.
If you do not select the check box to set all users and groups to global, manually set the dm_FederationUpdate job to active. Use Documentum Administrator to do that.
- The federation name must consist of ASCII characters and cannot contain any single quotes ('). You cannot change this name after the federation is created.
- The server uses the superuser name and password you provide for each member to connect to the member repository when running federation jobs and methods that update the global objects.
- If the member repository is running on a domain-required Windows server or if the superuser is in a domain that is different from the default, enter a domain for the superuser.

3.12.3 Implementing object replication

The following procedure lists the basic steps for planning and implementing object replication:

To implement object replication:

1. Define the business requirements for replication.
2. Determine whether your current infrastructure meets the needs of the business requirements.
3. Determine the needed computing resources.
4. Set up each site.
5. Define the replication jobs.

3.12.3.1 Defining business requirements

Begin planning object replication implementation by determining your business requirements for replication. Two considerations can affect your business requirements:

- A replication job can only replicate documents to one target repository from one source repository.
- A replication job does not replicate users, groups, or object types.

The first consideration means that more than one replication job can be required to satisfy a business requirement. For instance, to distribute documents to three geographically dispersed locations, you need three replication jobs.

The second consideration means that you must coordinate multi-site, multi-repository users, groups, and access permissions as a business function between repositories. Object types, users, and groups, are not replicated as part of the replication job. ACLs (dm_acl objects) can be replicated, depending on how you configure the job.

Setting up a repository federation that includes all repositories participating in object replication enables you to coordinate users, groups, and security access across repositories. In a federation, users, groups, and external ACLs are global objects. You can change them at the governing site and propagate the changes automatically to other members of the federation. Manually manage object types and formats regardless of whether the participating repositories are in the same federation.

If you select not to use a federation, manage users, groups, and security manually.

For example, suppose a planning group has users in two different repositories and the repositories are not members of the same federation. Tom, Dick, and Harry are users in repository 1, while Jane and Mary are users in repository 2. To allow these users to share documents of the object type planning_doc, the administrators for the two repositories must, at a minimum:

- Make sure that the planning_doc object type is defined identically in both repositories



Note: If you replicate an object into a repository in which the object's type does not exist, the operation creates the type in the target repository. However, if the type exists in both source and target repositories, define it identically in each.

- Create a planning group within each repository
- Add the users to both planning groups
- Create an ACL with an entry for the planning group that gives the group sufficient access to view replicated documents and their annotations
- Create a folder appropriate for the planning group's document sharing

The following sections describe the typical business requirements for users, groups, object types, security, workflow, and administration needs. These sections also discuss how these requirements affect your replication implementation.

3.12.3.1.1 Functional divisions and groups

If your company has clearly defined functional divisions, translate these divisions into appropriate OpenText Documentum CM group objects, to be defined in all repositories involved in replication. If you are developing an enterprise-wide replication plan before actually creating any repositories, create a standard script that defines these groups.

If you are not placing the repositories in a federation, you can run the standard script in each repository. This practice ensures that the group definitions meet the business requirement and are standard across all repositories. Edit the script for each repository so that the group creation statements are specific to the users in that repository.

If the participating repositories belong to a federation, it is only necessary to run the script at the governing repository site. The groups are automatically propagated to the other sites when the federation is created and the management jobs are active.

If you are converting existing repositories to accommodate a new or modified business requirement for replication, you might find that you must create new groups or modify existing groups. It still could be possible to utilize a standard script for this purpose, but some repository-by-repository modifications could also be necessary.

3.12.3.1.2 Document types

Document types usually evolve out of a combination of enterprise-wide and functional business requirements. The document types must have properties that capture all of the information necessary for users to access and utilize the document in all business contexts. To preserve this information in replicated documents, it is important to define and maintain enterprise-wide document type definitions.

Document types can be defined using a standard script, which is recommended if your repositories are not yet created. If you are converting existing repositories to meet replication business requirements, you might have to create new definitions or modify existing definitions.

Maintaining enterprise-wide document type definitions is desirable. It preserves all property and content information when a document is replicated. However, it is not mandatory for OpenText Documentum CM replication. If the definitions are not identical, the system copies all information possible and ignore any information that cannot be replicated.

For example, suppose the user-defined document type planning_doc has 15 user-defined properties including the property project_leader in repository 1. But it lacks that property in repository 2. Replication from repository 1 to repository 2 would result in a replica with 14 of the user-defined properties, but not project_leader. A replication from repository 2 to repository 1 would result in a replica with all 15 properties present, but with no information in the project_leader property.

If you replicate an object whose type is not defined in the target repository, the operation creates the type in the target repository as part of the replication process.

3.12.3.1.3 User distribution and geography

OpenText Documentum CM defines users at the repository level. If a user is defined in more than one repository, that user has a unique OpenText Documentum CM user ID in each repository. In a default installation, the server considers each a different user, even if the user_os_name and user_name properties are identical in both repositories. In a federation, users are global objects managed by the governing repository. The server considers users having the same user_os_name and user_name properties in different repositories to be identical.

There are no user management concerns if you are replicating between repositories in the same federation. The federation's management jobs ensure that the user definitions are the same in each member repository.

Replication between repositories that are not in a federation or not in the same federation does not require user definitions to be the same across the repositories. In such cases, the replication job maps the ownership of the replicated objects to users in the target repository.

If your company has no policy for uniquely identifying users across sites, you can define users repository by repository. If there is a policy for uniquely identifying

users across all sites, you can use this identification scheme in OpenText Documentum CM without modification. The product works in either situation.

When a new user joins a project or the company, it is up to the administrator to add the user to the appropriate groups. These groups include any groups participating in replication processes if necessary. Generally, cross-repository coordination is not required. However, you might have set up some customizations that require coordination. For example, if you create registered tables of remote repository users and user_names in each repository (to support cross-repository event notification, for example), adding a new user to a repository requires coordination. In such cases, the new user must be added to the registered tables in the remote repositories also.

3.12.3.1.4 Security

The choices you make for security depend on the replication mode you are using. There are two replication modes: nonfederated and federated. Each provides different security options. describes the security options available for each mode.

If you are using a nonfederated mode and select to assign the same ACL to all replicas, security requirements can dictate which documents are included in a replication job. You could implement a replication business requirement with complex security by creating an ACL containing grants to many groups within the target repository, each with different access rights. Alternatively, the replication job could be divided into a group of replication jobs, each with its own simple but unique ACL.

3.12.3.2 Infrastructure

After you define your business requirements for replication, determine whether your infrastructure meets the needs of the requirements. Infrastructure is defined as the hardware, networking software, and people that support OpenText Documentum CM replication. Determine the following:

- Whether you have adequate hardware resources
- Whether you want to perform replication online or offline
- How you want to assign the duties associated with managing a replication site.

The following sections address these issues.

3.12.3.2.1 Reference metrics

Reference metrics provide a baseline for capacity planning, identifying potential infrastructure weaknesses, and determining what performance can be expected. To help determine whether the hardware and software infrastructure at your site is adequate to support your replication needs, compute reference metrics on each server that participates in your replication configuration. Compute the metrics during both off-peak and peak times because replication jobs can run at both times.

Obtain the following baseline metrics:

- Server CPU capability

Record the elapsed time for a local OpenText Documentum CM client to create and save 1,000 objects that have no content. This operation is an approximating metric for gauging CPU capability. Comparing CPU capability across environments is difficult.

- Server disk capacity

Report `df` command information for the Linux servers.

- Server disk speed

Perform a local OpenText Documentum CM client Setfile/Getfile and record the elapsed time for each.

- Network speed

Perform a Setfile/Getfile between each server pair participating in replication and record the elapsed time for each.



Note: OpenText Documentum CM provides a Docbasic script that you can use as a base for writing a script to obtain all of the reference metrics except disk capacity. You can find the script in %DM_HOME%\unsupported\replicate\metric.ebs (Windows) or \$DM_HOME/unsupported/replicate/metric.ebs (Linux).

For example, the following table shows the baseline metrics obtained for the two servers participating in replication testing. The metrics are expressed as minutes.

Metric	Fox, off-peak	Fox, peak	Bison, off-peak	Bison, peak
Server CPU	3:40	8:25	3:50	7:14
Local Setfile	0:30	1:30	:17	1:25
Local Getfile	0:17	1:57	0:14	1:30
Remote Setfile	31:02	37:29	30:56	36:17
Remote Getfile	29:57	35:56	29:30	36:49

If you obtain inadequate metrics in any of these areas, factor that into your infrastructure planning:

- Alleviate CPU shortfalls by adding processors to your server or by putting RDBMS processing on a different server than Documentum CM Server.
- Alleviate disk shortfalls by procuring additional disk devices and rearranging the map of logical devices to controllers and physical disks.
- Alleviate network shortfalls by examining network router losses, procuring additional network bandwidth, or both.

If you cannot resolve network bandwidth shortfalls, consider using the offline replication option.

3.12.3.2.2 Network replication options

There are two basic options for object replication: online and offline.

Online replication is the default. In online replication, the replication job originates at the target site and performs the following tasks:

- Synchronously requests source-site processing
- Synchronously transfers the resulting dump file
- Synchronously performs the target-site processing to complete the replication

In offline replication, the replication job originates at the target site and requests the source-site processing. Asynchronously, the source site places the dump file in a requested location. A system administrator then uses ftp or tape to move the dump file from its source location to the target location. After the dump file is placed in its target location, the replication job picks up where it left off, performing the remaining target-site processing.

3.12.3.2.3 Replication system administration

In addition to the initial planning and coordination between sites, enterprise-wide replication has ongoing administrative tasks. For example, the duties associated with replication can include:

- Adding groups and users to the appropriate repositories
- Resolving problems when networks or repositories are not available
- Monitoring the progress of replication jobs and resolving failures
- Coordinating object type definitions across repositories

An enterprise with a number of large repositories at relatively autonomous sites can require one additional full-time individual per site to administer replication. This individual requires conventional OpenText Documentum CM skills and business knowledge to translate business replication requests into appropriate replication jobs.



Note: Auditing is not supported when performing refresh operations on a replica. Performing an audit each time a refresh operation is performed on a replica would affect performance and use up disk space.

3.12.3.3 Determining computing resources

Determining the computing resources required for replication is primarily a matter of:

- Listing business requirements
- Translating those requirements into specific replication jobs
- Extrapolating the required machine resources based on the parameters of each job

Disk space requirements and job scheduling at each site require examination. This section demonstrates this process by example.

3.12.3.3.1 Determining needed jobs

XYZ Enterprises has three geographically dispersed repositories: X, Y, and Z. They have two products on the market that were developed at their original site, X, which continues to control everything related to these products. However, Site Y needs rapid access to the documentation for products 1 and 2, and site Z needs rapid access to product 2 documentation.

Additionally, XYZ Enterprises is developing a new product that requires collaboration among all three sites. Every document produced with the new product is replicated from its originating site to the other two. All three sites are expected to generate large amounts of review annotations, which must also be replicated to all sites.

The company has defined the following six replication jobs to achieve these business objectives:

 **Example 3-1: Product 1 Replication Jobs:**

- Job 1: From X to Y once a week



 **Example 3-2: Product 2 Replication Jobs:**

- Job 2: From X to Y once a week
- Job 3: From Y to Z (X documents received indirectly from Y) once a week



 **Example 3-3: New Product Replication Jobs:**

- Job 4: From X to Y every 2 hours
- Job 5: From Y to Z every 2 hours
- Job 6: From Z to X every 2 hours



Jobs 1 and 2 represent the standard distribution of X documents to Site Y. Because it is possible to replicate replica documents, Job 3 completes the distribution by replicating X documents to Z through Y.

For Jobs 4, 5, and 6, each site's target and source folder for the new product documentation is called /NewProd. This configuration and the circular nature of Jobs 4-6 mean that each site has its own documents and annotations; in addition, all documents and annotations from the /NewProd folders of the other sites in its own / NewProd folder.

The following illustration depicts these jobs:

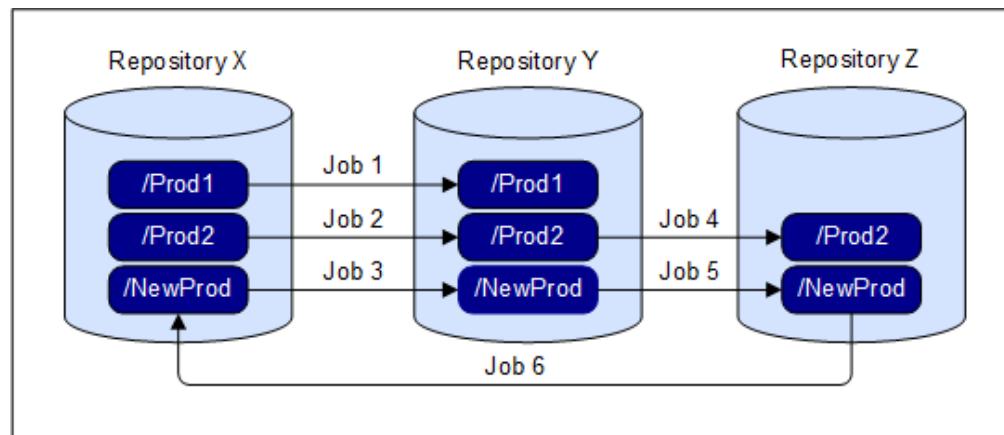


Figure 3-8: XYZ jobs

3.12.3.3.2 Disk space requirements

This section illustrates how to estimate the disk space required for replication as accurately as possible. The values used in this section for each required estimate are only examples. For your calculations, determine the correct figures for the required estimates.

3.12.3.3.2.1 For replicated documents

Use a table to calculate disk space requirements. First, compute the requirements for the source repository documents and metadata. This operation is a function of the:

- Number of source documents and virtual descendant (related) documents
- Estimated average document size
- Estimated rendition size
- Estimated annotation size

Most documents have multiple versions, so be sure to take versions into account also.

This table shows these figures for the XYZ Enterprises example.

	Product 1, X	Product 2, X	New Product, X	New Product, Y	New Product, Z
Number of documents	1,000	2,000	100	50	75
Number of versions	6	5	4	3	2
Content (KB)	10	20	5	15	18
Renditions (KB)	20	40	5	15	18
Annotations	0	0	33	5	5
Total content (KB)	30	60	43	35	41
Estimated Total (MB)	200	650	18	4.35	11.41

Total content is the sum of content, renditions, and annotations for each document.

The total represents the total size of the repository document and metadata content. It is the product of the total number of documents (number of documents times the number of versions per document) and the total number of bytes per document (total content + 2,500 bytes per document for metadata overhead), or

$$\text{Total} = (\text{no. of documents} \times \text{no. of versions per document}) \times (\text{total content} + 2,500 \text{ bytes})$$

The metadata overhead varies, depending on the complexity of the document types.

After you calculate source disk space, you can project the total requirement to each of the sites. The total requirement is the sum of the source and replicated documents at each site plus required temporary space for dump files.

3.12.3.3.2.2 Temporary space for dump files

Each replication site needs temporary space for dump files. describes the storage locations of the source and target dump files. Replication processing generates a dump file at the source site. It then stores it as the content of a document in the source repository before transferring it to the target repository. After the dump file arrives at the target site, it is filtered into another dump file specific to the target repository. Owner and ACL information for each document are modified, if necessary, to conform to replication job requirements, and some objects are removed. The temporary disk space available for dump files on the source and target sites must be twice the size of the largest dump files.

If disk space is limited on either the source or target site, consider running the job as a series of smaller dump and load operations.

If you are not planning to run the job using multiple dump and load operations, calculate temporary space. When doing so, assume that at some point a full refresh replication can occur. A full refresh replicates all documents in a single replication request rather than incrementally as they are modified.) Increase the temporary disk space if scheduling requires you to run multiple replication jobs simultaneously. The following table shows the calculations for XYZ Enterprises. The Temp column represents twice the disk space requirement of the largest full-refresh replication job at that site. In this example, Product 2 has the largest content size, so that figure is doubled for the Temp calculation.

Site	Product 1	Product 2	New Product	Storage	Temp	Total
X	200	650	22.76	872.76	1300	2,172.76
Y	200	650	22.76	872.76	1300	2,172.76
Z	-	650	22.76	672.76	1300	1,972.76

3.12.3.4 Job scheduling

A replication job consists of three components:

- Source site processing to generate the dump file
- Dump file transfer
- Target site processing to load the dump file

The processing time required to complete all three steps is the processing window for the job. To verify that your replication schedules are achievable, estimate the processing window for each job.

In the XYZ Enterprises example, Jobs 1, 2, and 3 can be scheduled for off-peak server usage times because they are run weekly. It is unlikely that these jobs will conflict with other uses of the servers. On the other hand, Jobs 4, 5, and 6 are scheduled to run every two hours. These jobs are running during peak load times for the source and target servers. If the processing window exceeds two hours, the schedule cannot be maintained.

To assess this possibility, develop an estimate of elapsed time for the three steps, based on reference metrics and the size of the replication job. Use the CPU metric and the local Setfile metric to estimate the time required for dump file generation at a source. Use the remote Getfile metric to estimate the time required for dump file transfer. Use the CPU metric to estimate the time required by the filter program and dump file load. The sum of these estimates provides an estimate for total processing time.

The following table shows the calculations for Job 4, using the peak reference metrics shown in sample table for reference metrices, and assuming a full refresh of the replicated objects. The metrics measuring documents processed per second are obtained by dividing the size of the server executable by the operation's elapsed

time. The server executable was the file used to generate the metrics. Peak values are used because Job 4 runs throughout the day.

Processing Step	Reference Metric	No. of Objects or File Size	Time
Generate dump file	0.5 docs/second	700	350 seconds
Store dump file	111K/second	22.76 MB	205 seconds
Transfer dump file	4.4K/second	22.76 MB	5172 seconds
Filter dump file	0.43 docs/second	700	301 seconds
Load dump file	0.43 docs/second	550	237 seconds
Total			6265 seconds (1 hr, 44 min, 25 sec)

3.12.3.4.1 Handling overlapping jobs

Overlapping jobs are two replication jobs that replicate the same object either into one repository or into and out of the same repository. For example, suppose you have two replication jobs with the same source and target repositories. If the execution times overlap and the jobs happen to replicate some of the same objects, errors can occur. Or, suppose one job A replicates into a target repository YY and another job B that replicates from repository YY. In such cases, if the execution times overlap, it is possible that job B attempts to replicate objects that job A is loading. This operation is an error.

The recommended way to avoid such clashes is to group jobs that can overlap into a job sequence. A job sequence is a set of jobs that are executed in a user-defined order. If a job in the sequence has a predecessor in the sequence, that job is not executed until its predecessor completes successfully. For more information about job sequences, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

3.12.3.5 Site setup

Establishing and following enterprise-wide standards for groups, document types, and security is critical to the success of replication.

Some other aspects of an installation must also be reviewed and any required reconfiguration performed before you start replication jobs. The following sections describe these considerations.

3.12.3.5.1 Connection broker setup and validation

To use all the features associated with object replication, each participating site must project to the connection broker at each of the other participating sites. However, if you want only to create read-only replicas in the target repository, then cross-projection is not required.

This section describes how to set up cross projection. It can be helpful to have a listing of the repositories already known to your local connection broker before setting up cross projection.

To add a new repository to a cross-projection environment:

1. Modify the server configuration object for the new repository to add connection broker projection targets for each remote site that is a potential replication target.

After you modify the server configuration object, reinitialize the server for the changes to take effect. Use Documentum Administrator to modify the server configuration object. For information, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

2. Modify the server configuration objects in existing repositories that are potential replication sources to add a connection broker projection target for the new repository.

After you modify the server configuration object, reinitialize the server for the changes to take effect. Use Documentum Administrator to modify the server configuration object. For information, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

3.12.3.5.2 Macintosh access protocol

If you intend to replicate Macintosh documents, all participating repositories must be using the same Macintosh access protocol. The protocol controls header, trailing, and type creator information that is processed to and from the resource fork when the client Foundation Java API is processing the resource fork to and from the server.

The access protocol is set in the `mac_access_protocol` property of the repository's configuration object. Although the replication setup scripts check for consistency, confirm that your sites are consistent before the installation. To determine which protocol a repository is using, execute the following DQL query:

```
SELECT "mac_access_protocol" FROM "dm_docbase_config"
```

3.12.3.5.3 Disk space for temporary files

Replication creates temporary files in both the source and target repositories. The largest temporary files are the dump files. The temporary files are stored in the %DOCUMENTUM%\share\temp\replicate (Windows) \$DOCUMENTUM/share/temp/replicate (Linux) directory in each installation. The replication operation creates a directory underneath this directory for each repository participating in replication on that machine. Make sure that this directory is located on a file system with enough space for the temporary files.

On Linux, if the directory is not located on a file system with enough space for these temporary replication-related files, you can allocate a separate area and create a Linux link, called replicate, to the share/temp area. For example:

```
% cd /u107/dm/* file system with adequate space */
% mkdir dmtemp % cd $DOCUMENTUM/share/temp
% ln -s /u107/dm/dmtemp replicate
% cd replicate <create directories for all repositories on site that will participate
in replication and set all permissions for the installation owner>
```

3.12.3.5.4 Content storage

Disk capacity for replication is a critical issue.

On the source side, the documents associated with the replication dump files are stored in a storage area called replicate_temp_store. This area is created by default in the same directory as the default SysObject file store area (filestore_01).

Where content is stored in the target repository is dependent on the replication mode chosen for the job.

3.12.3.5.5 Cabinets and folders

Object replication jobs replicate objects linked to a particular folder or cabinet called the source folder. The source folder is identified in the job definition. The replicated objects are stored in the target repository in a particular folder or cabinet designated as the target folder in the job definition. Both the source folder and the target folder must exist before you define the replication job. When you create these folders (or cabinets), keep in mind the following code page requirements:

- If the participating repositories have the same server_os_codepage value, the names of the source and target folders must be compatible with that code page.
- If the participating repositories have different server_os_codepage values, the names of the source and target folders must consist of only ASCII characters.

The replication process does not replicate the source folder or cabinet in the target repository's destination folder or cabinet. However, the process does replicate the folder hierarchy found within the source folder or cabinet.

If the source folder or cabinet contains a reference link, the replication process replicates the reference link but does not replicate any objects contained within the object pointed to by the reference link.

If two repositories are exchanging documents using folders with the same names and subfolder structures, name the subfolders identically in both repositories. For example, suppose that repository1 and repository2 both have /Product folders that are used for replication. In repository1, /Product has a subfolder named market_outlook, while in repository2, /Product has a sub folder named marketoutlook. The names of the subfolders only differ by one character, the underscore. However, because the names do not match exactly, after bidirectional replication both repositories have the following folder paths:

- repository1
 - /Product/market_outlook, which contains source documents
 - /Product/marketoutlook, which contains replicated documents
- repository2
 - /Product/market_outlook, which contains replicated documents
 - /Product/marketoutlook, which contains source documents

This operation was not the desired result. If subfolders are named consistently, all documents, whether replicas or not, appear in one folder in both the source and target repositories.

Additionally, Make sure that all users participating in the replication have permission to link documents into the source folder.

3.12.3.6 Defining jobs

New jobs are defined using Documentum Administrator. You must be a superuser to define a replication job.

Before you create a job, the target folder or cabinet for that job must exist. The server only allows you to specify an existing folder or cabinet as the target when you are defining a job. If the folder or cabinet does not exist when you define the job, save the unfinished job definition. Then create the folder or cabinet and finish the job definition.

3.12.3.6.1 Guidelines for all jobs

Observe the following guidelines when creating a job:

- To define a push replication job, create the job in the source repository. To define a pull replication job, create the job in the target repository. If you want the job run by a mediator repository, create the job in the mediator repository.
- If the underlying databases of the repositories are using different code pages, set the Codepage field for the replication job to UTF8. If the databases are using the same code page, set the field to that code page.
- Choosing fast replication can make the job run faster. However, it accomplishes the faster speed by limiting which related objects are replicated.

- If you define multiple jobs with the same target and source repositories, all jobs on the source repository must run under the same user account.
- If the job fails, you can set a flag in the job definition that directs the server to restart the replication job. However, we recommend that you Make sure that your job is correctly defined and running smoothly before you set that flag. The restart feature is intended to recover jobs that fail because some factor in their environment, such as the source or target repository, is unavailable.

3.12.3.6.2 Guidelines for multidump file jobs

Observe the following guidelines when defining a job that uses multiple dump and load operations to perform the replication:

- Set the -objects_per_transfer argument in the method arguments to number of objects you want to transfer in each operation.
It is recommended that you do not specify fewer than 1000 objects in the argument.
- You must define the job as a fast replication job.
- You cannot define the job as a manual transfer job.

3.12.3.6.3 Setting up tracing

Setting the trace level to any value except 0 turns on tracing for the replication job. Setting the argument to 0 turns off tracing for the job.

All tracing information appears in one file whether you are using one dump and load operation or multiple dump and load operations for the job.

3.12.3.7 Manual dump file transfers

If you manually transfer the replication job dump file to the target repository, place the dump file in the following location on the target repository:

- Windows:
`%DOCUMENTUM%\share\temp\replicate\<target_db_name>`
- Linux:
`$DOCUMENTUM/share/temp/replicate/<target_db_name>`

When you put the dump file in this location, be sure to give the file the same name as the original dump file. Within 60 minutes, the replication job agent detects the presence of the file and automatically loads the file.



Note: If the replication job is using multiple dump and load operations, you cannot use a manual file transfer.

3.12.4 Best practices for object replication

Observe the following rules regarding object replication to preserve structural and data consistency in the participating repositories:

1. If a folder is a target folder in one replication job and a source folder in another replication job, do not schedule the two jobs to run concurrently or overlapping.
2. Do not change job settings for the source and target after a job has been run.
3. If the same documents are replicated to multiple target folders in the same repository, the parameters of each job must match and the jobs must not run concurrently or overlapping. Version mismatch errors can occur if the jobs are run concurrently.
4. To delete a replica from a target repository, first move the source object out of the source folder. If there are multiple jobs replicating the object to that target repository, remove the object from each source folder that replicates to that target repository. Then, use a Prune method and delete the replica's entire version tree in the target repository.
5. Do not create local folders in or link local folders to a target folder.
6. Do not move a replicated folder in a target repository to another location.
It is acceptable to link a replicated folder to another, local folder or cabinet. However, do not remove the primary link to the target folder.
7. To delete a folder that contains replicas, first remove the replicas from the folder.

3.13 Managing single-repository models

This section contains procedures for administering the building blocks of a single-repository distributed configuration.

3.13.1 Adding a distributed component

Use this procedure if you are adding a new remote site to an existing single-repository distributed installation.

To add a new site to a distributed storage area installation:

1. Make sure that the installation at the new site complies with the guidelines
2. Make sure that there is enough disk space at the new site to support the content storage requirements of a distributed storage area.
3. At the new site:
 - a. Install a remote Content Server and Accelerated Content Services server.
Installing Documentum CM Server and the Accelerated Content Services server creates the required server configuration object for the new servers. The Accelerated Content Services configuration object updates the

acs.properties file for the new Accelerated Content Services server and creates a file store storage area at the remote site.

- b. Start the remote site's server.
- c. Add the file store storage area at the remote site to the distributed store storage area as a component.
- d. Add the new component to the **Far Stores** list in the server configuration object for each of the other component sites in the distributed storage area.
- e. Optionally, reset the proximity values for the remote Content Server.

The installation process automatically sets a remote server's proximity to its local connection broker as 9001 and its proximity to the primary Documentum CM Server as 9010. You can reset the value to reflect your network's topography.

4. Make sure that users and groups accessing distributed documents from the new site are authenticated by servers at all sites, using the same mechanism.

In a configuration using a distributed storage area, when a user connects to a repository, the client sends two connection requests: one to the data server and one to the remote Content Server. Each server must be able to authenticate the user's name and password using the same authentication mechanism.

5. Make sure that the machine hosting the Documentum CM Server is using UTC time and is synchronized with the other machines hosting distributed components.

Servers at the remote sites use global login tickets configured to be valid for 5 minutes to connect to the primary site to fetch content. If the clocks in the host machines are not synchronized, the connection can fail if the host machine finds the login ticket invalid if the time difference is greater than 5 minutes.

6. To set up repository access for web-based clients at the remote site:

- a. Define at least one network location for the Accelerated Content Services server at the site.
- b. Add the network location to the Accelerated Content Services configuration object.

Add the location to the Accelerated Content Services configuration object for each Accelerated Content Services server that can service requests from that network location.
- c. Define the network location's proximity to the Accelerated Content Services server or servers.
- d. Define the Accelerated Content Services server's projection to a connection broker.

3.13.2 Removing a distributed component

It is possible to remove a site from a distributed storage area. Removing a site does not destroy the underlying directory or the files stored at the site. Use the content storage facilities of Documentum Administrator to remove a component from a distributed storage area.

3.13.3 Removing files from component storage areas

When a user deletes a document from the repository, regularly scheduled repository maintenance removes the associated files from the storage areas. You can remove a file from a component storage area without removing the copies in other component areas and without destroying the associated SysObject, content object, or replica record object.

To remove a file without removing the associated objects, use the DELETE_REPLICA administration method. DELETE_REPLICA removes a file from a storage area and modifies the replica record object associated with the file's content object. If you remove the file using operating system commands, the replica record still indicates that the file exists in the storage area. This operation can cause errors when users try to access the file.

DELETE_REPLICA has one argument, STORE. This string argument identifies the storage area that contains the file you want to remove.

Using Documentum Administrator is the recommended way to execute DELETE_REPLICA. However, you can also use DQL EXECUTE or an apply method.

3.13.3.1 Using DQL EXECUTE

If you use DQL EXECUTE, the syntax is:

```
EXECUTE delete_replica FOR <content_object_id>WITH store = '<storage_area_name>'
```

Specify the object ID of the file's associated content object. *<storage_area_name>* is the name of the storage object for the component storage area from which you are removing the file.

3.13.4 Troubleshooting surrogate get

The surrogate get feature provides automatic on-demand content replication within the components of a distributed storage area. It is intended for configurations that are not using shared directories.

3.13.4.1 Tracing surrogate get

You can use the trace file to trace dm_SurrogateGet method operations. You can also turn on Trace Launch to trace the method's invocations.

3.13.4.1.1 The trace file

If surrogate tracing is turned on, all dm_SurrogateGet method operations and timing information are traced in a file called sugg.trc. The file is stored in %DOCUMENTUM%\bin (Windows) or \$DOCUMENTUM/bin (Linux).

3.13.4.1.2 Turning on trace file generation

The sugg.trc file is not generated by default. Manually turn on tracing for SurrogateGet.

To trace surrogate get operations:

1. Open the mthd6.ebs script in a text editor.

The script is typically stored in %DM_HOME%\install\admin (Windows) or \$DM_HOME/install/admin (Linux).

2. Remove the single quote (') at the beginning of the following line:

```
'ret% = dmAPIExec("trace,s0,10,sugg.trc")
```

3. Save your changes and exit the editor.

3.13.4.1.3 Tracing method invocations

To trace method invocations for surrogate get, set the Trace Launch property on the property page for the job in Documentum Administrator.

After you turn on Trace Launch for the method, the system logs dm_SurrogateGet method's command line in the server log file whenever the method is invoked.

3.13.4.2 Resolving problems

If the on-demand content replication of the SurrogateGet method is not working properly, check the following configuration items:

- The only_fetch_close property
- The get_method properties
- Connection broker projection targets
- Time settings on the participating host machines

3.13.4.2.1 The `only_fetch_close` property

The `only_fetch_close` property is defined for the distributed store object. It controls whether a server invokes the surrogate get method to fetch content files from component storage areas defined as far for the server.

Make sure that this property is set to TRUE. When the property is FALSE, the surrogate get method is not invoked.

If Trace Launch is turned on for SurrogateGet, you can view the server log file to determine if the surrogate get method is invoked when the server fetches a content file stored in a far storage area. The method's command line is recorded in the server log file when the method is invoked.

3.13.4.2.2 The `get_method` properties

The server invokes the method defined in the `get_method` property for the distributed store object and the component storage areas. All must contain `dm_SurrogateGet` if you are using the SurrogateGet tool or the name of your user-defined surrogate get method. Check the property's value and modify it if necessary in the distributed store object and in the component storage areas.

You can use Documentum Administrator to check the property's value.

3.13.4.2.3 Connection broker projection targets

Make sure that the server at the site where the content file is stored is projecting correct connection information to the connection broker used by the server invoking the SurrogateGet method.

The connection broker used by the server invoking SurrogateGet must know about the server at the site where the content file is stored. This means that the server at each site in a single-repository distributed configuration must project its connection information to the connection broker at each of the other sites in the configuration.

3.13.4.2.4 Time settings

If the job is failing, make sure that the host machine on which the job runs and the host machine of the Documentum CM Server to which the job is connecting are using UTC time. In addition, make sure that the clocks are synchronized.

Surrogate get uses a login ticket to connect to servers to obtain content files. The login tickets are valid for 5 minutes. If the machine on which the dm_SurrogateGet method is running and the host machine of the server to which it connecting are not synchronized in time, the ticket can be invalid (timed out) on the target due to perceived differences in time. In such cases, the job fails.

3.13.5 Overriding remote Content Server use

The remote Content Servers improve repository query performance for desktop users at remote sites. However, you can override this functionality on occasion. For example, most the administration methods that manipulate content files and storage areas require the client to use one server for both content and data requests.

There are two ways to override the remote Content Server functionality:

- Set the dfc.content.use_content_server key in the `dfc.properties` file to FALSE
- Specify a server when you issue the connection request

3.13.5.1 Using the `use_content_server` key

The `dfc.content.use_content_server` key is an optional key in the `dfc.properties` file. If you set the key to FALSE, desktop clients always connect to one server for both data and content requests. The key is TRUE by default.

With one exception, TRUE causes the client to connect to two different servers for data and content requests. The exception occurs when the connection request specifies the local server.

3.13.5.2 Using the connection request

Foundation Java API allows you to specify a repository using syntax that identifies which server to use for connection. The server configuration object identifies the name of the server.

The server is identified in the connection request. The desktop client application use that server for content and data requests when `dfc.content.use_content_server` is NULL or FALSE. When `dfc.content.use_content_server` is TRUE, the server identified in the connection request is used instead of the remote Content Server only if it is the user's local server.

3.13.5.3 Using both `use_content_server` key and the connection request

To illustrate how the `dfc.content.use_content_server` key setting interacts with a server specification in a connection request, assume that an enterprise has two servers, `cat` and `mouse`, installed on different machines and that:

- `cat`'s proximity to the connection broker is defined as 2.
- `mouse`'s proximity to the connection broker is defined as 9001

The following table describes the interaction between the `use_content_server` setting and a server specification in the connection request.

Connection request	Data server	remote Content Server	Explanation
<code>use_content_server</code> is not set			
<code>connect, <repositoryname>, <username>,<password></code>	<code>cat</code>	<code>mouse</code>	The connection request does not identify a server and <code>use_content_server</code> is not set, so the client connects to <code>cat</code> as the data server and <code>mouse</code> as the remote Content Server.
<code>connect, <repositoryname>.cat, <username>,<password></code>	<code>cat</code>	<code>cat</code>	The connection request identifies a server and <code>use_content_server</code> is not set to TRUE, so the client uses <code>cat</code> for both data and content requests.
<code>connect, <repositoryname>.mouse <username>,<password></code>	<code>mouse</code>	<code>mouse</code>	The connection request identifies a server and <code>use_content_server</code> is not set to TRUE, so the client uses <code>mouse</code> for both data and content requests.
<code>use_content_server=TRUE</code>			

Connection request	Data server	remote Content Server	Explanation
connect, <repositoryname>, <username>,<password>	cat	mouse	The connection request does not identify a server and use_content_server is set to TRUE, which reinforces the default behavior. The client connects to cat as the data server and mouse as the remote Content Server.
connect, <repositoryname>.cat, <username>,<password>	cat	mouse	The connection request specifies a server for the connection. However, the client connects to two different servers because the use_content_server key is TRUE.
connect, <repositoryname>.mouse <username>,<password>	mouse	mouse	The connection request identifies the local server, which is also the default remote Content Server. Consequently, the client uses mouse for both data and content requests.
use_content_server=FALSE			
connect, <repositoryname>, <username>,<password>	mouse	mouse	The connection request does not identify a server. Because use_content_server is FALSE, the client does not have to connect to different servers for data and content requests. Consequently, the client connects to the local server for both data and content requests.

Connection request	Data server	remote Content Server	Explanation
connect, <repositoryname>.cat, <username>,<password>	cat	cat	The connection request identifies a server. The client connects to the server for both data and content requests because use_content_server is FALSE, so there is no requirement to use different servers.
connect, <repositoryname>.mouse <username>,<password>	mouse	mouse	The connection request identifies a server. The client connects to mouse for both data and content requests because use_content_server is FALSE, so there is no requirement to use different servers.

3.13.6 Login failures in remote Content Server setups

When remote Content Servers are in an installation, remote clients connect to both a data server and a remote Content Server by default when a session starts. Each connection requires a separate user authentication.

If a login failure occurs at the data server site, the failure occurs when the user tries to log in to the repository.

If a login failure occurs at the remote Content Server site, the user is logged in to the repository, but all content transfer requests fail.

3.13.7 Tuning query performance

As users add documents to a distributed storage area, the dmi_replica_record tables in the repository grow. This growth affects the performance of queries against the documents in the distributed storage area. For best performance, make sure that you are updating the statistics for the RDBMS tables underlying the repository. The query optimizer uses statistics to select the best query plan.

You can generate statistics using either the Documentum Update Statistics system administration tool or the statistics tool provided by the underlying RDBMS. Update Statistics generates statistics for all tables in the repository. The RDBMS tools typically allow you to specify the table or tables for which you want to generate statistics.

For information about Update Statistics, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*. Consult the documentation provided by your RDBMS vendor for information about running statistics directly in the RDBMS.

3.14 Managing multi-repository models

This section describes how to perform common administrative tasks for multi-repository distributed configurations.

3.14.1 Manipulating a federation

This section contains procedures for managing a federation.

3.14.1.1 Adding a member

A repository can belong to only one federation. Consequently, the member you are adding cannot currently belong to any federation.

Use the federation management facilities of Documentum Administrator to add a member repository to a federation. For more information, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

3.14.1.2 Removing a member

You cannot remove a member repository while any federation jobs are running. Removing a member from a federation updates the federation object. When a federation job is running, the federation object is locked and cannot be updated.

Removing a member repository also sets the dm_FederationImport job in that repository to inactive and hidden. The a_is_hidden property is set to TRUE for the job.

It is not necessary to make the global objects in the repository local objects. However, if you intend to add the repository to a different federation, we recommend that you make the global objects in the repository local objects first.

Use the federation management facilities of Documentum Administrator to remove a member repository from a federation. For more information about the instructions, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

3.14.1.3 Destroying a federation

You cannot destroy a federation while any federation jobs are running. Destroying a federation updates the federation object. When a federation job is running, the federation object is locked and cannot be updated.

After you remove a federation, the former member repositories still have global users and groups. This operation does not affect the functioning of these repositories. However, we recommend that you make the global users and groups in these repositories local users and groups before adding the repositories to a new federation.

Use the federation management facilities of Documentum Administrator to destroy a federation. For more information about the instructions, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

3.14.1.4 Inactivating a governing repository

Inactivating a governing repository halts all federation-related jobs. To deactivate the governing repository, use the federation management facilities of Documentum Administrator. For more information about the instructions, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

3.14.2 User operations

This section contains information for managing the global users within a federation. Use the information in this section only if the global users are common only to the repositories participating in the federation.

A federation's global users are users who are present in all repositories in the federation. Global users are managed through the governing repository. You create a new global user in the governing repository, and the federation jobs automatically propagate the user to all member repositories. Similarly, if you delete a global user from the governing repository, a federation job automatically deletes the user from the member repositories.

Most of a user's properties are global properties. Global property values are the same in each repository in the federation. Global properties must be modified through the governing repository, using Documentum Administrator.

A few properties are local properties. Typically, local properties have different values in each member repository. However, there are four local properties that can behave like global properties. These four properties are `user_state`, `client_capability`, `workflow_disabled`, and `user_privileges`. When you create a global user, you can select to propagate the values of these properties to all member repositories when the properties are changed in the governing repository.



Note: Sysadmin and Superuser user privileges are never propagated even if you select to have the `user_privileges` properties managed as a global property. Extended user privileges are not propagated either.

Perform the procedures using Documentum Administrator if your site is not using an LDAP directory server. The procedures are performed on the governing repository, through the federation management facilities of Documentum Administrator.

If your site is using an LDAP directory server to implement global users across all repositories, use the procedures in the LDAP directory server documentation to add or change global user entries in the directory server. Do not use Documentum Administrator.

3.14.2.1 Creating a user

Use the federation management facilities of Documentum Administrator to add a global user to a governing repository. For more information about the instructions, see *OpenText Documentum Content Management - Administrator User Guide* (EDCAC250400-UGD). The online help also has information about user properties.

Use the following guidelines when creating a user:

- The name of the user must be unique among the user and group names in the governing repository.
- A global user's user_name, user_os_name, user_address, and user_db_name must be compatible with the server operating system code pages (server_os_codepage) of all participating repositories. If the repositories are using different code pages, this means that those properties must contain only ASCII characters.
- If a member repository has a local user with the same name as the global user, the global user overwrites the local user.
- Assigning a global external ACL to the user as the user's default ACL is recommended.



Note: If the ACL assigned to the user does not exist on the member repository, the server uses a system ACL named Global User Default ACL to create a custom ACL for the user. The permissions in Global User Default ACL are dm_world=3 (Read), dm_group=5 (Version), and dm_owner=7 (Delete). Documentum CM Server:

- Determines to which group the user belongs.
 - Creates a custom ACL from the Global User Default ACL that references that group.
 - Assigns that ACL to the user.
- If you check a Send to members? checkbox, the value of the associated local property is propagated to member repositories, with one exception. The exception occurs for user privileges and extended user privileges. If the assigned user privilege is Sysadmin or Superuser or any of the extended user privileges, even if the Send to members? checkbox is checked the privilege value is not propagated.

- OpenText Documentum CM client applications use the client capability setting to determine what functionality to make available to the user. Documentum CM Server does not use the client capability setting. The client documentation contains more information on the functionality provided with each capability level.

After you create a global user on the governing repository, the federation jobs automatically propagate the new user to the member repositories. The global property values assigned in the governing repository are copied to each member repository. The local properties are assigned default values. The following table describes the default values assigned to the local properties in the member repositories:

Property	Default setting
user_db_name	A null string
default_folder	If the member repository is the user's home repository, the default_folder property is set to the default_folder value specified when the user was created in the governing repository. If the repository is not the user's home repository, default_folder is set to /Temp.
user_privileges	The privilege level assigned when the user was created in the governing repository, unless the user was given Sysadmin or Superuser privileges. Superuser and Sysadmin privileges are not propagated automatically, which ensures that member repositories do not acquire unexpected users with these privileges. Only users who are installation owners retain Sysadmin or Superuser privileges when they are propagated to member repositories. All other users with Superuser or Sysadmin privileges in the governing repository are given no privileges in the member repositories (user_privileges=0).
user_state property	The user_state property is set to the value assigned when the user was created.
alias_set_id	The alias_set_id property is not assigned a default value.

3.14.2.2 Modifying user information

You can perform the following modifications for users:

- Modify global and local property values for a user
- Change the user's default ACL
- Change the user's state
- Change the user's home repository

Modify global users in the governing repository. For more information about the instructions, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

3.14.2.3 Renaming a user

Renaming a user affects more than just the user's dm_user object. The server stores a user's name in various properties for different object types. For example, r_creator_name and owner_name values are user names. User names also appear as values for the acl_domain and authors properties.

When you run the job that renames a user, it attempts to change all references to that user in the repository. If some of the references are in the properties of locked (checked out) objects, by default the job unlocks the objects (cancels the checkout) and makes the change. If you do not want the job to unlock these objects, be sure to set the Checked Out Objects preference to Ignore before running the job.

You can select whether to run renaming job immediately or the next time jobs are executed. In a large repository, the rename job can take a long time to complete. It touches many tables and can use a large amount of resources. Running the job during peak usage hours is not recommended.

Rename a user through the governing repository. Use the federation management facility of Documentum Administrator. For more information about the instructions, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

3.14.2.4 Making a local user global

To change a local user to a global user you can use Documentum Administrator or DQL.

3.14.2.4.1 Using Documentum Administrator

If the user is local to the governing repository, you can simply change the user to global in the governing repository. The user is then propagated to all members in the next update operation.

If the user is local to a member repository, you must create the user as a new global user in the governing repository for the federation. The new user is automatically propagated to all the member repositories in the next update operation. As part of the process, the user is updated to global in the original member repository also.

3.14.2.4.2 Using DQL

To use DQL to change a local user to a global user, use the UPDATE...OBJECT statement to set the value of the globally_managed property to TRUE for the user:

```
UPDATE "dm_user" OBJECT SET "globally_managed"=TRUE WHERE "user_name"='<local_user_name>'
```

3.14.2.5 Making a global user local

To change a global user to a local user, use Documentum Administrator. You make the change in the federation's governing repository. The next federation update operation updates the user information in the member repositories, making the user local.

3.14.2.6 Deleting a global user

Use Documentum Administrator to delete a global user. Use the federation management facilities to connect to the federation and delete a global user. For more information about the instructions, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*. Deleting a user also deletes all registry objects that reference the user as the subject of an audit or event notification request.

Renaming the user is recommended instead of deleting the user.

3.14.3 Group operations

This section contains the information about procedures for managing global groups within a federation.

Perform the procedures using Documentum Administrator if your site is not using an LDAP directory server. The procedures are performed on the governing repository, through the federation management facilities of Documentum Administrator.

If your site is using an LDAP directory server to implement global groups across all repositories, use the procedures in the LDAP directory server documentation to add or change global group entries in the directory server.

3.14.3.1 Creating a group

All the properties of a group are global. The values you define for a global group when you create it are propagated to all member repositories. The members of a global group must be global users or other global groups. To create a group, you must have Create Group, Sysadmin, or Superuser privileges.

A global group's group_address must be compatible with the server operating system code pages (server_os_codepage) of all participating repositories. If the repositories are using different code pages, group_address must consist of only ASCII characters.

The name of the group must be unique among the user and group names in the governing repository.



Note: Documentum CM Server stores all group names in lowercase.

If a member repository has a local group with the same name as the global group, the global group overwrites the local group.

You can use the federation management facilities in Documentum Administrator to create a global group. For information, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

Alternatively, you can execute DQL CREATE GROUP and UPDATE...OBJECT statements using IDQL or Documentum Administrator. After you create the group, update it to set the globally_managed property to TRUE. The syntax is:

```
CREATE GROUP <name_of_group>WITH MEMBERS <list_of_members>
UPDATE "dm_group" OBJECT SET "globally_managed"=TRUE WHERE "group_name"='<name_of_group>'
```

3.14.3.2 Modifying a group

To modify a group, you must be one of the following:

- The group's owner
- A superuser
- A member of the group that owns the group to be modified
- Identified in the group's group_admin property, either as an individual or a member of the group specified in the property

Only a superuser can change the ownership of an existing group. Only a superuser, the owner of a group, or a member of the group that owns the group can change the group_admin property of a group.

Use the federation management facilities in Documentum Administrator to modify a global group.

3.14.3.3 Renaming a group

To rename a global group, use Documentum Administrator. For information, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

3.14.3.4 Deleting a group

To delete a global group, use Documentum Administrator. You must be a superuser, the group's owner, or a member of the group that owns the group to delete a group. For information, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*. Deleting a group also deletes all registry objects that reference the group as the subject of an audit or event notification request.

Renaming the group is recommended instead of deleting the group.

3.14.3.5 Making a global group local

Documentum CM Server does not support directly changing a global group to a local group. Instead you create a matching local group and then delete the global group.

3.14.4 Modifying object replication jobs

You can modify a job after it is created. You can change the frequency with which a job is executed, suspend a job, and resume a job.

You can also change the settings for some properties that affect the replicas created by the job. Changing these settings only affects replicas created after the change. Replicas created by previous executions of the job are not affected. For example, if you change the ACL assigned to replicas, subsequent executions assign the new ACL to the replicas created by those executions, but existing replicas created by previous executions retain the original ACL.

If you want to change all replica objects in the target cabinet or folder, delete the replica objects and perform a full-refresh execution of the job.



Caution

Never change the source or destination folder. Running a job stores context information about the source and target cabinets and folders. If you change these parameters, define a new job.

Use the job management facilities in Documentum Administrator to change a job. For more information about the instructions, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

3.14.5 Obtaining a list of jobs

Use the job management facilities in Documentum Administrator to view a list of jobs defined for a particular repository. You can view all jobs in the repository or the jobs defined by a category. For example, you can view those jobs that operate on the repository as a whole or only content management jobs. For more information about the instructions, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

3.14.6 Scheduling federation jobs

Federation jobs manage global users, groups, and external ACLs in the federation. These jobs are executed automatically, in a sequence controlled by the dm_FederationUpdate job.

Federation jobs and the methods they invoke are installed as part of the Documentum CM Server installation process. The jobs are installed in the inactive state with the is_hidden property set to TRUE.

When you create a federation, only certain jobs are activated. These jobs are the dm_FederationUpdate and dm_FederationStatus jobs in the governing repository. If you check **Propagate Global Objects to Member Repositories** when you create the federation, they are activated and made visible automatically. Otherwise, do it manually.

All other federation jobs in the governing repository and member repositories remain inactive and invisible. the dm_FederationUpdate job running in the governing repository controls their execution. By default, the dm_FederationUpdate job runs once a day, during the night. When it executes, it runs the methods associated with the jobs in the correct sequence for each member repository.

You can change the scheduling for the dm_FederationUpdate job on the governing repository. However, we strongly recommend that you do not activate any of the other federation jobs. Doing so causes them to run out of sequence.



Note: To avoid federation job failures, include the %DM_HOME%\bin on Windows or the \$DM_HOME/bin on Linux path in the CLASSPATH environment variable.

3.14.7 Identifying the federation jobs operator

The federation jobs operator receives email sent by the system reporting the status of the federation jobs. By default, this user is identified in the operator_name property of the server's server configuration object. The value in operator_name defaults to the repository owner.

If the federation administrator is not the user identified in operator_name, reset the queueperson argument for the federation jobs to the federation administrator's user name. Use Documentum Administrator to display the Properties page for each job and reset the queueperson argument.

3.14.8 Tracing ACL replication in federation jobs

The dm_FederationUpdate job calls the dm_ACL_Repl`<repository_name>` job to replicate ACLs to member repositories. There is one dm_ACL_Repl job for each member repository. By default, the tracing level in the dm_ACL_Repl jobs is set to 0. To turn on tracing for the ACL replication jobs, set the method_trace_level argument for the jobs.

The tracing information is recorded in the job report generated by the federation update job.

3.14.9 Job reports and log files

Federation jobs and object replication jobs generate reports and log files.

3.14.9.1 Job reports

A job report summarizes the results of a job in an easily interpreted format. To view reports for federation jobs, you connect to the federation. To view job reports for object replication jobs, connect to the repository that executed the job. Use Documentum Administrator to view job reports.

In the repository, job reports are stored in /System/Sysadmin/Reports. Each time a job executes, the job report in this location is versioned.

The reports are also stored in the file system in %DOCUMENTUM%\dba\log\<repository_id>\sysadmin (Windows) or \$DOCUMENTUM/dba/log/<repository_id>/sysadmin. In this directory, the report name has the format <job_name>Doc.txt. The reports in this directory are overwritten each time the job executes.

Federation job reports are also stored in %DOCUMENTUM%\share\temp\ldif\<repository_name> (Windows) or \$DOCUMENTUM/share/temp/ldif/<repository_name> for access through Documentum Administrator.

3.14.9.2 Job log files

Each job execution generates a log file in addition to a report. The log file is typically used for debugging. If you want to know how long a job took to complete or why a job did not run to completion, look in the log file.

For object replication jobs, the log file information includes:

- Whether the job is working in Phase 1, 2, or 3
 - Phase 1 is when the dump file is created
 - Phase 2 is when the file is transferred to the target
 - Phase 3 is when the dump file is loaded into the target repository
- The current job status

- The elapsed time of the job
- The incremental time of the job.
- The incremental time is the length of time since the last status message.
- The elapsed time is the length of time since the job began.

To view federation job log files, you connect to the federation. To view the log files for object replication jobs, connect to the repository that executed the job. Use Documentum Administrator to view job log files.

In the repository, job log files are stored in /Temp/Jobs/<job_object_name>. The log files in this directory are versioned when the jobs execute.

The log files are also stored in the file system in %DOCUMENTUM%\dba\log \<repository_id>\sysadmin (\$DOCUMENTUM/dba/log/<repository_id>/sysadmin). The name of the log file has the format <job_name>Trace.txt. The log file in this directory is overwritten each time the job executes.

Federation job log files are also stored in %DOCUMENTUM%\share\temp\ldif \<repository_name> (\$DOCUMENTUM/share/temp/ldif/<repository_name>) for access through Documentum Administrator.

3.14.10 The dm_DistOperations job

The dm_DistOperations job performs inter-repository distributed operations. These tasks include:

- Propagating distributed events (dmi_queue_items) across repositories
- Creating checkout references for remote checkout operations
- Refreshing reference links

This job is one of the system administration tools installed with the Documentum CM Server installation. For more information about other tools, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

Like replication jobs, the dm_DistOperations jobs generates reports and log files. The report is saved in the repository in /System/Sysadmin/Reports/DistributedOperations.

The dm_DistOperations job runs continuously in each repository, polling every 5 minutes for operations it performs. The following table describes the job's arguments:

Argument	Value	Description
-process_queued_operations	Boolean	If TRUE, the job processes all dmi_queue_item objects that have remote_pending set to TRUE. The default value is TRUE.
-process_refreshes	Boolean	If TRUE, the job checks for and performs needed reference links refreshes. The default value is TRUE.
-refresh_checks_per_cycle	integer	Defines the maximum number of reference links to check for needed refresh in each job execution. The default value is 100.
superuser and passwords for each repository	string	The job must have a valid superuser name and password for each of the participating repositories. These values are entered as arguments using the format: $<\text{server_config_name} \text{ user_name}>,<\text{password}>[,<\text{domain}>]$ server_config_name is the object name of the server configuration object for the repository. The superuser and password must be valid for the repository. They can be the same or different for each repository. domain is required only when unique domain user is enforced.

3.14.11 Monitoring and debugging federation jobs

All federation jobs return one of three values: success, warning, or failure.

A warning indicates that the job encountered a non-fatal error. The job continued to execute but some errors can have occurred during its execution. Warnings typically occur in the following situations:

- The status or copy methods cannot establish a session with a member repository to retrieve the status or copy global objects.
- The status or copy methods cannot validate the dm_federation object in a member repository.
- The import method on a member repository experienced difficulties such as:
 - The default folder could not be created
 - The user's default ACL was not found, necessitating the use of the default system ACL

Failures indicate a fatal error that stopped the job's execution. Fatal errors can leave the dm_federation object locked in the repository. In such cases, manually check in the federation object.

If a warning or failure occurs:

1. Review the job report.
2. If necessary, review the job log file.

3.14.12 Recovering from replication job failures

Object replication failures generally result from an improperly configured source or target installation or from infrastructure inadequacies, such as insufficient network bandwidth or CPU capacity. If a replication job fails to complete successfully:

- Examine the Properties for the job to determine the last status for the job.
- Review the log file for the job for detailed information on why the job failed.
- If the error message in the trace file suggests that something is configured incorrectly, reconfigure it appropriately and rerun the job.

For example, if the replication job calls for replicas to be stored in a nonexistent file store, create the appropriate file store and rerun the job. For more information about the instructions on running a job outside of its scheduled time, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

- Make certain that adequate machine and network resources were available when the job failed.
- If the message in the log file suggests that there were insufficient resources available, rerun the job to see if the same failure occurs.

In general, replication job failures are not destructive. They do not harm existing replicas in the target repository. The next successful execution of the job will clean up after the failure.

Correcting the cause of the failure restores successful replication. However, as part of the process of correcting the problem and rerunning the job to check it, you can reset the job schedule. For more information about scheduling jobs, see *OpenText Documentum Content Management - Administrator User Guide* (EDCAC250400-UGD).

3.14.13 Clearing the replicate_temp_store storage area

If an object replication job is aborted or incomplete, Documentum CM Server can leave data in the replicate_temp_store storage area. (This area is used to hold the dump files created by object replication jobs for the duration of the job.) You can remove data left behind under such circumstances.

The directory location of the replicate_temp_store storage area for a particular repository is %DOCUMENTUM%\data\replicate_temp_store\<repository_id> (\$DOCUMENTUM/data/replicate_temp_store/<repository_id>), where <repository_id> is the hex value.

3.14.14 Handling replicas

This section contains procedures and information about management of replicas in a repository.

3.14.14.1 Defining a binding label for reference links

The binding between a mirror object and its source object is typically set by default when the user executes the operation that creates the reference link. For example, suppose a user, who is logged in to repository A, checks out the approved version of DocumentX, which resides in repository B. The server creates a reference link in repository A to the approved version of DocumentX.

If the user's operation does not identify a specific version of the object, the server binds the CURRENT version to the reference link by default.

You can override the binding specified at runtime by setting the ref_binding_label property in the session configuration object. For example, suppose ref_binding_label is set to CURRENT before the user checks out the approved version of DocumentX. Even though the user requests the approved version, the server binds the CURRENT version to the reference link. When the user opens the document, the server opens the CURRENT version. The setting in ref_binding_label overrides the binding label identified by the user.

If the version defined in ref_binding_label does not exist, the server returns an error.

To return to the default behavior, set the ref_binding_label property to a blank string.

3.14.14.2 Determining whether an object is a replica

Replica objects are identified in the user interface by a replica icon. Programmatically, you can identify a replica using the computed property `_isreplica`. This property is set to T for replicas. You can query this property using the Get method or the DQL SELECT statement. For DQL, specify the property name without the underscore (`isreplica`).

3.14.14.3 Determining a replica's source

To determine the source of a replica, examine its reference object or query the computed property `_masterdocbase` for the replica. This property returns an object's source repository.

3.15 Federation infrastructure

The table in this appendix describes the jobs, methods, files, and objects that support federations.

Object name	Type	Description
dm_FederationUpdate	job/method	Executes on the governing repository to run all the other methods in sequence, to push changes to users, groups, and ACLs to the member repositories. The job and its associated method have the same name.
dm_FederationStatus	job/method	Polls all the member repositories to determine the status of the dm_FederationImport jobs on the member. The job and its associated method have the same name.
dm_ldif_status	method	Called by dm_FederationStatus to poll the member repositories.
dm_FederationExport	job/method	Exports the user and group information from the governing repository to an LDIF file. The job and its associated method have the same name.

Object name	Type	Description
dm_ldif_export	method	Called by dm_FederationExport to generate an LDIF export file.
dm_FederationCopy	job/method	Transfers LDIF files to member repositories. The job and its associated method have the same name.
dm_ldif_copy	method	Called by dm_FederationCopy to transfer the LDIF files from the governing repository to member repositories.
dm_FederationImport	job/method	Imports an LDIF file into member repositories.
dm_ldif_import	method	Called by dm_FederationImport to import the users and groups from the LDIF file into a member repository.
dm_ACLReplication	job/method	A staging job that sets external ACLs for replication.
dm_ACLRepl_<repository>	job	Replicates external ACLs to member repositories. There is one job for each member repository.<repository> is the first 19 bytes of the repository name.
ldif	directory	Contains the LDIF files generated by dm_FederationExport. The directory is found in %DOCUMENTUM%\share\temp (\$DOCUMENTUM/share/temp).
dm_federation_log	registered table	Tracks certain changes to users and groups. This table is the change log.

Chapter 4

Foundation CMIS API

This documentation is for system administrators or programmers who wish to deploy OpenText Documentum Content Management (CM) Foundation CMIS API. It describes how to deploy Foundation CMIS API to a supported servlet container, as well as information about configuration of the Foundation CMIS API server environment.

4.1 Introduction

Foundation CMIS API is a web application. To deploy Foundation CMIS API, you must deploy a Foundation CMIS API web application archive file to an application server.

Make sure that your system meets the requirements specified in the product *Release Notes*.

With Foundation CMIS API latest version, you can download the latest version of Foundation CMIS API.

The latest version of Foundation CMIS API for Content Management Interoperability Services 1.1 is built on Apache Chemistry OpenCMIS Framework and the framework exposes the following URLs:

URL	Example
Browser binding	http://CMISERVER:8080/dctm-cmis/browser
RESTful AtomPub Binding 1.0	http://CMISERVER:8080/dctm-cmis/resources10
RESTful AtomPub Binding 1.1	http://CMISERVER:8080/dctm-cmis/resources
Web Services Binding 1.0	http://CMISERVER:8080/dctm-cmis/services10/cmis?wsdl
Web Services Binding 1.1	http://CMISERVER:8080/dctm-cmis/services/cmis?wsdl

4.2 Configuration settings

This section provides information on configuration settings that affect Foundation CMIS API, including JVM, Linux, and application properties settings.

4.2.1 General JVM configuration settings

To provide adequate heap space for the Foundation CMIS API web application, OpenText recommends the following JVM settings:

- -Xms512m
- -Xmx512m

4.2.2 Installing JDK

Download and install the supported version of Oracle JDK/OpenJDK version from the Oracle JDK/OpenJDK website. The *Oracle JDK and OpenJDK* documentation contains detailed information.

4.2.2.1 Using SSL communication

“Using SSL communication” on page 36 contains detailed information.

4.2.3 Using urandom generators on Linux systems

There are issues with implementation of pseudo-random number generators on Linux. For more efficient randomization, Linux systems should use urandom generators that are faster but less secure.

To change the source of secure random numbers from random to urandom, set the `java.security.egd` system property as follows:

```
-Djava.security.egd=file:///dev/urandom
```

Specifying this system property overrides the `securerandom.source` setting to `urandom`.

If the application server is on Red Hat Linux, the application server startup script must be modified to set the option in the JVM.

4.2.4 Foundation CMIS API configuration files

Foundation CMIS API uses these configuration files to set properties for different layers of the application:

- `dfc.properties`, which contains property settings for the underlying Foundation Java API client.
- `cmis-runtime.properties`, which includes properties specific to the Foundation CMIS API layer.

4.2.5 Foundation Java API configuration

The `dfc.properties` file provides property settings for the Foundation Java API runtime. This file is located in `WEB-INF/classes`.

The following table describes properties in the `dfc.properties` file that are relevant for Foundation CMIS API. For example, the `dfc.properties` files includes the critical settings that are required for Foundation CMIS API to reach a connection broker and connect to a Documentum CM Server.

Property	Value
<code>dfc.docbroker.host[0]</code>	The fully qualified host name for the connection broker. You can add backup hosts by adding new properties and incrementing the index number within brackets.
<code>dfc.docbroker.port</code>	If you wish to use a port for the connection broker other than the default of 1489, add a port key.
<code>dfc.globalregistry.repository</code>	The global registry repository name.
<code>dfc.globalregistry.username</code>	The user name of the global registry user. The global registry user, who has the default user name <code>dm_bof_registry</code> , must have read access to objects in the <code>/System/Modules</code> and <code>/System/NetworkLocations</code> only.
<code>dfc.globalregistry.password</code>	An encrypted password value for the global registry user.
<code>dfc.search.external_sources.enable</code>	True, to enable Documentum Federated Search Services; false, to disable ECIS. You must specify the Documentum Federated Search Services host machine name in <code>dfc.search.ecis.host</code> .

Property	Value
dfc.search.external_sources.host	Specifies the Documentum Federated Search Services host machine name. You must set <code>dfc.search.ecis.enable</code> to true.
dfc.cache.ddinfo.size	Valid values are 1 to 10000. Controls the memory cache size of the Documentum CM Server data dictionary. This parameter is required for the Foundation CMIS API type definition cache.
dfc.cache.type.currency_check_interval	Valid values are 0 to 86400. This parameter is required for the Foundation CMIS API type definition cache.

You can either copy the user name and encrypted password for the global registry user from the `dfc.properties` file on the global registry Documentum CM Server host, or you can select another global registry user and encrypt the password using the following command:

```
java -cp dfc.jar com.documentum.fc.tools.RegistryPasswordUtils <password_to_be_encrypted>
```

4.2.6 Foundation CMIS API runtime properties

The `cmis-runtime.properties` file enables you to set properties that affect application behavior at the Foundation CMIS API layer. This file is located in `WEB-INF/classes`.

These properties are optional unless otherwise specified, and if not specified defaults to a value documented in the following table. If a supplied value for an integer or Boolean property is invalid, the default value is used instead.

These items are cached:

- Repository MIME types
- Repository object types
- Foundation Java API session service tokens for logged-in users

Name	Description	Default value	Permissible values range
security.configuration.file	Required. File name of security (XWS-Security) configuration for SOAP binding web services.	cmis-security.xml (in <code>cmis-ws-binding.jar</code>)	security file name string

Name	Description	Default value	Permissible values range
cmis.mime_type.cache_expiration_after_x_seconds	<p>Indicates the expiration timeout for MIME type cache.</p> <p>Repository MIME types are cached in memory to help with performance.</p> <p>This property specifies how often the MIME type cache is flushed.</p>	3,600	1 - 8,640,000 (100 days)
cmis.token.cache_expiration_after_x_seconds	<p>Indicates the expiration timeout for service token cache.</p> <p>Service tokens for login users are cached in memory to save the cost of new Foundation Java API sessions.</p> <p>This property specifies how often the service token is flushed.</p>	3,600	1 - 8,640,000 (100 days)

Name	Description	Default value	Permissible values range
cmis.type_info.cache_expiration_after_x_seconds	<p>Indicates the expiration timeout (in seconds) for the Foundation CMIS API type definition cache.</p> <p>When the specified interval has elapsed and if the repository's object types have changed, then the Foundation CMIS API type definition cache is flushed and reloaded with the updated object types from the repository. All requests that require access to the type definition cache are blocked until the cache is reloaded.</p> <p>The repository's object type definitions are cached in memory to improve performance. In addition, object type and property definitions are loaded into the cache lazily.</p> <p>You might need to tune this value to optimize performance for your deployment.</p>	3,600	1 - 8,640,000 (100 days)
cmis.mime_type.cache_size	<p>The cache size for mime type.</p> <p>The cache size should not be less than the repository list size.</p>	10	1 - 10,000

Name	Description	Default value	Permissible values range
cmis.token.cache_size	The cache size for service token. The cache size should not be less than the repository list size.	10	1 - 10,000
cmis.type_info.cache_size	The cache size for Foundation CMIS API type definition. The cache size should not be less than the repository list size.	10	1 - 10,000
cmis.default_max_items	The default maximum number of items in a returned collection. This value is used if the client does not provide a value for maxItems. If value = -1 or value = 0 then the value is set to Integer.MAX_VALUE.	100	-1 - Integer.MAX_VALUE
cmis.max_items_upper_limit	The allowed maximum value for maxItems. This sets an upper limit on maxItems provided by a client. This setting is recommended for system scalability and performance. If value = -1 or value = 0 then the value is set to Integer.MAX_VALUE.	2,000	-1 - Integer.MAX_VALUE

Name	Description	Default value	Permissible values range
cmis.exception.full_message.append	<p>Indicates whether to output error messages from layers below Foundation CMIS API; that is, OpenText Documentum CM error messages.</p> <p>These messages can help to identify the root cause of exceptions.</p>	true	true, false
cmis.anonymous_access.repository[index]	<p>The name of the repository to which to grant anonymous access.</p> <p>If one repository is configured as anonymous accessible, set its repository name here. You can set multiple repositories for anonymous access, or set all available repositories to be anonymously accessible.</p>	Not-Set	valid repository name string
cmis.anonymous_access.principal.username[index]	The OpenText Documentum CM login name to be used for anonymous access to the repository with the same index.	Not-Set	valid user login name string
cmis.anonymous_access.principal.password[index]	The OpenText Documentum CM password for the user login with the same index.	Not-Set	valid user password

4.2.6.1 Anonymous access settings

You can configure a principal to allow access to a single repository, to multiple but not all repositories, or to all available repositories.

To make only one repository anonymously accessible, set the `anonymous_access` properties as follows:

```
cmis.anonymous_access.repository[0]=<reponame>cmis.anonymous_access.principal.username[0]
]=<user name>cmis.anonymous_access.principal.password[0]=<password>
```

To enable anonymous access to multiple repositories, configure each repository by incrementing the index on the properties:

```
cmis.anonymous_access.repository[0]=<reponame>cmis.anonymous_access.principal.username[0]
]=<user
name>cmis.anonymous_access.principal.password[0]=<password>cmis.anonymous_access.repo
tory[1]=<reponame1>cmis.anonymous_access.principal.username[1]=<user
name1>cmis.anonymous_access.principal.password[1]=<password1>
```

If all repositories available to the Foundation CMIS API services allow anonymous access, and if the user name and password for the principal are the same on all repositories, you can use the wildcard, * (asterisk), as follows:

```
cmis.anonymous_access.repository[*]=*
cmis.anonymous_access.principal.username[0]=<user name>
cmis.anonymous_access.principal.password[0]=<password>
```

4.2.6.2 Maximum items default and upper limit settings

The Content Management Interoperability Services specification defines the `maxItems` parameter as the “maximum number of items to return in a response”. Many Foundation CMIS API services/resources support this parameter for paging purposes. Typically, a Foundation CMIS API client provides a `maxItems` setting in requests to such resources and services. However, in cases when the client does not provide a value for `maxItems`, Foundation CMIS API uses a default value. The Foundation CMIS API server administrator can set this default using the `cmis.default_max_items` runtime property.

In some cases a client (perhaps with malicious intent) may set `maxItems` to an excessively large value in a request, which may negatively affect server performance. To guard against this possibility, the Foundation CMIS API server administrator can set an upper limit to `maxItems` in `cmis.max_items_upper_limit`.

If either property has a value of -1 or 0, Foundation CMIS API sets no upper bound on the number of items returned, so that the effective limit is `Integer.MAX_VALUE`. Foundation CMIS API determines the effective `maxItems` value using both of these property settings, as follows:

```
maxItems = MIN(client_or_default_max_items, server_max_items_upper_limit),
where a value of -1 or 0 is treated as equivalent to Integer.MAX_VALUE
```

4.2.6.3 Cross-Origin Resource Sharing filter for configuring sites to access Foundation CMIS API

You must create the Cross-Origin Resource Sharing (CORS) filter to configure the sites that are allowed to access Foundation CMIS API. Perform the following steps:

1. Stop the application server.
2. Edit the `cmis-runtime.properties` file.
3. To enable the CORS filter, set the value of `cmis.cors.filter.enabled` to true.
4. After the CORS filter is enabled, set the `cmis.cors.allowed.origins` property with an appropriate value. Otherwise, the request is rejected. For example:
`cmis.cors.allowed.origins=http://localhost:8080`
5. In addition, provide appropriate values for `cmis.cors.allowed.methods`, `cmis.cors.allowed.headers` and `cmis.cors.exposed.headers`.
6. Start the application server.



Note: When you enable the CORS filter, the origin request header must be included in the HTTP request headers.

4.2.7 Accessing Foundation CMIS API URL in an IPv6 environment

To access the Foundation CMIS API URL in an IPv6 environment, you must provide the IPv6 address within the square brackets in the URL.

4.2.8 Configuring rate limiting

Rate limiting is not set in Foundation CMIS API. You must configure the rate limiting with your infrastructure components such as load balancer or proxy server to prevent any application-level denial-of-service (DoS) attacks.

4.3 Configuring token-based authentication for Browser binding

When using Browser binding, you can configure Foundation CMIS API web application with token-based authentication using HTTPBasicAuthentication.

4.3.1 HttpBasicAuthentication with tokens

To configure HttpBasicAuthentication with tokens for Browser binding, change `web.xml` to specify the class in bold.

```
<servlet> <servlet-name>cmisbrowser</servlet-name>
<servlet-
class>com.emc.documentum.fs.cmis.impl.browser.servlet.EmcCmisBrowserBindingServlet</
servlet-class>
<!-- The default authentication mechanism for browser binding
is HTTP basic authentication with callContextHandler parameter set
to value
com.emc.documentum.fs.cmis.impl.auth.callcontexthandler.HttpBasicAuthCallContextHandler
The following customizations are allowed
on CMIS Documentum 1.1 for Browser binding:
To enable HTTP basic with tokens based authentication, modify
the following callContextHandler parameter by replacing
com.emc.documentum.fs.cmis.impl.auth.callcontexthandler.HttpBasicAuthCallContextHandler
with
com.emc.documentum.fs.cmis.impl.browser.token.impl.ExtendedTokenCallContextHandler
--><init-param> <param-name>callContextHandler</param-name>
<param-
value>com.emc.documentum.fs.cmis.impl.auth.callcontexthandler.HttpBasicAuthCallContextHan
dler</param-value>
</init-param>
<load-on-startup>2</load-on-startup>
</servlet>
```

4.4 Deploying on web application servers

You can deploy a Foundation CMIS API web application archive file on web application servers.

To deploy the Foundation CMIS API web application, deploy the appropriate archive file as shown in the following table:

Application server	Archive file
Apache Tomcat	dctm-cmis.war
JBoss	dctm-cmis-jboss.ear
WildFly	dctm-cmis-jboss.ear
IBM Liberty	dctm-cmis-websphere.ear



Notes

- If you want to hide the exception or stack traces from the response, add the following line to the startup script file of all the supported web application servers:

```
set "JAVA_OPTS=%JAVA_OPTS% -Dorg.apache.chemistry.opencmis.stacktrace.disable"
```

- If the web application server is using JDK 21, no additional Java options are required.

4.4.1 Apache Tomcat

Make sure that you are deploying Foundation CMIS API on a supported version of Apache Tomcat. The product *Release Notes* contains the information about the supported versions.

Make sure that the Tomcat JVM settings meet the recommendations (the Tomcat default settings may not be adequate). The Apache Tomcat website provides detailed information.

Copy the WAR file to the `<TomcatHome>/webapps` folder.

Tomcat extracts the WAR file to the `<TomcatHome>/webapps/<application_name>` folder, where `<application_name>` is the name of the WAR file without the file extension.



Note: For the JDK 17 support, add the following parameters in the `catalina.bat` file:

```
set "JAVA_OPTS=Dprogram.name=%PROGNAME% %JAVA_OPTS%"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-opens=java.base/java.net=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-opens=java.base/java.lang.ref=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-opens=java.naming/com.sun.jndi.toolkit.url=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-exports=java.base/sun.security.provider=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-exports=java.base/sun.security.pkcs=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-exports=java.base/sun.security.x509=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-exports=java.base/sun.security.util=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-opens=java.xml.crypto/com.sun.org.apache.xml.internal.security  
=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-opens=java.base/java.lang=ALL-UNNAMED"
```

For the Linux environment, add the following parameters in the `#Add the module start-up parameters` section:

```
JAVA_OPTS="$JAVA_OPTS  
--add-opens=java.base/java.net=ALL-UNNAMED"  
JAVA_OPTS="$JAVA_OPTS  
--add-opens=java.base/java.lang.ref=ALL-UNNAMED"  
JAVA_OPTS="$JAVA_OPTS
```

```
--add-opens=java.naming/com.sun.jndi.toolkit.url=ALL-UNNAMED"
JAVA_OPTS="$JAVA_OPTS
--add-exports=java.base/sun.security.provider=ALL-UNNAMED"
JAVA_OPTS="$JAVA_OPTS
--add-exports=java.base/sun.security.pkcs=ALL-UNNAMED"
JAVA_OPTS="$JAVA_OPTS
--add-exports=java.base/sun.security.x509=ALL-UNNAMED"
JAVA_OPTS="$JAVA_OPTS
--add-exports=java.base/sun.security.util=ALL-UNNAMED"
JAVA_OPTS="$JAVA_OPTS
--add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED"
JAVA_OPTS="$JAVA_OPTS
--add-opens=java.xml.crypto/com.sun.org.apache.xml.internal.security
=ALL-UNNAMED"
```

4.4.2 Configuring Red Hat JBoss Enterprise Application Platform (JBoss EAP)

Before deploying the Foundation CMIS API EAR file, perform the following steps:

1. Download the JBoss EAP 8 Update 4 patch file named `jboss-eap-8.0.4-runtime-maven-repository.zip` from the Red Hat website.
2. Extract the `jboss-eap-8.0.4-runtime-maven-repository.zip` file to a temporary location.
3. Apply the JBoss EAP 8 Update 4 patch.

Windows

For example, from `<JBoss EAP_Home>/bin`, run the following command as an administrator:

```
jboss-eap-installation-manager.bat update perform ^
--dir C:\jboss-eap-8.0 ^
--repositories mrrc::file:C:\jboss-eap-8.0.4.GA-runtime-maven-repository
\maven-repository ^
--offline
```

Linux

For example, from `<JBoss EAP_Home>/bin`, run the following command as an administrator:

```
./jboss-eap-installation-manager.bat update perform \
> --dir ../../jboss-eap-8.0 \
> --repositories mrrc::file:C:\jboss-eap-8.0.4.GA-runtime-maven-repository
\maven-repository \
> --offline
```

4. Extract the Foundation CMIS API EAR file:

```
jar -xvf <CMIS_EAR_JBOSS>
```

5. Open the `WEB-INF/classes` folder and edit the `dfc.properties` file with the correct connection broker details.

6. Create a JAR file for all the files in the APP-INF/classes folder and copy these JAR file into the lib folder.

```
jar -cvf classes.jar.
```

7. Create the EAR file.

```
jar -cvf dcm-cmis-jboss.ear.
```



Notes

- If you are deploying CMIS 1.1, then add the following entries into jboss modules\system\layers\base\sun\jdk\main\module.xml file under <dependencies> <paths> element:

```
<path
  name="com/sun/org/apache/bcel/internal"/> <path name="com/sun/org/apache/bcel/internal/classfile"/>
<path name="com/sun/org/apache/bcel/internal/generic"/> <path
  name="com/sun/org/apache/bcel/internal/util"/> <path name="com/sun/org/apache/
  regexp/internal"/>
<path name="com/sun/org/apache/xalan/internal"/> <path name="com/sun/org/apache/
  xalan/internal/extensions"/>
<path name="com/sun/org/apache/xalan/internal/lib"/> <path name="com/sun/org/apache/
  xalan/internal/res"/>
<path name="com/sun/org/apache/xalan/internal/templates"/> <path
  name="com/sun/org/apache/xalan/internal/utils"/> <path name="com/sun/org/apache/
  xalan/internal/xslt"/>
<path name="com/sun/org/apache/xalan/internal/xsltc"/> <path
  name="com/sun/org/apache/xalan/internal/xsltc/cmdline"/> <path
  name="com/sun/org/apache/xalan/internal/xsltc/cmdline/getopt"/> <path
  name="com/sun/org/apache/xalan/internal/xsltc/compiler"/> <path
  name="com/sun/org/apache/xalan/internal/xsltc/compiler/util"/> <path
  name="com/sun/org/apache/xalan/internal/xsltc/dom"/> <path name="com/sun/org/apache/
  xalan/internal/xsltc/runtime"/>
<path name="com/sun/org/apache/xalan/internal/xsltc/runtime/output"/>
<path name="com/sun/org/apache/xalan/internal/xsltc/trax"/> <path
  name="com/sun/org/apache/xalan/internal/xsltc/util"/> <path name="com/sun/org/
  apache/xerces/internal/dom"/>
<path name="com/sun/org/apache/xerces/internal/dom/events"/> <path
  name="com/sun/org/apache/xerces/internal/impl"/> <path name="com/sun/org/apache/
  xerces/internal/impl/dtd"/>
<path name="com/sun/org/apache/xerces/internal/impl/dtd/models"/>
<path name="com/sun/org/apache/xerces/internal/impl/dv"/> <path
  name="com/sun/org/apache/xerces/internal/impl/dv/dtd"/> <path name="com/sun/org/
  apache/xerces/internal/impl/dv/util"/>
<path name="com/sun/org/apache/xerces/internal/impl/dv/xs"/> <path
  name="com/sun/org/apache/xerces/internal/impl/io"/> <path name="com/sun/org/apache/
  xerces/internal/impl/msg"/>
<path name="com/sun/org/apache/xerces/internal/impl/validation"/>
<path name="com/sun/org/apache/xerces/internal/impl/xpath"/> <path
  name="com/sun/org/apache/xerces/internal/impl/xpath/regex"/> <path
  name="com/sun/org/apache/xerces/internal/impl/xs"/> <path name="com/sun/org/apache/
  xerces/internal/impl/xs/identity"/>
<path name="com/sun/org/apache/xerces/internal/impl/xs/models"/>
<path name="com/sun/org/apache/xerces/internal/impl/xs/opti"/>
<path name="com/sun/org/apache/xerces/internal/impl/xs/traversers"/>
<path name="com/sun/org/apache/xerces/internal/impl/xs/util"/>
<path name="com/sun/org/apache/xerces/internal/jaxp"/> <path
  name="com/sun/org/apache/xerces/internal/jaxp/datatype"/> <path
  name="com/sun/org/apache/xerces/internal/jaxp/validation"/> <path
  name="com/sun/org/apache/xerces/internal/parsers"/> <path name="com/sun/org/apache/
  xerces/internal/util"/>
<path name="com/sun/org/apache/xerces/internal/utils"/> <path
  name="com/sun/org/apache/xerces/internal/xinclude"/> <path name="com/sun/org/apache/
  xerces/internal/xni"/>
<path name="com/sun/org/apache/xerces/internal/xni/grammars"/>
<path name="com/sun/org/apache/xerces/internal/xni/parser"/> <path
```

```

name="com/sun/org/apache/xerces/internal/xpointer"/> <path name="com/sun/org/apache/
xerces/internal/xs"/>
<path name="com/sun/org/apache/xerces/internal/xs/datatypes"/>
<path name="com/sun/org/apache/xml/internal/dtm"/> <path name="com/sun/org/
apache/xml/internal/dtm/ref"/>
<path name="com/sun/org/apache/xml/internal/dtm/ref/dom2dtm"/>
<path name="com/sun/org/apache/xml/internal/dtm/ref/sax2dtm"/>
<path name="com/sun/org/apache/xml/internal/res"/> <path name="com/sun/org/
apache/xml/internal/resolver"/>
<path name="com/sun/org/apache/xml/internal/resolver/helpers"/>
<path name="com/sun/org/apache/xml/internal/resolver/readers"/>
<path name="com/sun/org/apache/xml/internal/resolver/tools"/> <path
name="com/sun/org/apache/xml/internal/security"/> <path name="com/sun/org/
apache/xml/internal/security/algorithms"/>
<path name="com/sun/org/apache/xml/internal/security/algorithms/implementations"/>
<path name="com/sun/org/apache/xml/internal/security/c14n"/> <path
name="com/sun/org/apache/xml/internal/security/c14n/helper"/> <path
name="com/sun/org/apache/xml/internal/security/c14n/implementations"/>
<path name="com/sun/org/apache/xml/internal/security/encryption"/>
<path name="com/sun/org/apache/xml/internal/security/exceptions"/>
<path name="com/sun/org/apache/xml/internal/security/keys"/> <path
name="com/sun/org/apache/xml/internal/security/keys/content"/> <path
name="com/sun/org/apache/xml/internal/security/keys/content/keyvalues"/>
<path name="com/sun/org/apache/xml/internal/security/keys/content/x509"/>
<path name="com/sun/org/apache/xml/internal/security/keys/keyresolver"/>
<path name="com/sun/org/apache/xml/internal/security/keys/keyresolver/
implementations"/>
<path name="com/sun/org/apache/xml/internal/security/keys/storage"/>
<path name="com/sun/org/apache/xml/internal/security/keys/storage/implementations"/>
<path name="com/sun/org/apache/xml/internal/security/signature"/>
<path name="com/sun/org/apache/xml/internal/security/transforms"/>
<path name="com/sun/org/apache/xml/internal/security/transforms/implementations"/>
<path name="com/sun/org/apache/xml/internal/security/transforms/params"/>
<path name="com/sun/org/apache/xml/internal/security/utils"/> <path
name="com/sun/org/apache/xml/internal/security/utils/resolver"/> <path
name="com/sun/org/apache/xml/internal/security/utils/resolver/implementations"/>
<path name="com/sun/org/apache/xml/internal/serialize"/> <path
name="com/sun/org/apache/xml/internal/serializer"/> <path name="com/sun/org/
apache/xml/internal/serializer/utils"/>
<path name="com/sun/org/apache/xml/internal/utils"/> <path
name="com/sun/org/apache/xml/internal/utils/res"/> <path name="com/sun/org/apache/
xpath/internal"/>
<path name="com/sun/org/apache>xpath/internal/axes"/> <path
name="com/sun/org/apache>xpath/internal/compiler"/> <path name="com/sun/org/apache/
xpath/internal/domapi"/>
<path name="com/sun/org/apache>xpath/internal/functions"/> <path
name="com/sun/org/apache>xpath/internal/jaxp"/> <path name="com/sun/org/apache/
xpath/internal/objects"/>
<path name="com/sun/org/apache>xpath/internal/operations"/> <path
name="com/sun/org/apache>xpath/internal/patterns"/> <path name="com/sun/org/apache/
xpath/internal/res"/>

```

- If your JDK version is 17, add the following parameters in the standalone.bat file:

```

set "JAVA_OPTS=-Dprogram.name=%PROGNAME% %JAVA_OPTS%"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.net=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.lang.ref=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.naming/com.sun.jndi.toolkit.url=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports java.base/sun.security.provider=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports java.base/sun.security.pkcs=ALL-UNNAMED"

```

```
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports java.base/sun.security.x509=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports java.base/sun.security.util=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports java.base/sun.security.tools.keytool=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.xml.crypto/com.sun.org.apache.xml.internal.security
=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.lang=ALL-UNNAMED"
```

- Creating or updating documents with large content (greater than 10 MB) on a JBoss application server is not supported by default. To enable this support, add the *max-post-size= "104857600"* parameter in the *http-listener* element of the *standalone.xml* file as follows:

```
<http-listener name="default" socket-binding="http" redirect-socket="https" max-
post-size="104857600"/>
```

To create secondary types in Foundation CMIS API with JBoss:

1. Extract *dctm-cmis-jboss.ear*.
2. Copy *cmis-sectype-aspect-api.jar* and *cmis-sectype-aspect-impl.jar* to the *CMIS_CUSTOM_JAR_DIR* in *C:\jboss-eap-8.0\standalone* folder.
3. Copy the path to *CMIS_CUSTOM_JAR_DIR*.
4. Set an environmental variable in *C:\jboss-eap-8.0\bin\standalone.conf.bat* file in line number 53 as follows: *CMIS_CUSTOM_JAR_DIR=C:\jboss-eap-8.0\standalone\CMIS_CUSTOM_JAR_DIR*.

4.4.3 WildFly

Before deploying the Foundation CMIS API EAR file, perform the following steps:

1. Extract the Foundation CMIS API EAR file:

```
jar -xvf <CMIS_EAR_WILDFLY>
```

2. Open the *WEB-INF/classes* folder and edit the *dfc.properties* file with the correct connection broker details.
3. Add the following parameters in the *standalone.bat* file for the JDK 11 support:

```
set "JAVA_OPTS=%JAVA_OPTS% -Djdk.io.File.enableADS=true"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.net=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.lang.ref=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.naming/com.sun.jndi.toolkit.url=ALL-UNNAMED"
```

```
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.provider=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.pkcs=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.x509=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.util=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.xml.crypto/com.sun.org.apache.xml.internal.security
=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.lang=ALL-UNNAMED"
```

For the Linux environment, add the following parameters in the `standalone.sh` file:

```
JDK_JAVA_OPTIONS="$JDK_JAVA_OPTIONS
--add-opens=java.base/java.lang=ALL-UNNAMED"
JDK_JAVA_OPTIONS="$JDK_JAVA_OPTIONS
--add-opens=java.base/java.io=ALL-UNNAMED"
JDK_JAVA_OPTIONS="$JDK_JAVA_OPTIONS
--add-opens=java.base/java.util=ALL-UNNAMED"
JDK_JAVA_OPTIONS="$JDK_JAVA_OPTIONS
--add-opens=java.base/java.util.concurrent=ALL-UNNAMED"
JDK_JAVA_OPTIONS="$JDK_JAVA_OPTIONS
--add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED"
JDK_JAVA_OPTIONS="$JDK_JAVA_OPTIONS
--add-opens=java.base/java.net=ALL-UNNAMED"
JDK_JAVA_OPTIONS="$JDK_JAVA_OPTIONS
--add-opens=java.base/java.lang.ref=ALL-UNNAMED"
JDK_JAVA_OPTIONS="$JDK_JAVA_OPTIONS
--add-opens=java.naming/com.sun.jndi.toolkit.url=ALL-UNNAMED"
JDK_JAVA_OPTIONS="$JDK_JAVA_OPTIONS
--add-exports=java.base/sun.security.provider=ALL-UNNAMED"
JDK_JAVA_OPTIONS="$JDK_JAVA_OPTIONS
--add-exports=java.base/sun.security.pkcs=ALL-UNNAMED"
JDK_JAVA_OPTIONS="$JDK_JAVA_OPTIONS
--add-exports=java.base/sun.security.x509=ALL-UNNAMED"
JDK_JAVA_OPTIONS="$JDK_JAVA_OPTIONS
--add-exports=java.base/sun.security.util=ALL-UNNAMED"
JDK_JAVA_OPTIONS="$JDK_JAVA_OPTIONS
--add-exports=java.base/sun.security.tools.keytool
=ALL-UNNAMED"
JDK_JAVA_OPTIONS="$JDK_JAVA_OPTIONS
--add-opens=java.base/java.lang=ALL-UNNAMED"
export JDK_JAVA_OPTIONS
```

Java options must be added in # Set default modular JVM options.

4. Create a JAR file for all the files in the APP-INF/classes folder and copy these JAR file into the lib folder.

```
jar -cvf classes.jar
```

5. Create the EAR file.

```
jar -cvf dcm-cmis-wildfly.ear
```

To create secondary types in Foundation CMIS API with Wildfly

1. Extract dcm-cmis-jboss.ear.
2. Copy cmis-sectype-aspect-api.jar and cmis-sectype-aspect-impl.jar to the CMIS_CUSTOM_JAR_DIR in C:\jboss-eap-8.0\standalone folder.
3. Copy the path to CMIS_CUSTOM_JAR_DIR.
4. Set an environmental variable in C:\jboss-eap-8.0\bin\standalone.conf.bat file as follows:
CMIS_CUSTOM_JAR_DIR=C:\jboss-eap-8.0\standalone\CMIS_CUSTOM_JAR_DIR.

4.4.4 IBM Liberty

Before deploying the Foundation CMIS API EAR file, perform the following tasks:

1. Extract the dcm-cmis-websphere.ear file.
2. Take a backup of the opencmis-dcm-connector-<release-version>-SNAPSHOT.jar file.
3. Copy all the files available in the dcm-cmis.war\WEB-INF\classes folder.
4. Navigate to the dcm-cmis-websphere.ear\lib folder.
5. Extract the opencmis-dcm-connector-<release-version>-SNAPSHOT.jar file.
6. Add the following parameters in the server.bat file for the JDK 17 support:

```
set "JAVA_OPTS=-Dprogram.name=%PROGNAME% %JAVA_OPTS%"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-opens=java.base/java.net=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-opens=java.base/java.lang.ref=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-opens=java.naming/com.sun.jndi.toolkit.url=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-exports=java.base/sun.security.provider=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-exports=java.base/sun.security.pkcs=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-exports=java.base/sun.security.x509=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-exports=java.base/sun.security.util=ALL-UNNAMED"
```

```

set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.xml.crypto/com.sun.org.apache.xml.internal.security
=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.lang=ALL-UNNAMED"

```

For the Linux environment, add the following parameters in the server file:

```

JAVA_HOME="/home/IBM/jdk-17.0.8" #JAVA_HOME path
JAVA_ARGS="-Dcom.ibm.jsse2.overrideDefaultTLS=true"
JAVA_OPTS="-Dprogram.name=%PROGNAME% %JAVA_OPTS%"
JDK_JAVA_OPTIONS
="--add-exports=java.base/jdk.internal.ref=ALL-UNNAMED
--add-opens=java.base/java.lang=ALL-UNNAMED
--add-opens=java.base/java.nio=ALL-UNNAMED
--add-opens=java.base/sun.nio.ch=ALL-UNNAMED
--add-opens=java.base/java.util=ALL-UNNAMED
--add-exports=java.naming/com.sun.jndi.ldap=ALL-UNNAMED
--add-opens=java.base/java.text=ALL-UNNAMED
--add-opens=java.naming/com.sun.jndi.toolkit.url=ALL-UNNAMED
--add-exports=java.base/sun.security.x509=ALL-UNNAMED
--add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED"

```

7. Update the `server.xml` file as follows:

```

<featureManager>
<feature>microProfile-6.0</feature>
<feature>adminCenter-1.0</feature>
</featureManager>
...
...
<cdi12 enableImplicitBeanArchives="false" />

```

8. Place the copied files ([step 3](#)) in the extracted JAR file.

To create secondary types in Foundation CMIS API with IBM Liberty:

1. From the `dctm-cmis-websphere.ear` file, copy `cmis-sectype-aspect-api.jar` and `cmis-sectype-aspect-impl.jar`.
2. Add the copied files in `CMIS_CUSTOM_JAR_DIR` folder in `C:\Program Files\IBM\WebSphere\Liberty\usr\servers\dctm-cmis`.
3. Copy the path to `CMIS_CUSTOM_JAR_DIR`.
4. Set an environmental variable in `C:\Program Files\IBM\WebSphere\Liberty\usr\servers\dctm-cmis\server.env` file as follows: `CUSTOM_DIR_PATH=C:\Program Files\IBM\WebSphere\Liberty\usr\servers\dctm-cmis\CUSTOM_DIR_PATH`.

4.4.5 Configuring Foundation CMIS API to connect to HashiCorp Vault

1. Perform all the steps as described in “[Configuring Documentum Secret Integration Service](#)” on page 49 and make sure that DSIS is running.
2. While deploying Foundation CMIS API on the supported application servers, update the `dfc.properties` file with the following entries:

```
dfc.dsds.enabled=true  
dfc.dsds.daemon.url=http://localhost:8200/dsis  
dfc.dsds.daemon.token=<token generated for the DSIS daemon agent>
```

For more information about setting the preceding entries in the `dfc.properties` file, see *OpenText Documentum Content Management - Foundation Java API Development Guide (EDCPKCL250400-DGD)*.

3. Redeploy the WAR or EAR file.
4. Restart the application server.

4.5 Post-deployment tasks

This section describes deployment validation and the Foundation CMIS API service addresses.

4.5.1 Licensing OpenText Documentum CM

OpenText Documentum CM uses OpenText Directory Services (OTDS) to apply licenses for all the OpenText Documentum CM components. For more information about procuring the license file and configuring OTDS and license, see “[Licensing OpenText Documentum CM](#)” on page 86.

4.5.2 Validation

On successful deployment, you should be able to access the Foundation CMIS API home page at the following URL:

```
http://<host>:<port>/<contextPath>
```



Note: The application context path will vary depending on your deployment. In most deployments the default context path is `dctm-cmis`, based on the name of the archive file.

4.5.3 RESTful AtomPub service document

The service document defining the RESTful AtomPub binding can be obtained from this address:

```
http://<host>:<port>/<contextPath>/resources/
```

4.5.4 Web service entry points

You can view the WSDL for any of the SOAP web services by using a URL like the following:

```
http://<host>:<port>/<contextPath>/services/RepositoryService?wsdl
```

The WSDL files for each of the Foundation CMIS API web services are essentially identical: each one defines endpoints for all of the Foundation CMIS API web services, which are shown in the following table:

Web Service	Address
ACLService	http://<host>:<port>/<contextPath>/services/ACLService
DiscoveryService	http://<host>:<port>/<contextPath>/services/DiscoveryService
MultiFilingService	http://<host>:<port>/<contextPath>/services/MultiFilingService
NavigationService	http://<host>:<port>/<contextPath>/services/NavigationService
ObjectService	http://<host>:<port>/<contextPath>/services/ObjectService
RelationshipService	http://<host>:<port>/<contextPath>/services/RelationshipService
RepositoryService	http://<host>:<port>/<contextPath>/services/RepositoryService
VersioningService	http://<host>:<port>/<contextPath>/services/VersioningService

4.5.5 Browser binding URLs

The document returned by the following Service URL provides the repository information for all available repositories: `http://<host>:<port>/<contextPath>/browser/`

Each repository information must contain the following two additional properties:

- `repositoryUrl`: The Repository URL
- `rootFolderUrl`: The Root Folder URL

4.6 Enabling OTDS authentication in Foundation CMIS API

Foundation CMIS API client applications can authenticate OTDS users by providing user and password information in the Foundation CMIS API request as follows:

- Password-based authentication: Provide the user name and password in the following format:

```
<user_login_name>,dm_otds_password=<user_password>
```

- Token-based authentication: Provide the user name and password in the following format:

```
<user_login_name>,dm_otds_ticket=<otds_token>
```

Chapter 5

Foundation Java API

This documentation describes how to install Foundation Java API.

This guide is intended for programmers or system administrators who must install Foundation Java API. For Windows systems, it assumes general familiarity with Windows operation. For Linux systems it assumes general familiarity with shells, permissions, and environment variables.

5.1 Prerequisites

This section describes the steps you must perform before installing Foundation Java API.

Any OpenText Documentum CM client product that uses Foundation Java API installs Foundation Java API. If both Foundation Java API and its client applications are compiled with the same Java version, you can upgrade Foundation Java API without upgrading the client applications.



Caution

If you previously installed additional classes on top of Foundation Java API in the Foundation Java API program root directory, check whether those classes are still available after you upgrade Foundation Java API. For example, you may have customized your system and created custom classes and installed them in the Foundation Java API shared directory after installing Foundation Java API.

Whenever you upgrade Foundation Java API, you must check if your additional classes need to be reinstalled after the upgrade.

Foundation Java API relies on certain environment variables. Make sure that these variables have correct values. “[Establishing the environment for Foundation Java API](#)” on page 368 explains how to set the environment variables.

Foundation Java API requires administrator privileges to run. At installation, the installer verifies that the installation account has those privileges.

5.1.1 Removing old files

This section explains the situations in which you must remove an older version of Foundation Java API, and possibly other programs as well, before installing Foundation Java API.

You can install Foundation Java API directly over Foundation Java API version 5.1 or later.

Install the new Foundation Java API without uninstalling the old one if the following conditions are true:

- The Foundation Java API installation you want to upgrade has a version number of 5.1 or later.
- The Foundation Java API installation is on a Documentum CM Server host.

In all other cases, uninstall the old Foundation Java API before installing the new one.

[“Uninstalling Foundation Java API” on page 380](#) explains how to remove Foundation Java API.

5.1.2 Upgrading client programs

Programs that use Foundation Java API are called client programs. Upgrading Foundation Java API does not affect client programs that use Foundation Java API version 5.3 or later.

In order to run more than one version of Foundation Java API on a Linux system, you must arrange to run the different Foundation Java API versions in different processes. Set the environment variables described in [“Establishing the environment for Foundation Java API” on page 368](#) and install the different versions of Foundation Java API in locations that you can distinguish from one another by those environment settings.

5.1.3 Establishing the environment for Foundation Java API

This section explains how to set the environment variables that Foundation Java API relies on.

[“Environment variables that Foundation Java API uses” on page 371](#) lists the environment variables that Foundation Java API relies on. To set the variables:

- Windows: Installation program sets the environment variables.

The Foundation Java API installation program for Windows sets environment variables. The only additional setting you need to perform is to add the JAR files to the classpath if you need to refer to Foundation Java API classes and interfaces in your Java programs. [“Foundation Java API classes location” on page 370](#) provides more details about what to place on the classpath.

- Linux: You set environment variables.

For Linux systems, the installation program does not set environment variables. If the installation program does not find the needed environment variables, it aborts the installation.

For Linux systems, the way to set environment variables depends on the shell that you use. Make sure that you set the variables in such a way that a process launched in a different shell has the same values defined. This means using `setenv` or `export` (depending on the shell). Do not use `set`, which defines variables only for the current shell, but not for any child shell.

5.1.3.1 Defining file system locations for Foundation Java API components

Foundation Java API maintains components at different file system locations. The following sections provide details about the locations Foundation Java API uses.

5.1.3.1.1 Foundation Java API program root directory

Foundation Java API installs program files in the program root directory path.

On Windows systems, the installation program asks for a program root directory. It uses `C:\Program Files\Documentum` if you do not specify a location.

On Linux systems, the installation program uses the `DOCUMENTUM_SHARED` environment variable to determine the program root directory. The installation program terminates the installation if it finds this variable undefined.

5.1.3.1.2 Foundation Java API user root directory

Foundation Java API creates client-oriented directories (for example, `checkout` and `export`) in the user root directory.

On Windows systems, the installation program asks for a user directory root and uses `C:\Documentum` if you do not specify a location.

On Linux systems, the installation program uses the `DOCUMENTUM` environment variable to determine the user directory root. The installation program terminates the installation if it finds this variable undefined.

5.1.3.1.3 Shared libraries directory

The installation program places shared libraries at specific locations relative to the program root directory.

On Windows systems, the installation program uses the shared subdirectory of the program root directory. It attaches the full path of this directory (followed by a separator character) in front of the value of the PATH system environment variable.

On Linux systems, the installation program uses the `dfc` subdirectory of the program root directory. You must place the full path of this directory onto the library path. The library path environment variable is `LD_LIBRARY_PATH` for Linux.

Environment variables can be set on Linux systems using the `setenv` script. The script can be found at `$DOCUMENTUM_SHARED/dfc/set_dctm_env.sh (.csh)`. You can source this file to properly set the environment variables from “[Environment variables that Foundation Java API uses](#)” on page 371.

5.1.3.1.4 Foundation Java API configuration files directory

The installation program creates the `config` directory to store configuration files. “[Using the Foundation Java API config directory](#)” on page 372 provides information about Foundation Java API configuration files. The installation program creates the `config` directory in the program root directory path on Linux systems and in the user root directory path on Windows systems. For Foundation Java API to operate properly, the full path to the `config` directory must appear in the classpath.

On Windows systems, the installation program attaches the full path of the `config` directory (followed by a separator character) in front of the value of the `CLASSPATH` system environment variable.

On Linux systems, you must place the full path of the `config` directory onto the classpath. For example, in the syntax of the `csh` shell, attach `$DOCUMENTUM_SHARED\config:` to the value of the `CLASSPATH` environment variable. You can do this before or after running the installation program, because the installation program does not use this setting.

5.1.3.1.5 Foundation Java API classes location

The Java runtime environment uses the `CLASSPATH` environment variable to find Foundation Java API classes and the `config` directory.

On a Windows system, the installation program places the full paths to `dctm.jar` and the `config` directory (with appropriate separators) at the front of the classpath.

On a Linux system, the installation program does not modify the classpath. You must place the full paths of `dctm.jar` and the `config` directory onto the classpath.

For both Windows and Linux systems, you must perform an additional step if you want the `javac` compiler to have access to Foundation Java API classes. The `javac` compiler does not recognize the JAR files specified in the manifest contained in `dctm.jar`.

5.1.3.2 Setting environment variables

Foundation Java API uses several environment variables to find its components. “[Defining file system locations for Foundation Java API components](#)” on page 369 describes the file system locations that the environment variables point to. On Windows systems, the installation program asks you for the information that it uses to set these variables. On Linux systems, you must set these variables before you run the installation program. “[Environment variables that Foundation Java API uses](#)” on page 371 lists these environment variables and summarizes the ways that Foundation Java API uses them.

Table 5-1: Environment variables that Foundation Java API uses

Variable	How Foundation Java API uses it	Windows value (installation program sets)	Linux value (you set)
DOCUMENTUM_SHARED	Determine the full path to the program root directory for Linux.	Not used by Windows systems.	Specify a value before installing Foundation Java API.
PATH	Find the directory containing Foundation Java API shared libraries (DLLs) on Windows.	Attach the full path (followed by a separator character) in front of the shared subdirectory of the OpenText Documentum CM program root.	Not used by Linux systems.
LD_LIBRARY_PATH	Find the directory containing Foundation Java API shared libraries on Linux.	Not used by Windows systems.	Add \$DOCUMENTUM_SHARED/dfc.
DFC_DATA	Deprecated variable.	“ Foundation Java API configuration files directory ” on page 370 provides information about what you should do instead of using this variable.	
DOCUMENTUM	Determine the full path to the user root directory.	Not used by Windows systems.	Specify a value before installing Foundation Java API.

Variable	How Foundation Java API uses it	Windows value (installation program sets)	Linux value (you set)
CLASSPATH	Allow Java runtime to find dctm.jar and the Foundation Java API config directory. “ Foundation Java API classes location ” on page 370 provides information about making Foundation Java API classes available to the javac compiler.	Attach (with appropriate separator characters) the full paths of dctm.jar and the config directory (for example, C:\Documentum\Shared\dctm.jar and C:\Documentum\config).	Add \$DOCUMENTUM_SHARED/dctm.jar and \$DOCUMENTUM_SHARED/config.



Note: Enable the DM_UPDATE_VDM_HIGH_LINK_COUNT environment variable and set it to 1. This allows server to update this property, which Foundation Java API uses to calculate the correct order number when more than seven nodes are to be inserted in a virtual document.

5.1.4 Using the Foundation Java API config directory

The Foundation Java API config directory contains Java properties files that control the behavior of Foundation Java API. The installation program creates the config directory if it does not already exist. “[Configuration files for Foundation Java API](#)” on page 372 describes the files in the config directory.

Table 5-2: Configuration files for Foundation Java API

File	Description
dfc.properties	Current configuration options for Foundation Java API.
dfcfull.properties	Template containing all possible configuration options. Do not modify this file. Copy sections into dfc.properties, as required.
log4j2.properties	Current configuration options for the log4j instance that underlies the unified logging system. For more information about the logging system, see <i>OpenText Documentum Content Management - Foundation Java API Development Guide (EDCPKCL250400-DGD)</i> .

File	Description
dbor.properties	Registry for business objects prior to the 5.3 release. Do not edit this file. For more information about how to use this file, see <i>OpenText Documentum Content Management - Foundation Java API Development Guide (EDCPKCL250400-DGD)</i> .

Each line of the Java properties file is either a comment (begins with #) or contains a statement of the form `key=value`, where `key` and `value` are character strings that comply with ISO 8859-1 encoding. For characters that do not comply with ISO 8859-1, use Unicode escapes. These are of the form \u followed by the four hexadecimal digits that represent the Unicode encoding (for example, \u2297) of the character.

The key cannot contain white space. The value can contain spaces and other special characters, but you must precede each with a backslash (\) character. For example, to indicate that the Foundation Java API configuration files are in C:\Documentum\ User Files\config, you can include the line:

```
dfc.data.dir=C:\\\\Documentum\\\\User\\\\Files
```

A backslash precedes each colon, backslash, or space.

The Java convention for expressing file paths allows you to write the same line as:

```
dfc.data.dir=C\\\\\\\\Documentum\\\\User\\\\Files
```

You do not need to precede a forward slash with an escape character.

After installation, the `dfc.properties` file contains the `dfc.data.dir` key. The corresponding value is the full path to the Foundation Java API data directory.

At a minimum, `dfc.properties` also contains the following keys:

- `dfc.docbroker.host[0]`
- `dfc.docbroker.port[0]`
- `dfc.tokenstorage.dir`
- `dfc.tokenstorage.enable`

5.1.5 Using Java 2 security

If you plan to use the Java 2 security, archive the policy file that you create at the startup of your Java Virtual Machine (JVM). For a standalone installation, specify the file in the command line startup arguments for your JVM. If you are deploying on an application server, refer to the application server documentation for the specifics of specifying Java 2 security policies. The Foundation Java API example policy template file is named `dfc.example.java.policy`, and is found in the `dfc.jar` file installed in the `DOCUMENTUM_SHARED` directory (by default, in Windows systems, `C:\Program Files\Documentum\Shared`).

If you deploy Foundation Java API on WebSphere Application Server, you must set the `dfc.security.enforce_existing_policy` JVM runtime system property before policy configuration.

Name	Description	Type	Required	Default
<code>dfc.security.enforce_existing_policy</code>	Specifies whether to enforce the existing security policy settings. true: Foundation Java API enforces the existing security policy settings and policy configuration is skipped. false: Foundation Java API does not enforce the existing security policy settings and you can initialize policy configuration to update the security settings.	boolean	Yes	false

5.1.6 Installing JDK

Download and install the supported version of JDK from Oracle JDK/OpenJDK website. *Oracle JDK and OpenJDK* documentation contains detailed information.

Perform the following tasks for the JDK 17 support:

- On Windows, set the `JAVA_OPTIONS` environment variable to the following value:

```
--add-opens=java.base/java.net=ALL-UNNAMED --add-opens=java.base/
java.lang.ref=ALL-UNNAMED --add-opens=java.naming/
com.sun.jndi.toolkit.url=ALL-UNNAMED --add-exports=java.base/
sun.security.provider=ALL-UNNAMED --add-exports=java.base/
sun.security.pkcs=ALL-UNNAMED --add-exports=java.base/sun.security.x509=ALL-UNNAMED
--add-exports=java.base/sun.security.util=ALL-UNNAMED --add-exports=java.base/
sun.security.tools.keytool=ALL-UNNAMED --add-opens=java.base/java.lang=ALL-UNNAMED
```

- On Linux, set the `JAVA_TOOL_OPTIONS` environment variable to the following value:

```
--add-opens=java.base/java.net=ALL-UNNAMED --add-opens=java.base/
java.lang.ref=ALL-UNNAMED --add-opens=java.naming/
com.sun.jndi.toolkit.url=ALL-UNNAMED --add-exports=java.base/
sun.security.provider=ALL-UNNAMED --add-exports=java.base/
sun.security.pkcs=ALL-UNNAMED --add-exports=java.base/sun.security.x509=ALL-UNNAMED
--add-exports=java.base/sun.security.util=ALL-UNNAMED --add-exports=java.base/
sun.security.tools.keytool=ALL-UNNAMED --add-opens=java.base/java.lang=ALL-UNNAMED
```

5.1.6.1 Using SSL communication

[“Using SSL communication” on page 36](#) contains detailed information.

5.1.7 Additional pre-installation requirements tasks on Windows and Linux

On Windows, disable the IP Helper service from the Windows Services console. This method disables the Teredo Tunneling Pseudo-Interface. Then, restart the machine.

On Linux, the Foundation Java API installation program assumes a video capability of at least 256 colors and at least 800 by 600 screen resolution. You must also make sure that:

- `/usr/dt/bin` and `/usr/openwin/bin` are on the path
- `DISPLAY` is set to `localhost:0.0`
- `XWindows` is installed on the Linux host to run the graphical installation program. The `xterm` program may be installed in various locations depending on the operating system and software packages installed.
- Following RPM packages are available on a 64-bit Linux system:
 - `libXp....i686`
 - `libXi....i686`
 - `libXtst....i686`

- libXt....i686
 - libgcc....i686
 - libXrender....i686
 - xeyes
 - gnome-packagekit
- Random generator on Linux is enabled: A connection request through SSL waits for a random number to be created and a delay is noticed. To avoid the delay, you can enable or disable the random generator on the Linux machine. Perform the following boot safe (inittab) on all Foundation Java API machines:

Start the random generator as root as follows:

```
/sbin/rngd -b -r /dev/urandom -o /dev/random
```

You cannot use a telnet session to install Foundation Java API as the installation program provides a graphical interface. Install from the system console, or use an X server to perform the installation remotely. However, be careful when you install remotely with a *DISPLAY* setting to `localhost:0.0`, as the output is sent to that terminal rather than the one at which you are working.

5.2 Installing Foundation Java API

You can install Foundation Java API on:

- A middle-tier system.

For example, to support Web Development Kit or Documentum CM Server methods.

- An end user computer.

Foundation Java API runs on a Java virtual machine (JVM) on the machine from which you call it.



Note: Using Foundation Java API with an application server may further restrict the supported versions.

5.2.1 Using GUI

1. If you want to use HashiCorp Vault, perform all the tasks as described in “Configuring Documentum Secret Integration Service” on page 49 and make sure that DSIS is running.
2. (Linux) Set environment variables, as described in “Establishing the environment for Foundation Java API” on page 368 and “Environment variables that Foundation Java API uses” on page 371.
3. Run `dfcSetup.exe` (Windows) or `dfcSetup.bin` (Linux).
4. On the welcome page, click **Next**.

5. Accept the license agreement and click **Next**.
6. If you want to configure the HashiCorp Vault secrets, on the **DFC Vault Configuration** page, select the **Enable Vault** check box.
7. If you select the **Enable Vault** check box, provide the following information and click **Next**:
 - a. **DSIS URL:** Provide a value in the following format:

`http://localhost:<port mentioned in application.properties>/dsis`
 - b. **DSIS Token:** Provide the `dsis.dctm.token` token value as described in “[Configuring Documentum Secret Integration Service](#)” on page 49.

! Important

If you enabled the HashiCorp Vault configuration, the installer retrieves all the password information automatically from the HashiCorp Vault server. Make sure that you have stored all the required secrets as described in “[Configuring Documentum CM Server to use HashiCorp Vault](#)” on page 38.

8. Specify the directory into which the installation program should place the Foundation Java API programs and click **Next**.
 The default value for this directory is `C:\Program Files\Documentum`. “[Defining file system locations for Foundation Java API components](#)” on page 369 provides information about how Foundation Java API uses the value you supply.
 The installation program skips this step if it finds a registry entry that contains the required information.
9. Specify whether to install optional components for developers and click **Next**.
 Select the **Developer Documentation** check box to request installation of Javadocs. The installation program places Javadocs into the `help/dfc` subdirectory of the Foundation Java API program root directory. After installation, to view the Javadocs, open `index.html` from this subdirectory.
10. Specify the root directory for OpenText Documentum CM user information and click **Next**.
11. Provide the connection broker information and specify if you want to enable SSL certificates:
 - **Connection Broker Port:** Type the connection broker port number.
 - **Connection Broker Host:** Type the host name. You can use an IP address or a symbolic address (for example, `MyHost.MyCompany.com`).
 - **Use certificates:** Select if you want to enable SSL certificates.
 If **Use certificates** is selected, provide the Foundation Java API trust store information:

- **TrustStore**: The location of the Foundation Java API trust store. For example, \$DOCUMENTUM\secure\dfc.keystore.
- **Password**: The password of the trust store file.

Select **Use Default Java TrustStore** if you want to use the default Java trust store specified by the *javax.net.ssl.trustStore* JVM property.



Note: The default value of the *dfc.session.secure_connect_default* attribute is *try_native_first*. If you want to connect in secure mode, change the attribute value to *secure* or *try_secure_first* in the *dfc.properties* file.

Click **Next**.

12. Browse and specify the directory where the supported version of JDK is installed and click **Next**.

13. Review the summary.

Click **Back** if you want to change previous settings. Otherwise, click **Next**.

14. Use the check box to tell the installation program whether you want to identify the global registry for this Foundation Java API to use.

The installation program skips this step if it finds the required information in the *dfc.properties* file.

It is safe to have the check box cleared if you do not have the necessary information. You can set the information manually in the *dfc.properties* file after the installation program has finished installing Foundation Java API.

If you select the check box, the installation program requests the repository and connection credentials, and provides a check box you can use to disable validation. Use that check box if the repository does not exist or is unavailable. Otherwise, the installation program checks to see if it can use the credentials you provide.

15. Click **Finish**.

16. (Linux) The installation program replaces copies of the shared library that it finds, but other copies may exist on the machine. It is safe to replace all of them with the current version, but if you do not want to do so, you must make sure that the old version does not precede the current version in any path environment variable that the current Foundation Java API might use. If the machine has a Documentum CM Server installation, you must manually replace the DMCL shared library that is in the *bin* directory of Documentum CM Server.

5.2.2 Using command line

The Foundation Java API installation program provides capabilities to support silent installation, that is, invoking the installation program from a command line and giving it a configuration file that allows the installation to proceed without further interaction.

1. Create the configuration file.

To install silently, you must first create a configuration file. To do this, use a command such as the following at a command prompt:

```
dfcSetup.exe -r C:\<Silent installer Property file name>
```

This is the command for Windows. For other operating systems, use the appropriate executable file rather than `dfcSetup.exe`. You can replace `C:\<Silent installer property file name>` with any file you select. Provide the full path, not a path relative to the current directory.

Running this command creates `<Silent installer property file name>` as an installer configuration file. It does this by running the installation program interactively and saving your inputs.



Caution

This process records the information during a real-time installation. If you use this method to create your configuration file, it performs an actual installation during the process.

2. Edit the configuration file created in [step 1](#).

Edit the `<Silent installer property file name>` file and set the following:

```
INSTALLER_UI=silent
```

3. Set the password for global registry as follows:

```
DFC.SECURE.GR_PASSWORD=<global registry password>
```

4. Run the installation program silently.

To run the installation program silently, use a command such as the following at a command prompt:

```
dfcSetup.exe -f C:\<Silent installer property file name>
```

Use the same executable file and configuration file as specified in [step 1](#) and [step 2](#).

It is recommended to always restart after a silent installation because the silent installation program does not say whether a restart is required.

After completing the installation, perform the following steps:

1. Navigate to the folder where the `commons-lang-<packaged-version>.jar.update.bak` file exist. Rename the file from `commons-lang-<packaged-version>.jar.update.bak` to `commons-lang-<packaged-version>.jar`.

2. Configure the shutdown listener entry in `web.xml` of your application (for example, Documentum Administrator and Documentum Webtop) deployed in Tomcat application server to avoid the `LicenseThread` not shutdown properly warning while you shut down Tomcat as follows:

```
<listener>
<listener-class>com.documentum.fc.client.web.ShutdownListener</listener-class>
</listener>
```

5.3 Post-installation task

5.3.1 Licensing OpenText Documentum CM

OpenText Documentum CM uses OpenText Directory Services (OTDS) to apply licenses for all the OpenText Documentum CM components. For more information about procuring the license file and configuring OTDS and license, see “[Licensing OpenText Documentum CM](#)” on page 86.

5.4 Uninstalling Foundation Java API

This section explains how to uninstall Foundation Java API. “[Removing old files](#)” on page 368 lists the situations in which you must uninstall an old version of Foundation Java API before installing the current version. Installation program uses InstallAnywhere.

Regardless of which operating system you use, you cannot uninstall Foundation Java API if any program has locked any portion of it. You must stop any program that uses the Foundation Java API that you want to uninstall. Stopping an application server terminates any web applications running on it, even those that do not use Foundation Java API.

5.4.1 Uninstalling from Windows

1. Change the startup setting to manual for each service that uses Foundation Java API.

This prevents such services (for example, an application server) from restarting themselves after you restart and locking Foundation Java API again before you can uninstall it.

2. Use Control Panel to remove Foundation Java API runtime environment.
3. If prompted to do so, restart the system.
4. Restore any startup settings you changed in [step 1](#).

5.4.2 Uninstalling from Linux

To uninstall Foundation Java API on a Linux system, run the `uninstall.bin` program.

For older versions of Foundation Java API, these programs are available in the `_uninst` subdirectory of the OpenText Documentum CM program root directory. For recent versions, it is available in `_uninst/dfc`.

5.5 IPv6 support

This section discusses the IPv6 support.

5.5.1 OpenText Documentum CM client connection process (dual-stack mode)

During startup, Documentum CM Server projects its host information such as IP address, port number, and so on to its connection broker, which maintains the list of all Documentum CM Servers to which a client can connect. The Foundation Java API client connects to Documentum CM Server using IPv6 or IPv4. The client first tries to communicate using IPv6. If a connection is not established with IPv6, the client tries IPv4. When the client exhausts both IPv6 and IPv4 connectivity options, an error message is displayed. The client caches the connectivity option for the last successful connection to the Documentum CM Server. This improves performance by avoiding repeated attempts to communicate with connection options in which connectivity is not established.



Note: The Foundation Java API cache is refreshed at regular (configurable) time intervals.

5.5.2 Configuring Foundation Java API client

The client retrieves the information about the Documentum CM Server by using the full qualified server name (`docbase_name.server_config_name@host_name`) from the connection brokers listed in the `dfc.properties` file. In the `dfc.properties` file, you can specify an IP address or host name on which the connection broker runs. If you specify an IPv6 address, enclose the IP address in square brackets as per the IPv6 convention.

```
dfc.host.name[0] = [2001:0db8:1234:0000:0000:0000:0000:0000]
```

The following applies when a dual-stack client host connects to a connection broker running on a dual-stack machine:

- When you specify an IP address in the `dfc.properties` file, the client uses that IP address and connects without any further processing. For example, if you specify an IPv4 address, the client uses IPv4 for communication.

- When you specify a host name in the `dfc.properties` file, the client resolves all available IP addresses for that host name before determining the connection protocol. When the connection broker runs on a dual-stack machine, the client resolves both IPv4 and IPv6 addresses. The client keeps track of the IPv4 address and selects the best available IPv6 address for the host from Unicast Global, Site Local, and Link Local, in the order specified.

To configure Foundation Java API installed on a dual-stack machine for native IPv4 operation, perform the following tasks:

- Specify an IPv4 address in the `dfc.properties` file.
- Disable the dual-stack operation for Java Virtual Machine.

5.5.3 Configuring JVM for IPv4 and IPv6

A custom property setting in JVM used by the operating system determines the communications protocol used by the operating system. By default, the value of the `java.net.preferIPv4Stack` property is set to `false` to support dual-stack communications.

To configure a host:

- For native IPv4: Set the value of `java.net.preferIPv4Stack` to `true`.
- For native IPv6: Set the value of `java.net.preferIPv6Addresses` to `true` and the value of `java.net.preferIPv4Stack` to `false`.

5.6 Troubleshooting

The installation program maintains an error log file named `install.log` in the current working directory. If it cannot write into the working directory, it writes in the home directory of the user who initiated the installation.

Chapter 6

Foundation SOAP API

This documentation describes how to deploy the Foundation SOAP API web services to a supported web application server, as well as information about configuration of the Foundation SOAP API server environment.

6.1 Introduction

Foundation SOAP API provides a set of technologies that enables service-oriented programmatic access to the Documentum CM Server platform and related products. The intended audience is system administrators or programmers who must deploy Foundation SOAP API.

6.2 About Foundation SOAP API deployment

6.2.1 Local and remote Foundation SOAP API applications

One of the first considerations in planning an application using Foundation SOAP API is whether to use the local Foundation SOAP API Java services, or whether to use the remote Foundation SOAP API web services.

The local Foundation SOAP API services are packaged in Java libraries available in the Foundation SOAP API SDK. A local service consumer runs in the same Java Virtual Machine as the services that it invokes, which connect to repositories using an underlying Foundation Java API client. For more information on developing local Foundation SOAP API services, see *OpenText Documentum Content Management - Foundation SOAP API Development Guide (EDCPKSVC250400-PGD)*.

The remote Foundation SOAP API services are SOAP-based web services hosted in a J2EE servlet container. In a remote Foundation SOAP API application, the Foundation SOAP API web services are invoked by a SOAP client.

If any of the Foundation SOAP API client must be used with current version of Foundation SOAP API server, then you must repackage the SDK applications with the following Foundation SOAP API SDK client versions: current version and later till 16.4 P14 and later, or 7.3 P25 and later.

6.2.2 Supported environments

Before you begin deployment, review the product *Release Notes* to make sure that the environment to which you are deploying is supported. The product *Release Notes* contains the information about the supported web application server environments. Make sure that you are using a supported web application server version on a supported operating system, and that any required updates from the web application server vendor have been applied. Make sure that your web application server is running on a supported version of the JVM.

You must install a Foundation SOAP API server on a machine that is different from the one on which Documentum CM Server is installed.

Clustered deployment is supported on all supported web application servers. In clustered deployments, Foundation SOAP API must be deployed to each node in the cluster. In addition, the class loader settings on each node must be configured according to the instructions for the specific web application server.



Note: Rate limiting is not set in Foundation SOAP API. You must configure the rate limiting with your infrastructure components such as load balancer or proxy server to prevent any application-level denial-of-service (DoS) attacks.

6.2.3 Web services archive files

To deploy Foundation SOAP API web services, you need one of the following two files:

- `dfs.ear`
- `dfs.war`

You must use the `dfs.war` file for deployment on Tomcat. For all other supported application servers, use `dfs.war` or `dfs.ear`.

6.2.4 Clustered deployment for load balancing

Foundation SOAP API web services can be deployed in web application server clusters. Deployment of Foundation SOAP API to clusters is supported only for purposes of load balancing, and requires sticky sessions.

Other than the requirement to configure sticky sessions, there is no procedure specific to Foundation SOAP API for clustered deployment, so you can follow the procedure recommended by the web application server vendor for deploying web applications in a clustered configuration.

6.2.4.1 Failover not supported

While Foundation SOAP API supports clustered deployments for purposes of load balancing, it does not support failover (high availability) out of the box. There may be workarounds that enables failover using third-party products, which are likely to impose limitations on functions available to Foundation SOAP API client applications (such as registered service contexts, cached queries, and client-orchestrated UCF). To pursue these options you should contact and arrange for a consultation with OpenText Global Technical Services.

6.3 Configuration

This section covers configuration settings that are applicable to Foundation SOAP API deployments on any web application server.

6.3.1 Configuring JVM settings

To provide adequate heap space, OpenText recommends the following JVM settings:

- -Xms512m
- -Xmx512m

6.3.2 Setting for Message Transmission Optimization Mechanism content transfer mode

The Foundation SOAP API .NET client is based on Windows Communication Framework (WCF), which provides three modes for Message Transmission Optimization Mechanism (MTOM) content transfer: buffer, streaming, and chunk. For streaming and chunk modes, WCF requires that the corresponding service operation (such as get or create) take only one argument (the input stream). This conflicts with the design of Foundation SOAP API, such that Foundation SOAP API can only use the MTOM buffer mode with a .NET client. This results in unusually high memory requirements, especially when trying to transfer large content payloads when Accelerated Content Services is unavailable or switched off on the server, because the entire content must be buffered in memory before transfer. Normally a .NET client uses Accelerated Content Services if it is available for content download operations, so under typical conditions the memory limitation is not encountered. However, if Accelerated Content Services content is unavailable, or if the client attempts to upload a very large content stream to the server using MTOM content transfer mode, it may result in exceeding the server's capacity to buffer the content.

Use the following configurations to avoid the memory leak issue:

- Enable Accelerated Content Services/Branch Office Caching Services for content download operations. To make sure that the `urlContent` type is returned by Foundation SOAP API, use the `urlReturnPolicy` setting. The client can use the `urlContent` returned by Foundation SOAP API to request content transfer from the Accelerated Content Services server.

- Use UCF as the content transfer mode for content download operations. UCF orchestrates content transfer in both directions between the client and the Accelerated Content Services server.
- Optionally, make sure that both the Foundation SOAP API .NET client and JVM that runs the Foundation SOAP API server have enough memory to buffer the content. However, be aware that in this case the application is limited to transfer of content in the range of hundreds of megabytes for a 32-bit JVM, because on most modern 32-bit Windows systems the maximum heap size ranges from 1.4G to 1.6G. Although this specific limitation does not apply to a 64-bit version of Windows, the issue still exists if you do not ensure that there is sufficient heap space to buffer very large objects in memory.

6.3.3 Installing JDK

Download and install the supported version of Oracle JDK/OpenJDK version from Oracle JDK/OpenJDK website. The *Oracle JDK and OpenJDK* documentation contains detailed information.

6.3.3.1 Using SSL communication

[“Using SSL communication” on page 36](#) in [“Documentum CM Server” on page 9](#) contains detailed information.

6.3.4 Configuring Foundation Java API

The `dfc.properties` file provides property settings for the Foundation Java API runtime that Foundation SOAP API depends on. This file is located in `APP-INF/classes` if you are deploying the EAR file, or in `WEB-INF/classes` if you are deploying the WAR file.

If you prefer, you can use a `#include` statement to point to a properties file outside of the web application on the local file system. This can make access to some settings more convenient and allows you to modularize your configuration settings:

```
#include C:\Documentum\config\dfc.properties
```

6.3.4.1 Connection broker and global registry properties

The `dfc.properties` file includes the critical settings that are required for Foundation SOAP API to reach a connection broker and connect to a Documentum CM Server.

Table 6-1: `dfc.properties` connect and global registry properties

Property	Value
<code>dfc.docbroker.host[0]</code>	The fully qualified host name for the connection broker. You can add backup hosts by adding new properties and increment the index number within brackets.

Property	Value
dfc.docbroker.port	If you wish to use a port for the connection broker other than the default of 1489, add a port key.
dfc.globalregistry.repository	The global registry repository name.
dfc.globalregistry.username	The user name of the global registry user. The global registry user, who has the default user name dm_bof_registry, must have READ access to objects in the /System/Modules and /System/NetworkLocations only.
dfc.globalregistry.password	An encrypted password value for the global registry user.

You can either copy the user name and encrypted password for the global registry user from the `dfc.properties` file on the global registry Documentum CM Server host, or you can select another global registry user and encrypt the password using the following command format:

```
java -cp dfc.jar com.documentum.fc.tools.RegistryPasswordUtils <password to be encrypted>
```

6.3.4.2 Properties required by the Schema service

The following two properties are required for SchemaService performance:

```
dfc.cache.ddinfo.size=10000
dfc.cache.type.currency_check_interval=86400
```

- The `dfc.cache.ddinfo.size` property can take values ranging from 1 to 10000.
- The `dfc.cache.type.currency_check_interval` property can take values ranging from 0 to 86400.

6.3.4.3 Reusing privileged Foundation Java API client instance

To reuse the privileged Foundation Java API client instance, retain the IP address or host name of the web application server machine and specify the following in the `dfc.properties` file:

```
dfc.security.keystore.file=
#dfc.keystore location
```

6.3.4.4 Trusted login on the same host as Documentum CM Server

Trusted login is disabled by default. If required, you can set the `dfc.session.allow_trusted_login` property.

6.3.5 Configuring `dfs-runtime.properties`

Foundation SOAP API configuration settings are set in the `dfs-runtime.properties` file. This file is located in the `emc-dfs-rt.jar` file that resides in the `dfs.ear` file. If you edit this file, repackage it in the `dfs.ear` file with the same file and packaging structure. To avoid the inconvenience of extracting the JAR file to modify these properties, there are two options for using property files:

- Use an external configuration file that can be located in `APP-INF/classes` if you are deploying the EAR file, or in `WEB-INF/classes` if you are deploying the WAR file. You must rename this file to `local-dfs-runtime.properties`.
- If you are using the productivity layer runtime (in a web application using Foundation SOAP API in local mode), you can use an external properties file on the local file system specified by the `-Ddfs.runtime.properties.file` JVM parameter.

For example:

```
-Ddfs.runtime.properties.file=
C:\\DFS\\config\\dfs-runtime.properties
```

The following list describes the precedence that these files take depending on their location:

- a `local-dfs-runtime.properties` file in the local classpath
- a runtime properties file specified with `-Ddfs.runtime.properties.file` in productivity layer deployments only
- a `dfs-runtime.properties` file packaged with the `emc-dfs-rt.jar` file

The Foundation SOAP API application must be restarted after any changes to the configuration. As a best practice, use the provided configuration file that is deployed in the `emc-dfs-rt.jar` file for your base settings and use an external file to override settings that you specifically wish to change.

6.3.5.1 Required settings

The `tracing.enabled` and `resource.bundle` Foundation SOAP API property settings are mandatory. An exception is thrown if either of the setting is not provided in the configuration file. The settings in the properties file as delivered are as follows:

```
# mandatory runtime properties
tracing.enabled = false
resource.bundle = dfs-messages
resource.bundle.1 = dfs-services-messages
resource.bundle.2 = dfs-bpm-services-messages
```

6.3.5.2 Configuration settings in `dfs-runtime.properties`

By modifying the corresponding property in the `dfs-runtime.properties` file, you can perform the following tasks:

- Enable or disable AspectJ tracing for Foundation SOAP API.
- Specify the maximum number of elements that `QueryService` can return.
- Specify the cache size in elements or rows for the query.
- Specify the amount of cache to retain if a query result is not stored in the cache.
- Specify the period for the cache housekeeper to run for the Query service.
- Specify the period for the query cache manager to clean the expired cache.
- Specify the maximum number of elements in the query cache manager.
- Specify the period for the cache housekeeper to run for the Agent service.
- Set the `ContextRegistry` service expiration variables.
- Specify how often expired service contexts are being removed from the cache to free up resources.
- Specify the expiration in minutes of Foundation SOAP API temporary content files since the last modification date.
- Specify the initial interval of Foundation SOAP API scheduled task for temporary content files cleanup.
- Specify the folder in which Foundation SOAP API temporary content files are stored; if not specified, the temporary folder is determined.
- Specify whether to throw an exception if any downloaded JAR such as `ucf-installer.jar` is not signed using a trusted certificate.

More detailed property descriptions are available as comments in the `dfs-runtime.properties` file. For more information about these properties, see the `dfs-runtime.properties` file packaged in the `emc-dfs-rt.jar` file.

6.3.6 Configuring Single sign-on properties

The `dfs-sso-config.properties` file specifies values that the Foundation SOAP API services use to process HTTP requests and pass expected values to the Single sign-on (SSO) plug-in using Foundation Java API. This file is located in `WEB-INF/classes` for the WAR package and located in `APP-INF/classes` for the EAR package.

This file is required in the Foundation SOAP API web service application. The client productivity layer runtime also uses this file to determine the header and cookie names to use for the user name and password in the HTTP request to the Foundation SOAP API service. If the file is not found, the Foundation SOAP API client runtime uses default values for the header and cookie name. Therefore, if you want to use special (non-default) names for the header and cookie in a Foundation SOAP API application that uses the client productivity layer, you should copy `dfs-sso-config.properties` to a folder on your Foundation SOAP API client application classpath.

Table 6-2: `dfs-sso-config.properties`

Property name	Description
<code>sso.type</code>	The SSO server type. Supported values are <code>dm_otds_token</code> for passing OTDS token and <code>dm_otds_password</code> for passing OTDS password.
<code>user.header.name</code> , <code>user.header.name.<integer value></code>	A list of possible names of HTTP headers in the HTTP request that can potentially contain user names. If more than one header from the list is found in the HTTP request, Foundation SOAP API uses the first header from the list that it finds.
<code>password.cookie.name</code>	The name of the cookie in the HTTP request that contains the SSO ticket.
<code>sso.argument</code>	The SSO server address.

Some sample settings for OTDS are as follows:

```
# type of single sign on server sso.type = dm_otds_token
# list of possible names of headers that contain user name.
# In case there will be more than one header with
# user name the first found header will be used.user.header.name = remoteUserName
# name of the cookie containing the sso ticket      password.cookie.name = ssoTicket
# sso argument to specify SSO proxy server
# so that SSO plug-in in Documentum CM Server side will know the SSO server address
# sso.argument =
```

6.3.7 Configuring secure socket layer for Foundation SOAP API

This section discusses how to configure secure socket layer (SSL) for Foundation SOAP API. Settings on various web application servers may differ from each other. For detailed operations to enable SSL for a specific web application server, see the corresponding documentation.

To configure SSL for Foundation SOAP API:

1. Get a certificate. You can purchase certificates issued by certificate authority (CA). Alternatively, you can create self-signed certificates.
2. Enable SSL on the web application server that hosts Foundation SOAP API. For more details, see the documentation of your web application server.
3. On the Foundation SOAP API client, test whether SSL is correctly enabled on the web application server by entering the following URL in a browser:

```
https://<server>:<port>/services/core/ObjectService?WSDL
```

If SSL is correctly enabled, the browser displays the contents of the WSDL file.

4. Export the certificate to a local file by using the certificate export wizard for later use.
5. Import the certificate on the client.

a. Foundation SOAP API client:

- If you use the Foundation SOAP API .NET productivity layer, import the certificate to the Windows certificate manager.
- If you use the Foundation SOAP API Java productivity layer, import the certificate to the keystore of the local JRE that runs the Foundation SOAP API client.

Sample command:

```
%JRE_HOME%\bin\keytool
-import -alias test -file dfs.cert -keystore
"%JRE_HOME%\lib\security\cacerts"
-keypass <password>
```



Note: Keytool is a key and certificate management utility. The sample command is for reference purpose only. For more information about the keytool utility, see *Oracle* documentation.

b. UCF client:

- If you use the UCF Java client and use the private JRE to launch the UCF Java client, import the certificate to the keystore of the private JRE.
- No additional configuration is needed to launch the UCF .NET client.

6. If you use the Foundation SOAP API .NET productivity layer, modify the application configuration file such as `app.config` or `web.config`. Update the security mode attribute to `Transport` as follows:

```
<system.serviceModel>
  <bindings>
    <basicHttpBinding>
      <binding name="DfsAgentService" ...>
        <security mode="Transport">
          <transport clientCredentialType="None"
                    proxyCredentialType="None" realm="" />
          <message clientCredentialType="UserName"
                    algorithmSuite="Default" />
        </security>
      </binding>
      <binding name="DfsContextRegistryService" ...>
        <security mode="Transport">
          <transport clientCredentialType="None"
                    proxyCredentialType="None" realm="" />
          <message clientCredentialType="UserName"
                    algorithmSuite="Default" />
        </security>
      </binding>
      <binding name="DfsDefaultService" ...>
        <security mode="Transport">
          <transport clientCredentialType="None"
                    proxyCredentialType="None" realm="" />
          <message clientCredentialType="UserName"
                    algorithmSuite="Default" />
        </security>
      </binding>
    </basicHttpBinding>
  </bindings>
</system.serviceModel>
```



Note: If the application configuration does not exist, you have to create one. For more samples of the configuration file, see the Foundation SOAP API Software Development Kit.

6.3.8 Configuring Transport Layer Security 1.2

Foundation SOAP API uses Transport Layer Security (TLS) 1.0 at runtime.

To enable TLS 1.2:

1. Enable strong cryptography in .NET Framework by adding the following registry keys:
 - [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework\v4.0.30319]
“SchUseStrongCrypto”=dword:00000001
 - [HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\.NETFramework\v4.0.30319]
“SchUseStrongCrypto”=dword:00000001
2. Secure Channel (Schannel) TLS 1.2 by adding the following registry keys:
 - [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2]

- [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2\Client]

“DisabledByDefault”=dword:00000000

“Enabled”=dword:00000001
- [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2\Server]

“DisabledByDefault”=dword:00000000

“Enabled”=dword:00000001

6.4 Deploying on web application servers

The following sections provide information on deployment of Foundation SOAP API, or of any Enterprise Content Services services, on supported web application servers.



Note: If the web application server is using JDK 21, no additional Java options are required.

6.4.1 Apache Tomcat

Make sure that the Tomcat JVM settings meet the recommendations specified in “[Configuring JVM settings](#)” on page 385. Apache Tomcat does not support EAR archives. Therefore, to deploy on Tomcat, download `dfs.war` for Foundation SOAP API (or the provided WAR file for deploying another set of Enterprise Content Services services).

To perform a simple WAR file deployment, copy the WAR file to the `<Tomcat_Home>/webapps` folder. Tomcat extracts the WAR file to the `<Tomcat_Home>/webapps/<application_name>` folder, where `<application_name>` is the name of the WAR file without the file extension. The web application name is included in the service address as follows:

```
http://localhost:8080/emc-dfs/services/<module name>/<service name>
```

In Foundation SOAP API productivity-layer, clients modify the `contextRoot` settings in the `dfs-client.xml` file, as required.

If you do not want to specify the web application name in the address, deploy Foundation SOAP API as the Tomcat root application. To deploy Foundation SOAP API as the Tomcat root application, rename the WAR file to `ROOT.war` and delete or rename the `webapp/ROOT` application folder. The services are available at the location as it is shown throughout the Foundation SOAP API documentation:

```
http://localhost:8080/services/<module name>/<service name>
```

For the JDK 17 support, add the following parameters in the `catalina.bat` file:

```
set "JAVA_OPTS=-Dprogram.name=%PROGNAME% %JAVA_OPTS%"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
```

```
--add-opens=java.base/java.net=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.lang.ref=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.naming/com.sun.jndi.toolkit.url=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.provider=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.pkcs=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.x509=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.util=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.xml.crypto/com.sun.org.apache.xml.internal.security
=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.lang=ALL-UNNAMED"
```

For more information, refer to the Apache Tomcat website.



Note: If you want to access Foundation SOAP API URL in an IPv6 environment, perform the steps described in “[Accessing Foundation SOAP API URL in an IPv6 environment](#)” on page 399.

6.4.2 WildFly

1. Copy the Foundation SOAP API EAR file to your `<WildFly-version>\standalone\deployments\` folder.
2. **Optional** If you want to access Foundation SOAP API URL in an IPv6 environment, perform the steps described in “[Accessing Foundation SOAP API URL in an IPv6 environment](#)” on page 399.
3. For the JDK 17 support, add the following parameters in the `standalone.bat` file:

```
set "JAVA_OPTS=-Dprogram.name=%PROGNAME% %JAVA_OPTS%"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.net=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.lang.ref=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.naming/com.sun.jndi.toolkit.url=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.provider=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.pkcs=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
```

```
--add-exports=java.base/sun.security.x509=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.util=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.xml.crypto/com.sun.org.apache.xml.internal.security
=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.lang=ALL-UNNAMED"
```

4. Restart the WildFly server using the following command depending on your operating system and communication protocol:

- On IPv4:

Windows:

```
standalone.bat -b 0.0.0.0
```

Linux:

```
standalone.sh -b 0.0.0.0
```

- On IPv6:

Windows:

```
standalone.bat -b [<IPv6 address>]
```

Linux:

```
standalone.sh -b <IPv6 address>
```

6.4.3 Red Hat JBoss Enterprise Application Platform (JBoss EAP)

1. Download the JBoss EAP 8 Update 4 patch file named `jboss-eap-8.0.4-runtime-maven-repository.zip` from the Red Hat website.
2. Extract the `jboss-eap-8.0.4-runtime-maven-repository.zip` file to a temporary location.
3. Apply the JBoss EAP 8 Update 4 patch.

Windows

For example, from `<JBoss EAP_Home>\bin`, run the following command as an administrator:

```
jboss-eap-installation-manager.bat update perform ^
--dir C:\jboss-eap-8.0 ^
--repositories mrrc::file:C:\jboss-eap-8.0.4.GA-runtime-maven-repository
\maven-repository ^
--offline
```

Linux

For example, from `<JBoss EAP_Home>/bin`, run the following command as an administrator:

```
./jboss-eap-installation-manager.bat update perform \
>   --dir ../../jboss-eap-8.0 \
>   --repositories mrrc::file:C:\jboss-eap-8.0.4.GA-runtime-maven-repository
\maven-repository \
>   --offline
```

4. Copy the Foundation SOAP API EAR file to your <*JBoss EAP-version*> \standalone\deployments folder.
5. **Optional** If you want to access Foundation SOAP API URL in an IPv6 environment, perform the steps described in “[Accessing Foundation SOAP API URL in an IPv6 environment](#)” on page 399.
6. Restart the JBoss EAP server using the following command depending on your operating system and communication protocol:
 - On IPv4:
Windows:

```
standalone.bat -b 0.0.0.0
```

Linux:

```
standalone.sh -b 0.0.0.0
```
 - On IPv6:
Windows:

```
standalone.bat -b [<IPv6 address>]
```

Linux:

```
standalone.sh -b <IPv6 address>
```



Notes

- For the JDK 17 support, add the following parameters in the standalone.bat file:

```
set "JAVA_OPTS=-Dprogram.name=%PROGNAME% %JAVA_OPTS%"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-opens=java.base/java.net=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-opens=java.base/java.lang.ref=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-opens=java.naming/com.sun.jndi.toolkit.url=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-exports=java.base/sun.security.provider=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-exports=java.base/sun.security.pkcs=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-exports=java.base/sun.security.x509=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"  
--add-exports=java.base/sun.security.util=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"
```

```
--add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.xml.crypto/com.sun.org.apache.xml.internal.security
=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.lang=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.util=ALL-UNNAMED"
```

- To avoid any JBoss EAP log issues, add the following line to the existing contents of the `jboss-deployment-structure.xml` file:

```
<exclude-subsystems><subsystem name="logging"/></exclude-subsystems>
```

6.4.4 IBM WebSphere Liberty

1. Configure Java.

To configure Java, specify the Java to be used in environment variables in the Liberty application server machine. After the `JAVA_HOME` and Path are set, the `<yourserver> console.log` file displays the Java that has been used to start the server.

2. Create a Liberty web application server manually.

The Liberty profile supports multiple servers using the same installation. Each server is defined in its own folder, where the folder name is the server name.

- a. Open a command line, and change the folder to `wlp/bin/`. For example, `C:\IBM\WebSphere\Liberty\bin\`.

- b. Run the `server create <server name>` command to create a server. For example:

```
C:\IBM\WebSphere\Liberty\bin>server create serverdfs
```

3. For the JDK 17 support, add the following parameters in the `server.bat` file:

```
set "JAVA_OPTS=-Dprogram.name=%PROGNAME% %JAVA_OPTS%"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.net=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.lang.ref=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.naming/com.sun.jndi.toolkit.url=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.provider=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.pkcs=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.x509=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.util=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%"
```

```
--add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.xml.crypto/com.sun.org.apache.xml.internal.security
=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.lang=ALL-UNNAMED"
```

4. Update the `featureManager` property in the `server.xml` file as follows:

 **Note:** Additionally, if IBM Liberty is SSL enabled, add `<feature>ssl-1.0</feature>` to the `featureManager` property.

```
<featureManager>
<feature>servlet-6.0</feature>
<feature>jsp-3.1</feature>
</featureManager>
```

5. Copy the `dfs.war` file into the folder that you created in [step 2.b](#) (for example, `/usr/servers/serverdfs/dropins`).

If you want to start, stop, or check the status of the Liberty web application server, use the following example commands:

- To start:

```
C:\IBM\WebSphere\Liberty\bin>server start serverdfs
```

- To stop:

```
C:\IBM\WebSphere\Liberty\bin>server stop serverdfs
```

- To check the status:

```
C:\IBM\WebSphere\Liberty\bin>server status serverdfs
```

6.4.5 Configuring Foundation SOAP API to connect to HashiCorp Vault

1. Perform all the tasks as described in [“Configuring Documentum Secret Integration Service” on page 49](#) and make sure that DSIS is running.
2. While deploying Foundation SOAP API on the supported application servers, update the `dfc.properties` file with the following entries:

```
dfc.dsds.enabled=true
dfc.dsds.daemon.url=http://localhost:8200/dsis
dfc.dsds.daemon.token=<token generated for the DSIS daemon agent>
```

For more information about setting the preceding entries to the `dfc.properties` file, see *OpenText Documentum Content Management - Foundation Java API Development Guide (EDCPKCL250400-DGD)*.

3. Redeploy the WAR file.
4. Restart the application server.

6.4.6 Accessing Foundation SOAP API URL in an IPv6 environment

1. Search for `-Djava.net.preferIPv4Stack=true` in all the configuration files in the web application servers' home folder.

2. Change the value as follows:

From

```
-Djava.net.preferIPv4Stack=true
```

To

```
-Djava.net.preferIPv4Stack=false -Djava.net.preferIPv6Addresses=true
```

3. Search for `127.0.0.1` in all the configuration files in the web application servers' home folder.

4. Change the value from `127.0.0.1` to `[::1]`.

5. To access the Foundation SOAP API URL in an IPv6 environment, you must provide the IPv6 address within the square brackets in the URL.

6.4.7 Configuring UCF client

Enable the `client.acs.proxy.enable` option in the `ucf.installer.config.xml` file and set the appropriate proxy server details in the UCF client machine to avoid the UCF client failing to connect to the Accelerated Content Services or Branch Office Caching Services server which are behind proxy server.

6.4.8 Validating Foundation SOAP API deployment

To test whether your Foundation SOAP API deployment is successful, initially, open a service WSDL in a browser. For example, to test a Foundation SOAP API instance deployed on localhost at port 8080, perform the following task:

Open the QueryService WSDL in your browser using the following URL:

```
http://localhost:8080/services/core/QueryService?wsdl
```

If you have deployed the WAR file on Tomcat and you have not deployed Foundation SOAP API as the Tomcat root web application, then the web application name is included in the address. For example:

```
http://localhost:8080/emc-dfs/services/core/QueryService?wsdl
```

You can further test the deployment by running a simple sample consumer, such as the `TQueryServiceTest.java` or `QueryServiceTest.cs` consumers provided in the Foundation SOAP API Software Development Kit. For information about running these consumers, see *OpenText Documentum Content Management - Foundation SOAP API Development Guide (EDCPKSVC250400-PGD)*.

6.5 Post-deployment task

6.5.1 Licensing OpenText Documentum CM

OpenText Documentum CM uses OpenText Directory Services (OTDS) to apply licenses for all the OpenText Documentum CM components. For more information about procuring the license file and configuring OTDS and license, see “[Licensing OpenText Documentum CM](#)” on page 86.

6.6 Enabling OTDS authentication in Foundation SOAP API services

Foundation SOAP API client applications can authenticate OTDS users by providing user and password information in the Foundation SOAP API request as follows:

- Password-based authentication: Provide the user name and password in the following format:
`<user_login_name>,dm_otds_password=<user_password>`
- oAuth2 token-based authentication: Provide the user name and password in the following format:
`<user_login_name>,dm_otds_oauth=<oAuth2_token>`



Note: From the 22.4 release, Foundation SOAP API supports OTDS SSO. For more information, see *OpenText Documentum Content Management - Foundation SOAP API Development Guide (EDCPKSVC250400-PGD)*.

Chapter 7

Foundation REST API

This documentation describes how to deploy OpenText Documentum Content Management (CM) Foundation REST API to a supported web application server.

7.1 Introduction

Foundation REST API is a set of RESTful web service interfaces that interact with OpenText Documentum CM.

7.2 Configuring Foundation Java API

The `dfc.properties` file provides property settings for the Foundation Java API runtime. Foundation REST API depends on these property settings. The `dfc.properties` file is located in `WEB-INF/classes` when you deploy the WAR file.

You can use an `#include` statement to point to a properties file outside of the web application on the local file system. Using `#include` statements in this manner makes access to some settings more convenient and allows you to modularize your configuration settings. Here is an example that shows you how to use an `#include` statement with a file path:

```
#include C:\Documentum\config\dfc.properties
```

7.2.1 Connection broker and global registry properties

The `dfc.properties` file includes critical settings that are required for RESTful Services to reach a connection broker and connect to Documentum CM Server.

Property	Value
<code>dfc.docbroker.host[0]</code>	The fully qualified host name for the connection broker. You can add backup hosts by adding new properties and increment the index number within the brackets.
<code>dfc.docbroker.port[0]</code>	When you use a port for the connection broker other than the default of 1489, add a port key.
<code>dfc.globalregistry.repository</code>	The global registry repository name.
<code>dfc.globalregistry.username</code>	The user name of the global registry user. The global registry user, who has the default user name <code>dm_bof_registry</code> , must have read access only to the objects that are in the <code>/System/Modules</code> folder and the <code>/System/NetworkLocations</code> folder.

Property	Value
dfc.globalregistry.password	An encrypted password value for the global registry user.

To reach a connection broker, you may use one of the following options:

- Copy the user name and encrypted password for the global registry user from the `dfc.properties` file on the global registry Documentum CM Server host.
- Select another global registry user and encrypt the password using the following command:

```
java -cp dfc.jar com.documentum.fc.tools.RegistryPasswordUtils <password  
to be Encrypted>
```

7.3 Deploying on web application servers

The following sections provide information on deployment of Foundation REST API on supported web application servers.

7.3.1 Apache Tomcat

Follow these steps to perform a simple WAR file deployment on an Apache Tomcat server:

1. Stop the Apache Tomcat server.
2. Copy the WAR file to the `<Tomcat_installdir>/webapps` folder.
3. For the JDK 17 support, add the following parameters in the `catalina.bat` file:

```
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%  
--add-opens=java.base/java.lang=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%  
--add-opens=java.base/java.io=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%  
--add-opens=java.base/java.util=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%  
--add-opens=java.base/java.util.concurrent=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%  
--add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%  
--add-opens=java.base/java.net=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%  
--add-opens=java.base/java.lang.ref=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%  
--add-opens=java.naming/com.sun.jndi.toolkit.url=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%  
--add-opens=java.naming/com.sun.jndi.ldap=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%  
--add-exports=java.base/sun.security.provider=ALL-UNNAMED"
```

```

set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.pkcs=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.x509=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.util=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED"

```

4. Start the Apache Tomcat server.

7.3.2 WildFly

1. Copy the WAR file to your <WildFly_installdir>\standalone\deployments folder.
2. **Optional** If you want to access Foundation REST API URL in an IPv6 environment, perform the steps described in “[Accessing Foundation REST API URL in an IPv6 environment](#)” on page 407.
3. For the JDK 17 support, add the following parameters in the standalone.bat file:

```

set "JAVA_OPTS=-Dprogram.name=%PROGNAME% %JAVA_OPTS%"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.net=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.lang.ref=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.naming/com.sun.jndi.toolkit.url=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.provider=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.pkcs=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.x509=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.util=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.xml.crypto/com.sun.org.apache.xml.internal.security
=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.lang=ALL-UNNAMED"

```

4. Restart the WildFly server using the following command depending on your operating system and communication protocol:
 - a. On IPv4:
 - Windows:

```
standalone.bat -b 0.0.0.0
```

Linux:

```
standalone.sh -b 0.0.0.0
```

- b. On IPv6:

Windows:

```
standalone.bat -b [<IPv6 address>]
```

Linux:

```
standalone.sh -b <IPv6 address>
```

7.3.3 Red Hat JBoss Enterprise Application Platform (JBoss EAP)

Deploying Foundation REST API on the JBoss EAP server is similar to any other web application deployment on the JBoss server.

To download and update JBoss EAP 8 Update 4 patch:

1. Download the JBoss EAP 8 Update 4 patch file named `jboss-eap-8.0.4-runtime-maven-repository.zip` from the Red Hat website.
2. Extract the `jboss-eap-8.0.4-runtime-maven-repository.zip` file to a temporary location.
3. Apply the JBoss EAP 8 Update 4 patch.

For example, from `<JBoss_EAP_Home>\bin`, run the following command as an administrator:

```
jboss-eap-installation-manager.bat update perform ^
    -dir C:\jboss-eap-8.0 ^
    --repositories mrrc::file:C:\jboss-eap-8.0.4.GA-runtime-maven-repository\maven-
repository ^
    --offline
```

For more information on how to deploy web application on the JBoss server, see the Red Hat website.



Notes

- For the JDK 17 support, add the following parameters in the `standalone.bat` file:

```
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.lang=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.io=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.util=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.util.concurrent=ALL-UNNAMED"
```

```

set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.net=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.lang.ref=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.naming/com.sun.jndi.toolkit.url=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.naming/com.sun.jndi.ldap=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.provider=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.pkcs=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.x509=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.util=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED"

```

- To avoid any JBoss EAP log issues, add the following line to the existing contents of the `jboss-deployment-structure.xml` file:

```
<exclude-subsystems><subsystem name="logging"/></exclude-subsystems>
```

7.3.4 IBM WebSphere Liberty

- Configure Java.

To configure Java, specify the Java to be used in environment variables in the Liberty application server machine. After the `JAVA_HOME` and Path are set, the `<yourserver> console.log` file displays the Java that has been used to start the server.

- Create a Liberty web application server manually.

The Liberty profile supports multiple servers using the same installation. Each server is defined in its own folder, where the folder name is the server name.

- Open a command line, and change the folder to `<Liberty_installdir>\bin`. For example, `C:\IBM\WebSphere\Liberty\bin`.
- Run the `server create <server name>` command to create a server. For example:

```
C:\IBM\WebSphere\Liberty\bin>server create serverrest
```

- For the JDK 17 support, add the following parameters in the `server.bat` on Windows or `server.sh` on Linux:

```

set "JAVA_OPTS=-Dprogram.name=%PROGNAME% %JAVA_OPTS%"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.net=ALL-UNNAMED"

```

```
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%  
--add-opens=java.base/java.lang.ref=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%  
--add-opens=java.naming/com.sun.jndi.toolkit.url=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%  
--add-exports=java.base/sun.security.provider=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%  
--add-exports=java.base/sun.security.pkcs=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%  
--add-exports=java.base/sun.security.x509=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%  
--add-exports=java.base/sun.security.util=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%  
--add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%  
--add-opens=java.xml.crypto/com.sun.org.apache.xml.internal.security  
=ALL-UNNAMED"  
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%  
--add-opens=java.base/java.lang=ALL-UNNAMED"
```

4. Update the `featureManager` property in the `server.xml` file as follows:

```
<featureManager>  
<feature>servlet-6.0</feature>  
<feature>jsp-3.1</feature>  
</featureManager>
```

5. Copy the `dctm-rest.war` file into the folder that you created in [step 2.b](#). For example, `C:\IBM\WebSphere\Liberty\serverrest\dropins`.
6. If you want to start, stop, or check the status of the Liberty web application server, use the following example commands:

- To start:

```
C:\IBM\WebSphere\Liberty\bin>server start serverrest
```

- To stop:

```
C:\IBM\WebSphere\Liberty\bin>server stop serverrest
```

- To check the status:

```
C:\IBM\WebSphere\Liberty\bin>server status serverrest
```

7.4 Post-deployment task

7.4.1 Licensing OpenText Documentum CM

OpenText Documentum CM uses OpenText Directory Services (OTDS) to apply licenses for all the OpenText Documentum CM components. For more information about procuring the license file and configuring OTDS and license, see “[Licensing OpenText Documentum CM](#)” on page 86.

7.5 Configuring Foundation REST API to connect to HashiCorp Vault

1. Perform all the tasks as described in “[Configuring Documentum Secret Integration Service](#)” on page 49 and make sure that DSIS is running.
2. While deploying Foundation REST API on the supported application servers, update the `dfc.properties` file with the following entries:

```
dfc.dsism.enabled=true
dfc.dsism.daemon.url=http://localhost:8200/dsis
dfc.dsism.daemon.token=<token generated for the DSIS daemon agent>
```

For more information about setting the preceding entries to the `dfc.properties` file, see *OpenText Documentum Content Management - Foundation Java API Development Guide (EDCPKCL250400-DGD)*.

3. Redeploy the WAR file.
4. Restart the application server.

7.6 Accessing Foundation REST API URL in an IPv6 environment

If you want to access Foundation REST API URL in an IPv6 environment, perform the following steps:

1. Search for `-Djava.net.preferIPv4Stack=true` in all the configuration files in the web application servers’ home folder.
2. Change the value as follows:

From

```
-Djava.net.preferIPv4Stack=true
```

To

```
-Djava.net.preferIPv4Stack=false -Djava.net.preferIPv6Addresses=true
```

3. Search for `127.0.0.1` in all the configuration files in the web application servers’ home directory.
4. Change the value from `127.0.0.1` to `[::1]`.

5. To access the Foundation REST API URL in an IPv6 environment, you must provide the IPv6 address within the square brackets in the URL.

7.7 Validating Foundation REST API deployment

To validate that your RESTful Services deployment is successful, access the Home Document resource in your web browser. The Home Document resource provides an entry point to all of the other Foundation REST API resources available.

For example, to validate that a RESTful Service instance is successfully deployed on localhost at port 8080, use the following URL:

```
http://localhost:8080/dctm-rest/services
```

If you can access the preceding URL, then your installation of Foundation REST API is successful.

Chapter 8

Documentum Administrator

This documentation describes how to deploy the Documentum Administrator application. This guide is intended for administrators who are deploying Documentum Administrator. The intended audience is expected to be familiar with the Windows and Linux operating systems and are able to install and configure a Java 2 Enterprise Edition application server.

Documentum Administrator is a Documentum CM Server and repository administration tool. Documentum Administrator runs on an application server host.

For more information about how to use Documentum Administrator to administer and configure Documentum CM Server and repositories, see *OpenText Documentum Content Management - Server Administration and Configuration Guide (EDCCS250400-AGD)* and *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.

8.1 Planning for deployment

8.1.1 Required and optional supporting software

Before deploying Documentum Administrator, install the following components:

- Documentum CM Server and its associated database.
- Documentum CM Server global repository.
- Connection broker.
- Java 2 Enterprise Edition application server or servlet container.
- Oracle JDK/OpenJDK. See “[Installing JDK](#)” on page 35.

8.1.2 Typical configuration

When deployed on a single application server, Documentum Administrator requires the following network components:

- Application server host on which to deploy Documentum Administrator
- Separate Documentum CM Server host with a repository and one or more Documentum CM Servers
- Global registry repository
- Client hosts that run a supported web browser

Documentum Administrator can be deployed in supported clustered environments.

**Caution**

In production systems, for security and performance reasons, do not install the Documentum CM Server and Documentum Administrator on the same host. In addition, do not deploy web applications to the internal application server embedded in the Documentum CM Server. Non-xCelerated Composition Platform applications (such as Documentum Administrator and Documentum Webtop) cannot be deployed on the application server instance where xCelerated Composition Platform runtime is hosted because of conflicting `dfc.jar` instances on the classpath. Do not deploy Documentum Administrator on the same application server where xCelerated Composition Platform is deployed.

8.1.3 Application server host requirements

The application server host used for Documentum Administrator requires the following:

- Directory name restriction

Java does not allow directories containing the following characters, which must not appear in the folder names or paths of OpenText Documentum CM applications:

! \ / : * ? " < > |

- Content transfer folder permissions

The content transfer folder on the application server host is used to store files temporarily when they are transferred between the repository and the client machine. The default content transfer folder is specified in the `<DA_WAR_DEPLOYMENT>/wdk/app.xml` file as the value of `<server>.contentlocation`. The application server instance owner must have write permissions on this temporary content transfer location.

Some application servers require policies that grant permissions to write to these directories. Refer to deployment information for your application server to see OpenText Documentum CM policy settings.

- DNS resolution

The Domain Name Server (DNS) must be configured properly to resolve IP addresses based on the URL used to access the server.

8.1.4 Installing JDK

Download and install the supported version of Oracle JDK/OpenJDK from the Oracle JDK/OpenJDK website. For more information, see the *Oracle JDK and OpenJDK* documentation.

8.1.4.1 Using SSL communication

For information, see “[Using SSL communication](#)” on page 36.

8.1.5 Configuring Documentum Administrator to connect to HashiCorp Vault

1. Perform all the tasks as described in “[Configuring Documentum Secret Integration Service](#)” on page 49 and make sure that DSIS is running.
2. While deploying Documentum Administrator on the supported application servers, update the `dfc.properties` file with the following entries:

```
dfc.dsds.enabled=true  
dfc.dsds.daemon.url=http://localhost:8200/dsis  
dfc.dsds.daemon.token=<token generated for the DSIS daemon agent>
```

For more information about setting the preceding entries to the `dfc.properties` file, see *OpenText Documentum Content Management - Foundation Java API Development Guide (EDCPKCL250400-DGD)*.

3. Redeploy the WAR file.



Note: This step is required only if you are deploying Documentum Administrator on the Red Hat JBoss and WildFly application servers.

4. Restart the application server.

8.1.6 Customizing Documentum Administrator

Customization of Documentum Administrator is not supported.

8.2 Preparing the client hosts

8.2.1 Ensuring a certified JVM on browser clients

Browser client hosts require a certified version of the Java virtual machine (JVM) to initiate content transfer in Documentum Administrator.

8.2.2 Installing browser extension and native client application

For more information about installing browser extension and native client application, see *OpenText Documentum Web Development Kit and Webtop - Deployment Guide (EDCCLWT-AGD)*.

8.3 Preparing the application server host

8.3.1 Application servers

Before deploying Documentum Administrator, make sure that your J2EE application server or servlet container is a supported version that serves sample JavaServer Pages successfully. Your selected application server and optional external web server must be certified for Documentum Administrator.

There is no support for installing or running application servers. The documentation for each application server contains instructions on how to install, stop, start, and run the application server.

8.3.2 Setting the Java memory allocation

Java memory allocation settings affect the application server performance. OpenText recommends using the following settings:

- Minimum memory allocation

The minimum recommended Java memory allocation values for application servers on a small system are:

```
-Xms1024m -Xmx1024m
```

- Session caching

Document caching can consume at least 80 MB of memory. User session caching can consume approximately 2.5 MB to 3 MB per user. Consequently, 50 connected users can consume over 200 MB of VM memory on the application server. Increase the values to meet the demands of the expected user load.

To achieve better performance, add these parameters to the application server startup command line:

```
-Dserver  
-XX:+UseParallelOldGC
```

Performance improves because the Java client VM is not suitable for long running server jobs. The default Java garbage collector cannot clean up the heap

quickly enough, especially when the application server machine runs on multiple CPUs.

For more information about setting the Java memory allocation, see *Java* documentation.

8.3.3 Turning off failover

If your application server and environment combination does not support failover, you can disable failover in `custom\app.xml` by adding the following element:

```
<failover>
  <enabled>false</enabled>
</failover>
```

If you do not turn off failover, the failover validation messages are logged in the application server log, but these validations do not interfere with operations. Do not use the application in a failover environment.

8.3.4 Preparing environment variables for non-default Foundation Java API locations

The `dfc.data.dir` Foundation Java API environment variable specifies the base location for content transfer on the application server host. This location is specified as the value of the `dfc.data.dir` key in the `dfc.properties` file located within the application WAR file in `WEB-INF/classes`. If this variable is not set in the environment for the application server, the default location is the `Documentum` subfolder of the current working folder. The current working folder contains the application server executable. For example, in Apache Tomcat, the location is `CATALINA_HOME/bin`.

By default, the checkout and export folders are subfolders of the `dfc.data.dir` folder, and the user folder is the same as `dfc.data.dir`. If you wish to use non-default locations for these folders, create environment variables for `dfc.checkout.dir`, `dfc.export.dir`, and `dfc.user.dir`. The default value of `dfc.registry.mode`, which corresponds to the `dfc.registry.mode` key in the `dfc.properties` file, is `file`. By default, the full path to this file is `dfc.user.dir/documentum.ini`. For a non-default file name or location, specify it as the value of the `dfc.registry.file` environment variable.

8.3.5 Configuring Apache Tomcat

Disable tag reuse in Apache Tomcat in the `web.xml` file in the `/conf` folder. Find the JSP servlet entry in the `web.xml` file. Add the `enablePooling` initialization parameter, disable pooling, and then restart the web application server:

```
<servlet>
  <servlet-name>jsp</servlet-name>
  <servlet-class>org.apache.jasper.servlet.JspServlet</servlet-class>
  <init-param>
    <param-name>enablePooling</param-name>
    <param-value>false</param-value>
  </init-param>
  <init-param>
</servlet>
```

If you use a Windows service to start Tomcat, update the `tomcat\bin\tomcat10w.exe` file. Open `tomcat10w` and in **Java9 options** on the **Java** tab, add the following parameters:

```
--add-opens=java.base/java.net=ALL-UNNAMED
--add-opens=java.base/java.lang.ref=ALL-UNNAMED
--add-opens=java.naming/com.sun.jndi.toolkit.url=ALL-UNNAMED
--add-exports=java.base/sun.security.provider=ALL-UNNAMED
--add-exports=java.base/sun.security.pkcs=ALL-UNNAMED
--add-exports=java.base/sun.security.x509=ALL-UNNAMED
--add-exports=java.base/sun.security.util=ALL-UNNAMED
--add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED
--add-opens=java.base/java.lang=ALL-UNNAMED
```

For the JDK 17 support, add the following parameters in the `rem Configure module` start-up parameters section in the `catalina.bat` file:

```
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.net=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.lang.ref=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.naming/com.sun.jndi.toolkit.url=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.provider=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.pkcs=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.x509=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.util=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.xml.crypto/com.sun.org.apache.xml.internal.security
=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
```

```
--add-opens=java.base/java.lang=ALL-UNNAMED"
```

For the JDK 21 support, add the following parameter in the `rem Configure` module start-up parameters section in the `catalina.bat` file:

```
set "JAVA_OPTS=%JAVA_OPTS% -Djava.locale.providers=COMPAT,SPI"
```

8.3.6 Configuring WildFly

1. If available, delete the `dfc.keystore` and `wdk.keystore` files in `<WildFly_Home>\bin` on Windows or `<WildFly_Home>/bin` on Linux.
2. Before creating the `web-inf-classes.jar` file, update the `dfc.properties` file.

On Windows, create an encrypted password using the following command:

```
java  
--add-exports=java.base/sun.security.provider=ALL-UNNAMED  
--add-exports=java.base/sun.security.pkcs=ALL-UNNAMED  
--add-exports=java.base/sun.security.x509=ALL-UNNAMED  
--add-exports=java.base/sun.security.util=ALL-UNNAMED  
--add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED  
--add-opens=java.base/java.lang=ALL-UNNAMED  
-cp ../../lib/dfc.jar; ./lib/commons-io-<packaged-version>.jar; ./lib/commons-lang-<packaged-version>.jar; ./lib/bcprov-jdk<packaged-version>.jar  
TrustedAuthenticatorTool <PASSWORD>
```

where `<packaged-version>` is the supported version packaged with the Documentum Administrator WAR file.

On Linux, create an encrypted password using the following command where `<packaged-version>` is the supported version packaged with the Documentum Administrator WAR file.

```
java  
--add-exports=java.base/sun.security.provider=ALL-UNNAMED  
--add-exports=java.base/sun.security.pkcs=ALL-UNNAMED  
--add-exports=java.base/sun.security.x509=ALL-UNNAMED  
--add-exports=java.base/sun.security.util=ALL-UNNAMED  
--add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED  
--add-opens=java.base/java.lang=ALL-UNNAMED  
-cp ../../lib/dfc.jar; ./lib/commons-io-<packaged-version>.jar; ./lib/commons-lang-<packaged-version>.jar; ./lib/bcprov-jdk<packaged-version>.jar  
TrustedAuthenticatorTool <PASSWORD>
```

If HashiCorp Vault is enabled, the presets and preferences password are retrieved from the HashiCorp Vault server. If HashiCorp Vault is not enabled, you must manually update the encrypted password in the `wdk/app.xml` file for presets and preferences passwords.

3. Move the keystore files from `<WebApp_Root>\WEB-INF\classes` on Windows or `<WebApp_Root>/WEB-INF/classes` on Linux to the `bin` folder in the `<WildFly_Home>` folder.

4. Copy the contents of the `classes` folder from `<WebApp_Root>\WEB-INF\classes` on Windows or `<WebApp_Root>/WEB-INF/classes` on Linux to a temporary location (for example, `Temp-Loc`).

Run the following command at `Temp-Loc` to create a `web-inf-classes` JAR file:

```
jar -cvf web-inf-classes.jar *
```

5. Copy the `web-inf-classes.jar` file to `<WebApp_Root>\WEB-INF\lib` on Windows or `<WebApp_Root>/WEB-INF/lib` on Linux.
6. Delete the `classes` folder from `<WebApp_Root>\WEB-INF` on Windows or `<WebApp_Root>/WEB-INF` on Linux.
7. Disable the tag pooling. To disable the tag pooling, open the `standalone.xml` file in `<WildFly_Home>\standalone\configuration` on Windows or `<WildFly_Home>/standalone/configuration` on Linux in an editor, search for the `jsp-config` tag, update the tag to `jsp-config tag-pooling=false/`, and then save the file.
8. Configure the binding address by replacing `127.0.0.1` with the application server host IP address in the `wsdl-host` and `interfaces` tags in the `standalone.xml` file.
9. Add the following parameters in the `%WILDFLY_HOME%\bin\standalone.conf.bat` file:

```
set "JAVA_OPTS=%JAVA_OPTS% -Djava.locale.providers=COMPAT,SPI  
--add-opens=java.base/java.lang=ALL-UNNAMED  
--add-opens=java.base/java.lang.invoke=ALL-UNNAMED  
--add-exports=java.base/sun.security.provider=ALL-UNNAMED  
--add-exports=java.base/sun.security.pkcs=ALL-UNNAMED  
--add-exports=java.base/sun.security.x509=ALL-UNNAMED  
--add-exports=java.base/sun.security.util=ALL-UNNAMED  
--add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED"
```
10. Run the following command at `<WebApp_Root>` to repackage the Documentum Administrator WAR file:

```
jar -cvf da.war *
```

8.3.7 Configuring Red Hat JBoss Enterprise Application Platform (JBoss EAP)

1. Download the JBoss EAP 8 Update 4 patch file named `jboss-eap-8.0.4-runtime-maven-repository.zip` from the Red Hat website.
2. Extract the `jboss-eap-8.0.4-runtime-maven-repository.zip` file to a temporary location.
3. Apply the JBoss EAP 8 Update 4 patch.

Windows

For example, from `<JBoss EAP_Home>/bin`, run the following command as an administrator:

```
jboss-eap-installation-manager.bat update perform ^
--dir C:\jboss-eap-8.0 ^
--repositories mrrc::file:C:\jboss-eap-8.0.4.GA-runtime-maven-repository
\maven-repository ^
--offline
```

Linux

For example, from `<JBoss EAP_Home>/bin`, run the following command as an administrator:

```
./jboss-eap-installation-manager.bat update perform \
> --dir ../../jboss-eap-8.0 \
> --repositories mrrc::file:C:\jboss-eap-8.0.4.GA-runtime-maven-repository
\maven-repository \
> --offline
```

4. If available, delete the `dfc.keystore` and `wdk.keystore` files in `<JBoss EAP_Home>/bin` on Windows or `<JBoss EAP_Home>/bin` on Linux.
5. Before creating the `web-inf-classes.jar` file, update the `dfc.properties` file.

On Windows, create an encrypted password using the following command:

```
java
--add-exports=java.base/sun.security.provider=ALL-UNNAMED
--add-exports=java.base/sun.security.pkcs=ALL-UNNAMED
--add-exports=java.base/sun.security.x509=ALL-UNNAMED
--add-exports=java.base/sun.security.util=ALL-UNNAMED
--add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED
--add-opens=java.base/java.lang=ALL-UNNAMED
-cp ../../lib/dfc.jar;../../lib/commons-io-<packaged-version>.jar;../../lib/commons-lang-<packaged-version>.jar;../../lib/bcprov-jdk<packaged-version>.jar
TrustedAuthenticatorTool <PASSWORD>
```

where `<packaged-version>` is the supported version packaged with the Documentum Administrator WAR file.

On Linux, create an encrypted password using the following command:

```
java
--add-exports=java.base/sun.security.provider=ALL-UNNAMED
--add-exports=java.base/sun.security.pkcs=ALL-UNNAMED
```

```
--add-exports=java.base/sun.security.x509=ALL-UNNAMED  
--add-exports=java.base/sun.security.util=ALL-UNNAMED  
--add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED  
--add-opens=java.base/java.lang=ALL-UNNAMED  
-cp ../../lib/dfc.jar:../../lib/commons-io-<packaged-version>.jar: ../../lib/commons-lang-<packaged-version>.jar: ../../lib/bcprov-jdk<packaged-version>.jar  
TrustedAuthenticatorTool <PASSWORD>
```

where *<packaged-version>* is the supported version packaged with the Documentum Administrator WAR file.

If HashiCorp Vault is enabled, the presets and preferences password are retrieved from the HashiCorp Vault server. If HashiCorp Vault is not enabled, you must manually update the encrypted password in the `wdk/app.xml` file for presets and preferences passwords.

6. Move the keystore files from `<WebApp_Root>\WEB-INF\classes` on Windows or `<WebApp_Root>/WEB-INF/classes` on Linux to the `bin` folder in the `<JBoss EAP_Home>` folder.
7. Copy the contents of the `classes` folder from `<WebApp_Root>\WEB-INF\classes` on Windows or `<WebApp_Root>/WEB-INF/classes` on Linux to a temporary location (for example, `Temp-Loc`).

Run the following command at `Temp-Loc` to create a `web-inf-classes` JAR file:

```
jar -cvf web-inf-classes.jar *
```

8. Copy the `web-inf-classes.jar` file to `<WebApp_Root>\WEB-INF\lib` on Windows or `<WebApp_Root>/WEB-INF/lib` on Linux.
9. Delete the `classes` folder from `<WebApp_Root>\WEB-INF` on Windows or `<WebApp_Root>/WEB-INF` on Linux.
10. Disable the tag pooling. To disable the tag pooling, open the `standalone.xml` file in `<JBoss EAP_Home>\standalone\configuration` on Windows or `<JBoss EAP_Home>/standalone/configuration` on Linux in an editor, search for the `jsp-config` tag, update the tag to `jsp-config tag-pooling=false/`, and then save the file.
11. Configure the binding address by replacing `127.0.0.1` with the application server host IP address in the `wsdl-host` and `interfaces` tags in the `standalone.xml` file.
12. Run the following command at `<WebApp_Root>` to repackage the Documentum Administrator WAR file:

```
jar -cvf da.war *
```



Note: When deploying Documentum Administrator in JBoss EAP, add `<path name="com/sun/jndi/url/rmi"/>` to `<JBoss EAP deployment folder>\modules\system\layers\base\sun\jdk\main\module.xml` in the `paths` tag.

8.3.8 Configuring IBM WebSphere Liberty

For information about configuring IBM WebSphere Liberty, see *OpenText Documentum Web Development Kit and Webtop - Deployment Guide (EDCCLWT-AGD)*.

Notes

- For the JDK 17 support, add the following parameters in the `server.bat` file:

```
set "JAVA_OPTS=%JAVA_OPTS%
--add-opens=java.base/java.lang=ALL-UNNAMED"
set "JAVA_OPTS=%JAVA_OPTS%
--add-opens=java.base/java.io=ALL-UNNAMED"
set "JAVA_OPTS=%JAVA_OPTS%
--add-opens=java.base/java.util=ALL-UNNAMED"
set "JAVA_OPTS=%JAVA_OPTS%
--add-opens=java.base/java.util.concurrent=ALL-UNNAMED"
set "JAVA_OPTS=%JAVA_OPTS%
--add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.net=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.lang.ref=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.naming/com.sun.jndi.toolkit.url=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.provider=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.pkcs=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.x509=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.util=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS%
--add-opens=java.base/java.lang=ALL-UNNAMED"
```

- Update the `featureManager` property in the `server.xml` file as follows:

```
<featureManager>
<feature>jakartaee-10.0</feature>
</featureManager>
...
<proxyHTTP host=<provide IP address of host machine> port="3128" />
<proxyHTTPS host=<provide IP address of host machine> port="3128" />
```

- Set the value of `response_content_length_header` in the `wdk/app.xml` file to `true` as follows:

```
<response_content_length_header>true</response_content_length_header>
```

If you encounter any issues with certification, use `try_native_first`.

8.3.9 Preparing the application server for Java 2 security

If you plan to use Java 2 security for securing access to available system resources in your Documentum Administrator installation, then use the Java policy configuration file that is packaged with your application server. To help you update the Java policy configuration file of the application server, an example policy template file, `Webtop.example.java.policy`, is included in the Documentum Administrator installation. The file specifies the permissions required to access the Documentum Administrator classes. The `Webtop.example.java.policy` file is included in the `da.war` file, and gets extracted into the `<da_app_root>` folder.



Caution

Do not omit any permission specified in the `Webtop.example.java.policy` file while incorporating the permissions in the application server Java policy configuration file. Otherwise, Documentum Administrator might fail to start or some features might fail to work.

1. Navigate to `<da_app_root>\Webtop.example.java.policy` and identify the permissions that must be incorporated into the application server security policy file.
2. Navigate to the policy file of your application server.
Based on the syntax and locations specified in the application server documentation, add or update the permissions identified in the `Webtop.example.java.policy` file in the policy file of the application server.
3. Configure your application server to select the security policy files.

8.3.10 Preparing to use an external web server

External web servers are sometimes used as a front end to the application server. If the external server does not support HTTP 1.1 chunked encoding, configure UCF to use an alternative chunked encoding.

If you are deploying in a managed server or network deployment environment, the external web server must provide session affinity support.

8.4 Deploying Documentum Administrator

8.4.1 Prerequisites

Make sure that you follow all requirements described in the pre-deployment checklist.

Use the following checklist to verify that you have performed all required tasks when you deploy or upgrade Documentum Administrator:

Requirement	For more information
Review the release notes for the release you are deploying or to which you are upgrading.	<i>Release Notes</i>
Validate your hardware configuration.	<i>Release Notes</i>
Validate your application server and clients operating systems.	<i>Release Notes</i>
Create required operating system accounts.	Network administrators
Verify that the application server instance owner has write permissions on the temporary content transfer directories.	Network administrators
Determine the repositories to which end users connect.	Network administrators
Determine the connection brokers to which the repositories project.	Network administrators
Determine which repository on the network is the global registry repository, and obtain the global registry user name and password.	Network administrators
Determine which repositories are used to store presets and user preferences.	Network administrators
Determine whether language packs are required.	<i>Release Notes</i>
Prepare the application server host and application server software according to the vendor requirements.	Specific requirements should be met for the application server host
Disable the IP Helper service from the Windows Services console and restart the machine. This method disables the Teredo Tunneling Pseudo-Interface.	Network administrators

8.4.2 Deploying the WAR file

Before Documentum Administrator can connect to repositories, provide the connection broker and global registry values in the `dfc.properties` file.

Documentum Administrator requires a Documentum CM Server version 6.0 or later global registry. The global registry is a central repository that serves the following purposes:

- Deploys service-based business objects (SBOs)
- Stores network location objects
- Stores application presets, unless another repository is configured in the `app.xml` file
- Stores persistent user preferences, unless another repository is configured in the `app.xml` file

For more information about enabling a repository as a global registry, see [step 18](#) in [“Creating a repository” on page 59](#). You can copy information from the `dfc.properties` file that the Documentum CM Server installer generated onto your global registry host. The generated `dfc.properties` file contains the connection broker address and the encrypted global registry user login information.

1. Configure the connections in the `dfc.properties` file before deployment:
 - a. Unpack the Documentum Administrator WAR file.
 - b. Open the `WEB-INF/classes/dfc.properties` file.
 - i. Add the fully qualified host name for the connection broker to the following key. You can increment the index number within brackets to add backup hosts.

```
dfc.docbroker.host[0]=host_name
```
 - ii. To use a port for the connection broker other than the default of 1489, add a port key to the `dfc.properties` file as follows:

```
dfc.docbroker.port=port_number
```
 - iii. Add the global registry repository name to the following key:

```
dfc.globalregistry.repository=repository_name
```
 - iv. Add the user name of the `dm_bof_registry` user to the following key:

```
dfc.globalregistry.username=dm_bof_registry_user_name
```

The global registry user, who has the user name `dm_bof_registry`, has read access only to objects in the `/System/Modules` and `/System/NetworkLocations`.
 - v. If HashiCorp Vault is not enabled, add an encrypted password value for the following key:

```
dfc.globalregistry.password=encrypted_password
```

You can either copy the user name and encrypted password from the `dfc.properties` file on the global registry Documentum CM Server host or you can select another global registry user and encrypt the password using the following command:

```
java -cp dfc.jar com.documentum.fc.tools.RegistryPasswordUtils  
password_to_be_encrypted
```

 **Note:** The folder containing the `javaw.exe` file must be on the system path.

- vi. If HashiCorp Vault is enabled, perform the following steps:
 - A. Perform all the steps as described in “[Configuring Documentum Secret Integration Service](#)” on page 49.
 - B. Update the `dfc.properties` file with the following entries:


```
dfc.dsds.enabled=true  
dfc.dsds.daemon.url=http://localhost:8200/dsis  
dfc.dsds.daemon.token=<token generated for the DSIS daemon agent>
```
- vii. If the Documentum CM Server, connection broker, and the repository are configured in the non-anonymous SSL mode, then provide the following parameters in the `dfc.properties` file:
 - A. Add the secure connection mode and set it to secure first as follows:


```
dfc.session.secure_connect_default = try_secure_first
```
 - B. Add the trust store path as follows:


```
dfc.security.ssl.truststore=<dfc truststore path>
```
 - C. If HashiCorp Vault is not enabled, add the trust store password as follows:


```
dfc.security.ssl.truststore_password=<password>
```
 - D. If HashiCorp Vault is enabled, add the trust store password as follows:


```
dfc.security.ssl.truststore_password=<secret_name>/<key_name>
```
 - E. Specify whether to use the existing trust store as follows:


```
dfc.security.ssl.use_existing_truststore=<true or false>
```
 - F. Specify the crypto repository to connect as follows:


```
dfc.crypto.repository=repository_name
```
 - G. On the global registry repository host, locate the Documentum CM Server installation folder. On Windows hosts, the default installation folder is `C:\Documentum`. On Linux hosts, the `$DOCUMENTUM` environment variable specifies this folder.
 - H. Open the `config\dfc.properties` file.
 - I. If HashiCorp Vault is not enabled, copy the following keys and their values from the file:

```

dfc.docbroker.host[0]=address
dfc.docbroker.port[0]=port_number
dfc.globalregistry.repository=repository_name
dfc.globalregistry.username=user_name
dfc.globalregistry.password=encrypted_password
dfc.crypto.repository=repository_name
dfc.session.secure_connect_default=try_secure_first

```

- J. If HashiCorp Vault is enabled, copy the following keys and their values from the file:

```

dfc.docbroker.host[0]=address
dfc.docbroker.port[0]=port_number
dfc.globalregistry.repository=repository_name
dfc.globalregistry.username=user_name
dfc.globalregistry.password=<secret_name>/<key_name>
dfc.crypto.repository=repository_name
dfc.session.secure_connect_default=try_secure_first

```

- K. Add `dfc.session.allow_trusted_login` and set the value to `false`.
- L. Add `dfc.security.ssl.use_anonymous_cipher` and set the value to `true`.
- M. Enable and configure the optional presets and preferences repositories in the `dfc.properties` file.

By default, presets and persistent preferences are stored in the global repository. For better performance, you can configure your Documentum Administrator to use different repositories for presets and persistent preferences.

Add your preferences repository settings to the `app.xml` file in the `/custom` folder of the application. Copy the entire `<preferencesrepository>` element from the `/wdk/app.xml` file into the `/custom/app.xml` file and then specify your repository.

Table 8-1: Preferences configuration elements

Element	Description
<code><preferencesrepository></code>	Contains a <code><repository></code> element. If this element is not present, user preferences are stored in the global repository, which can slow down performance.
<code><repository_path></code>	Specifies the path within the preference repository in which to store preferences. If the path does not exist at application startup, then it is created.
<code><repository></code>	Specifies the repository in which to store preferences, preferably not the global repository.

To enable users to create presets using the presets editor, assign the `dmc_wdk_presets_coordinator` role to those users.

To configure the password in presets and preferences repositories, perform the following steps:

- I. Login to IAPI as an administrator to change the default passwords of `dmc_wdk_presets_owner` and `dmc_wdk_preferences_owner` users in Documentum CM Server.
 - To change the password for the `dmc_wdk_presets_owner` user, run the following command:


```
retrieve,c,dm_user where user_name='dmc_wdk_presets_owner';
set,c,1,user_password
<enter new password>
save,c,1
```
 - To change the password for the `dmc_wdk_preferences_owner` user, run the following command:


```
retrieve,c,dm_user where
user_name='dmc_wdk_preferences_owner';
set,c,1,user_password
<enter new password>
save,c,1
```
-  **Note:** Make sure that you follow the password complexity rules. “[Password complexity rules](#)” on page 52 contains detailed information.
- II. Encrypt the passwords in Documentum Administrator using TrustedAuthenticatorTool located at `WEB-INF/classes`.
 - Windows: Navigate to the `WEB-INF/classes` folder and run the following command format:


```
java
--add-exports=java.base/sun.security.provider
=ALL-UNNAMED
--add-exports=java.base/sun.security.pkcs
=ALL-UNNAMED
--add-exports=java.base/sun.security.x509
=ALL-UNNAMED
--add-exports=java.base/sun.security.util
=ALL-UNNAMED
--add-exports=java.base/sun.security.tools.keytool
=ALL-UNNAMED
--add-opens=java.base/java.lang
=ALL-UNNAMED
-cp ../../lib/dfc.jar;../../lib/commons-io-<packaged-version>.jar; ../../lib/commons-lang<packaged-version>.jar; ../../lib/bcprov-jdk<packaged-version>.jar
TrustedAuthenticatorTool <PASSWORD>
```

where `<packaged-version>` is the supported version packaged with the Documentum Administrator WAR file.
 - Linux: Navigate to the `WEB-INF/classes` folder and run the following command format:

```
java
--add-exports=java.base/sun.security.provider
=ALL-UNNAMED
--add-exports=java.base/sun.security.pkcs
=ALL-UNNAMED
--add-exports=java.base/sun.security.x509
=ALL-UNNAMED
--add-exports=java.base/sun.security.util
=ALL-UNNAMED
--add-exports=java.base/sun.security.tools.keytool
=ALL-UNNAMED
--add-opens=java.base/java.lang
=ALL-UNNAMED
-cp .../lib/dfc.jar:.../lib/commons-io-<packaged-
version>.jar: .../lib/commons-lang-<packaged-
version>.jar: .../lib/bcprov-jdk<packaged-version>.jar
TrustedAuthenticatorTool <PASSWORD>
```

where *<packaged-version>* is the supported version packaged with the Documentum Administrator WAR file.

- III. If HashiCorp Vault is enabled, the presets and preferences password are retrieved from the HashiCorp Vault server. If HashiCorp Vault is not enabled, you must manually update the encrypted password in the `wdk/app.xml` file for presets and preferences passwords.

- IV. Save the `wdk/app.xml` file.

- viii. Save the `dfc.properties` file.



Note: If you create a WAR file from this application folder, make sure that any paths that you specify in the `dfc.properties` file are valid folders on the application server. In addition, make sure that the application server instance owner has the `write` permission on the specified folders.

2. **Optional** Add language packs to and configure them in the Documentum Administrator WAR file.
 - a. Unpack the language pack ZIP file into the root Documentum Administrator WAR folder.
 - b. Add the required locale for `<supported_locales>` in `da/custom/app.xml`.
For example, for the Japanese language pack, add `<locale>ja_JP</locale>` to `da/custom/app.xml` as follows:

```
<supported_locales>
    <locale>en_US</locale>
    <locale>ja_JP</locale>
<supported_locales>
```
3. Rearchive the WAR file.



Note: This step is required only if you are deploying Documentum Administrator on the Red Hat JBoss and WildFly application servers.

4. Deploy the WAR file according to the deployment instructions in your application server documentation.
5. **Optional** If you have installed the Japanese language pack and the repository is on a non-Japanese operating system, then you must populate the data dictionary with the Japanese data dictionary files by running `dd_populate.ebs` on a Japanese operating system. For more information about populating the data dictionary in a repository from a non-English host, see *OpenText Documentum Content Management - Server Administration and Configuration Guide* (EDCCS250400-AGD).



Note: If you have created a repository on a Japanese operating system, then the data dictionary is automatically populated with the Japanese data dictionary files.

8.4.3 Configuring and using SAP solutions in Documentum Administrator

The WebAdmin component to configure and administer Documentum Archive Services for SAP Solutions and Documentum Content Services for SAP Solutions is included in the Documentum Administrator WAR file.

After the successful deployment of Documentum Administrator, you can view the **Content Services for SAP** node in Documentum Administrator.

Install Documentum Archive Services for SAP Solutions and Documentum Content Services for SAP Solutions as described in *OpenText Documentum Archive Services for SAP Solutions - Installation Guide* (EDCCOSAPAR250400-IGD) and *OpenText Documentum Content Services for SAP Solutions - Installation Guide* (EDCCOSAPCS250400-IGD) respectively.

8.4.4 Enabling Foundation Java API memory optimization

Foundation Java API diagnostics are enabled by default. To free up memory resources, disable the `dfc.diagnostics.resources.enable` parameter in the `dfc.properties` file. Add the following line to your `dfc.properties` file:

```
dfc.diagnostics.resources.enable=false
```

8.4.5 Configuring UCF

For information about configuring UCF, see *OpenText Documentum Web Development Kit - Development Guide (EDCPKCLWT160709-PGD)*.



Note: UCF has a dependency on Microsoft Visual C and C++ runtime libraries. Web Development Kit based applications such as Documentum Administrator that use UCF, require the Microsoft Visual C++ Redistributable as a prerequisite. Microsoft has consolidated the Microsoft Visual C++ Redistributable packages for versions 2015 to 2022 into a single package. You must download this package from the Microsoft website for the appropriate architecture (x86 or x64).

8.4.6 Enabling retention of folder structure and objects on export

To enable retaining the same folder structure (as the one in the repository) and the contained objects on the local file system when the parent folder is exported, add the following element to the app.xml file in the custom folder:

```
<deepexport>
<enabled>true</enabled>
</deepexport>
```

The default value is false.

8.4.7 Enabling external searches

To allow users to search external sources, an administrator must configure a connection to a Federated Search Services server.

8.4.7.1 Configuring the connection to the search server

To enable the Federated Search Services Server to query external sources:

1. Unpack the client application WAR file.
2. Open the dfc.properties file in WEB-INF/classes.
3. Enable the Federated Search Services server using the setting as follows:

```
dfc.search.ecis.enable=true
```
4. Specify the RMI Registry host for the Federated Search Services server using the setting as follows:

```
dfc.search.ecis.host=<host_IP>
```



```
dfc.search.ecis.port=<port>
```

where

- <host_IP> is IP address or machine name of the Federated Search Services server.

- <port> is the port number that accesses the Federated Search Services server. The default port is 3005.

8.4.7.2 Configuring the connection to the backup search server

You can set a backup server in case the primary Federated Search Services server is unreachable. If a Foundation Java API application cannot connect to the primary Federated Search Services server to query external sources, the backup server is contacted. You can define the time period after which the application tries to connect again to the primary server. To define the backup server, specify the RMI host and port in the `dfc.properties` file.

- `dfc.search.ecis.backup.host`: Host of the backup Federated Search Services server. The default value is `localhost`.
- `dfc.search.ecis.backup.port`: Port of the backup Federated Search Services server. The default value is 3005.
- `dfc.search.ecis.retry.period`: Waiting period before retrying to connect to the primary Federated Search Services server. This time is in milliseconds. The default value is 300000.

8.4.8 Requirement for full-text indexing

If you use Documentum Administrator to administer full-text indexing, a fully-qualified domain name must identify where the application server is installed. For example, the host name `tristan.documentum.com` is acceptable, but an IP address such as `123.45.6.789` is not acceptable.

8.4.9 Resource Management availability

If Resource Management is installed, the RMI port used to manage the resources must be open. If a firewall separates the machine hosting Documentum Administrator from the remote resource, the RMI port must be open and not obstructed by the firewall. In addition, configure the Domain Name Server to resolve the IP addresses based on the URL used to access the server.

8.4.10 Enable presets for Administrator Access and Resource Management

When deploying Documentum Administrator, the Enable/Disable Presets flag in the application custom `app.xml` file must be set to `True`, as it impacts the following functionality:

- Administrator Access: If the preset flag is disabled, the Administrator Access functionality in Documentum Administrator is disabled.
- Resource Management: If the preset flag is disabled, the ability to dynamically access or modify the resource agent information in the global registry is disabled. Resource Management still functions for resource agents defined in the static

configuration file, but administrators cannot add, modify, or delete resource agents using Documentum Administrator.



Note: The Enable/Disable Presets flag in the custom app.xml file for Documentum Administrator overrides the presets flag in Web Development Kit.

8.4.11 Data Visualization reporting solutions

Data Visualization Reports (DVR) is a reporting solution that helps you to access the OpenText Documentum CM metadata and retrieve information in the form of reports. Using DVR, the administrators and business users can create new reports, view, run, and edit the Out-Of-The-Box (OOTB) reports.

DVR supports the following:

- Configuring Information Hub (iHub) reporting servers.
- Designing different reports for repository using OpenText Analytics Designer.
- Publishing reports on the iHub server.
- Connecting to the iHub servers and running various types of reports.

You can access the metadata using the OpenText Analytics Designer and iHub to support the following:

- Reporting against OpenText Documentum CM types or registered tables.
- Using multiple types or tables within a single report.

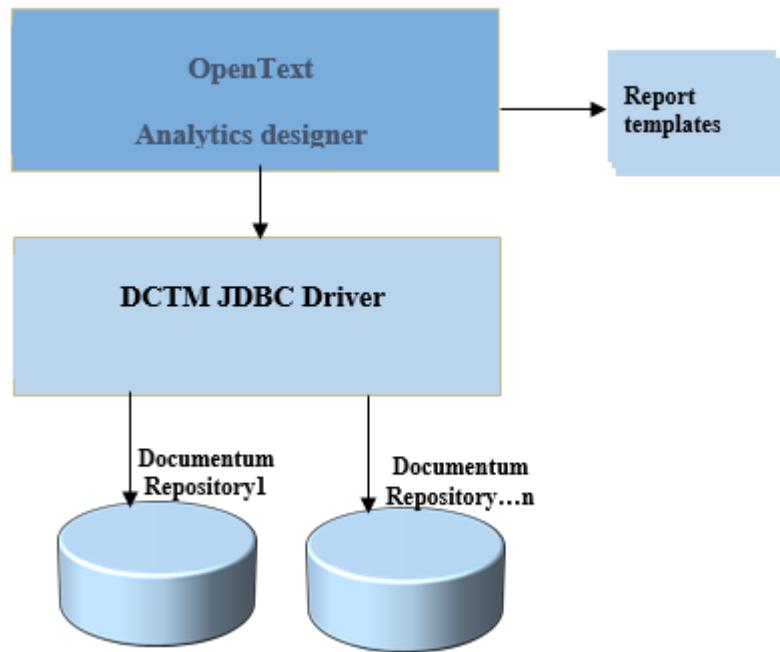


Figure 8-1: Design of reporting solutions

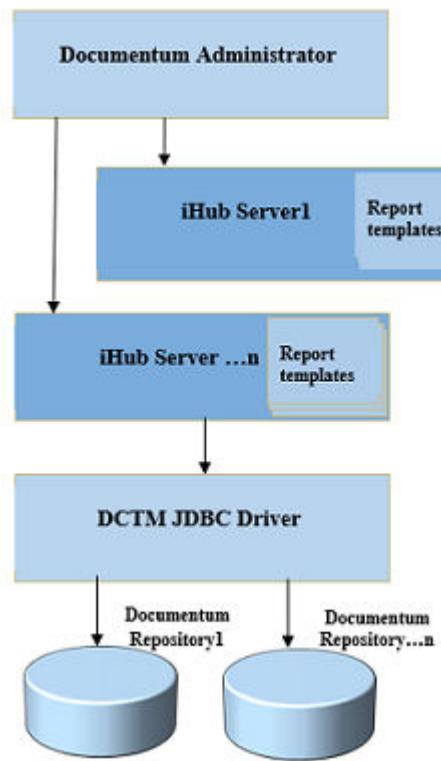


Figure 8-2: Runtime of reporting solutions

8.4.11.1 Prerequisites

- To design DVR, you must install the following software:
 - Documentum CM Server
 - OpenText Analytics Designer 16.2
 - Oracle JDK 1.8
 - Java Database Connectivity (JDBC)
 - Foundation Java API
- To run the reports, you must install the following software:
 - Documentum CM Server
 - iHub Server 16.2
 - Java Database Connectivity (JDBC)
 - Documentum Administrator
 - Foundation Java API

- DVR components

8.4.11.2 OpenText Analytics Designer

OpenText Analytics Designer, a report designer tool, is used to design the reports to connect to repository. For reporting solutions, Documentum JDBC driver is used to connect to repository. The connection details of reports is dynamically set at runtime while running the reports to the OpenText Documentum CM application.

8.4.11.3 iHub

iHub is a scalable analytics and data visualization platform that enables you to design, deploy, and manage secure, interactive web applications, reports, and dashboards fed by multiple data sources. iHub supports high volume of users, and its integration APIs enable embedded analytic content in any application, displayed on any device. The data visualization OOTB reports are published on to the iHub server. iHub authentication mechanism is customized to perform the authentication against both repository and iHub server as follows:

- The repository and user against which the reports run, is dynamically set at run time depending on the repository and user who has logged into Documentum Administrator application.
- A configuration file is maintained in the iHub server to map the OpenText Documentum CM user with the iHub user for seamless iHub portal authentication.
- For security purposes, the repository password is not passed from Documentum Administrator to iHub in normal communication. Users need to provide the OpenText Documentum CM password to open the iHub portal page. If iHub is configured in the SSL mode, then OpenText Documentum CM login token is passed from Documentum Administrator to iHub and users are automatically logged into iHub.

8.4.11.4 Designing new reports on OpenText Documentum CM data using OpenText Analytics Designer

1. Make sure that you have installed OpenText Analytics Designer.
2. Configure the OpenText Analytics Designer for JDBC driver.
 - a. Extract the `DataVisualizationReports.zip` file that is part of the Documentum CM Server installer package.
The ZIP file contains the following files:
 - `Dvr_iHubApp.zip`
 - `OTDctmReportTemplates.zip`
 - `ReportPublisher.zip`
 - b. Extract all the extracted ZIP files and retain the folder names as the name of the ZIP files.

- c. Copy all the files from Dvr_iHubApp\dmjdbc_driver to <Analytical Designer Home>\AnalyticsDesigner\plugins\org.eclipse.birt.report.data.oda.jdbc_<version>\drivers.
 - d. Modify the dfc.properties file in <Analytical Designer Home>\AnalyticsDesigner\plugins\org.eclipse.birt.report.data.oda.jdbc_<version>\drivers to point to the appropriate connection broker.
 - e. Create new JAR file (for example, properties_bundle.jar).
 - f. Move the dfc.properties and dmjdbc.properties files (remove from <Analytical Designer Home>\AnalyticsDesigner\plugins\org.eclipse.birt.report.data.oda.jdbc_<version>\drivers) into the new JAR file you created in **step 2.e**. Then, add the JAR file to the <Analytical Designer Home>\AnalyticsDesigner\plugins\org.eclipse.birt.report.data.oda.jdbc_<version>\drivers folder.
3. Open OpenText Analytics Designer.
 4. Create a new BIRT project of type Report and select the simple listing report template.
 5. Create a new data source of type JDBC Data source and provide the following values for the driver information:
 - Driver class: com.documentum.oca.jdbc.jdbc20.DjdbcDriver
 - Database URL: jdbc:documentum:oca:docbase@<repository_name>
 - Repository user name and password
 6. Create a new dataset from the data source you created in **step 5** and input the DQL. For example, select object_name,owner_name from dm_document.
 7. Drag and drop the appropriate dataset columns into the report. For more information about designing the report, see *OpenText Analytics Designer User Guide*.

8.4.11.5 Types of OOTB reports

OOTB reports contain reports defined with common data available across all implementations of OpenText Documentum CM. This section describes each category of OOTB reports with a description of each report DQL query.

OOTB reports can be categorized as follows:

- Repository Objects Reports: A family of reports that display details pertaining to documents that have the maximum size (in bytes). All reports in this category display data in a tabular format, and include pie charts for graphical representation of information.
- Workflow Reports: A family of reports that display details pertaining to workflows, current state of workflows (such as Unknown, Dormant, Running, Finished, Terminated, Running, Active, Acquired, and Paused states), start and end dates, and details about Workflow owner. All reports in this category

display data in a tabular format, and include pie chart for graphical representation of information.

- Retention Management Reports: A family of reports that specify details on retention policy objects such as Promotion, Disposition, Retention, and Physical Record objects. All reports in this category display data in a tabular format.
- Administration Reports: A family of reports that specify details to administration.

You can navigate to any of these report category types from the iHub reporting server portal page and perform various operations. For more information about navigating to report category types, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250400-UGD)*.



Note: If a report contains large data such as 1 million records, then the fetching of the report and rendering may take time. Set an appropriate value such as 1000 (in seconds) for MaxSyncJobRuntime in `<iHub Home>\modules\BIRTiHub\iHub\shared\config\acservrconfig.xml` in iHub server.

8.4.11.6 Configuring DVR

1. Make sure that you have installed Documentum CM Server and Documentum Administrator.
2. Extract the `DataVisualizationReports.zip` file.

The ZIP file contains the following files:

- `Dvr_iHubApp.zip`
- `OTDctmReportTemplates.zip`
- `ReportPublisher.zip`

3. Extract all the extracted ZIP files and retain the folder names as the name of the ZIP files.

8.4.11.6.1 Publishing DVR OOTB reports

1. Open the `ReportsPublisher` folder.
2. Configure the following parameters in the `config.properties` file:
 - URL: URL of the iHub server.
Provide the appropriate `<IP address or host name of the iHub server>`.
 - REPORTS_REMOTE_FOLDER: Name of the folder in the iHub server in to which the publisher publishes the OOTB reports. The default folder is `/Home/administrator`.
 - REPORTS_LOCAL_FOLDER: Local folder of the report templates. The default folder is `../OTDctmReportTemplates/OTDctm`.



Note: If you have followed the [step 2](#) and [step 3](#) steps as is, then you need not change the default folder of REPORTS_LOCAL_FOLDER.

- RESOURCES_REMOTE_FOLDER: Resource folder of the iHub server. The default folder is /Resources. Do not change the name of the default folder.
- RESOURCES_LOCAL_FOLDER: Name of the local resource folder. The default folder is ../OTDctmReportTemplates/Resources.



Note: If you have followed the [step 2](#) and [step 3](#) steps as is, then you need not change the default folder of RESOURCES_LOCAL_FOLDER.

3. Run run.bat or run.sh and select the appropriate options to either version the reports or to overwrite.

If any of the files you are trying to publish already exists in the iHub server in the same folder structure, then a prompt appears to select one action from the following options:

```
<FILE> already exists in iHub.  
Select the options below on what you want to do with file?  
Replace the file (r)  
Replace all files (R)  
Do not replace, but create a new version of file (v)  
Do not replace, but create a new version for all the remaining files (V)  
Skip the file (s)  
Skip all files (S)
```

If publishing is successful, then the following message appears:

```
INFO: Completed the operation of uploading all the files
```

8.4.11.7 Configuring iHub to run reports

This section describes how to configure iHub for providing required permissions to OpenText Documentum CM users to run reports.

To open the iHub portal page from Documentum Administrator, you must map the iHub user with the repository user. A mapping file `dctm_ihub_user_map.json`, is provided to map the iHub user with the repository user. In this file, by default, administrator user is mapped as the default iHub user. If you do not want to map the administrator user or do not want to have any default user, then you must edit or remove this default user entry in the configuration and then have a separate repository user to iHub user mapping.

Here is the code snippet of the `dctm_ihub_user_map.json` file:

```
{  
  "DEFAULT_VOLUME": {  
    "name": "Default Volume",  
    "users": {  
      "DEFAULT_USER": {  
        "name": "Administrator",  
        "password": ""  
      },  
      "dmadmin": {  
        "name": "dmadmin",  
        "password": ""  
      }  
    }  
}
```

```

    }
}
}
```

- **VOLUME:** By default, the DVR publisher publishes the OOTB reports into Default Volume and this value need not be changed.
- **DEFAULT_USER:** User configured here is the default iHub user to log in to the iHub server if there is no 1-1 mapping available for the repository and the iHub user. If you do not want to have any user as the default user then remove this setting. The default iHub user is Administrator and can be changed.

If you want to have a separate user mapping between OpenText Documentum CM and iHub user, then add a new entry (in bold) as in the following example:

```
{
  "DEFAULT_VOLUME": {
    "name": "Default Volume",
    "users": {
      "DEFAULT_USER": {
        "name": "Administrator",
        "password": ""
      },
      "dmadmin": {
        "name": "dmadmin",
        "password": ""
      },
      "dctm_testuser

```

where OpenText Documentum CM user is `dctm_testuser` and iHub user is `ihub_testuser` with password value as `testpassword`.

To create new iHub users and map with OpenText Documentum CM user:

1. Log in to iHub as iHub administrator using the following URL in a supported browser:
`http://<IP Address or name of the host>:8700/iportal`
By default, administrator user is set with a blank (empty) password. Login as an administrator user.
2. Click **iHub Administration** in **Administrator** to open the iHub administration page.
3. In the administration page, create a corresponding iHub user for each repository user you wish to map and also to run the reports using Documentum Administrator. Click **+Add User** and provide all the details.
An iHub user is created. Map this iHub user in the `dctm_ihub_user_map.json` configuration file.
4. Open the iHub portal page.

5. Position the pointer over ellipsis (...) beside the DVR folder and click Share.
6. Select the iHub user who needs the permissions to run the reports and then select the **Visible**, **Execute**, **Grant**, **Secure Read**, **Write**, **Read**, **Delete**, and **All** options. In addition, select the **Apply these privilege settings to the contents of the folder**, **Recursively include subfolders and their contents**, and **Replace existing privilege settings** options.

8.4.11.7.1 Setting up iHub for OpenText Documentum CM authentication and to use JDBC driver

1. Modify the Web.xml file in *<iHub Home>\modules\BIRTiHub\iHub\web\birtservice\WEB-INF* to have the following entry:

```
<context-param>
<!--
Enable External Authentication by setting the Security Adapter class name.
SECURITY_MANAGER_CLASS is deprecated.
-->
<param-name>SECURITY_ADAPTER_CLASS</param-name>
<param-value>com.documentum.dvr.ipse.DvrIPSE</param-value>
</context-param>
```

2. In the iHub server, copy all the files from Dvr_iHubApp\dmjdbc_driver to *<iHub Home>\modules\BIRTiHub\iHub\web\birtservice\WEB-INF\platform\plugins\org.eclipse.birt.report.data.oda.jdbc_<version>\drivers*.
3. The repository against which the reports has to be run, the corresponding connection broker has to be configured on the iHub server. Modify the dfc.properties file in *<iHub Home>\modules\BIRTiHub\iHub\web\birtservice\WEB-INF\platform\plugins\org.eclipse.birt.report.data.oda.jdbc_<version>\drivers* to point to the appropriate connection broker.
4. Copy the following JAR files and DVR_IPSE.jar from Dvr_iHubApp\lib and Dvr_iHubApp\dmjdbc_driver to *<iHub Home>\modules\BIRTiHub\iHub\web\birtservice\WEB-INF\lib* of the iHub server:
 - aspecjrt
 - com.actuate.iportal
 - dfc
 - gson-2.3
 - javaserver
 - org.eclipse.birt.runtime-4.5.0a
5. Copy the dfc.properties and dctm_ihub_user_map.json files from *<iHub Home>\modules\BIRTiHub\iHub\web\birtservice\WEB-INF\platform\plugins\org.eclipse.birt.report.data.oda.jdbc_<version>\drivers* to *<iHub Home>\modules\BIRTiHub\iHub\web\birtservice\WEB-INF\classes*.



Note: Make sure that the dfc.properties file in **step 3** and **step 5** are same.

6. Take a backup of login.jsp from <iHub Home>\modules\BIRTiHub\iHub\web\birtservice\iportal\activePortal\private.
7. Copy and replace login.jsp from Dvr_iHubApp\web to <iHub Home>\modules\BIRTiHub\iHub\web\birtservice\iportal\activePortal\private. You can overwrite when prompted.

8.5 Post-deployment tasks

8.5.1 Licensing OpenText Documentum CM

OpenText Documentum CM uses OpenText Directory Services (OTDS) to apply licenses for all the OpenText Documentum CM components. For more information about procuring the license file and configuring OTDS and license, see “[Licensing OpenText Documentum CM](#)” on page 86.

8.5.2 Configuring OpenText Directory Services

OpenText Directory Services (OTDS) is an authentication server that supports standard OAuth 2.0. The OAuth 2.0 authorization framework is the industry-standard protocol for authorization. It enables third-party application to obtain limited access to an HTTP service.

To install and use OTDS, you must first install and configure the following software:

- Documentum CM Server
- Active Directory (AD), preferably, on a separate machine
- OTDS 16.2.1 with otds-admin and otdswn services in running state
- Documentum Administrator

8.5.2.1 Configuring OTDS integration with Documentum CM Server

Perform the following configurations steps in OTDS:

1. Type the following URL to configure properties in OTDS and log in with the default administrator, otds-admin@otds.admin:
`http://<otds_ip>:<otds_port>/otds-admin`
2. Create a partition. You can create partition using AD server details, create user and group filters, monitor protocol, perform AD to OTDS attributes mapping, and so on.
3. Create a resource. Resource represents any ECM server such as Documentum CM Server. In resource, you can configure resource REST (Generic) server URL (`http://<JMS_Host>:<JMS_Port>/dmotdsrest`), administrator credentials of repository, and OTDS to repository attribute mappings.
4. Map the resources to partition in access role. Access role is created automatically when creating a resource. In access role, you need to add the

partition and the users or groups that need to be pushed to Documentum CM Server.

OpenText Directory Services documentation contains detailed information.

8.5.2.2 Configuring OTDS authentication in Documentum CM Server

1. At the command prompt in Documentum CM Server, type `iapi` followed with repository name, user administrator and password.
2. Update the Documentum CM Server configuration object with `app_server_uri` and `app_server_name`.
3. Configure `otds_rest_credential_url` and OTDS certificate in the `otdsauth.properties` file.
4. Save the `otdsauth.properties` file and restart the Java Method Server server.

For more information about configuring OTDS authentication in Documentum CM Server, see *OpenText Documentum Content Management - Server Administration and Configuration Guide* (EDCCS250400-AGD).

8.5.2.3 Configuring OAuth clients on OTDS

1. Type the following URL to configure OAuth clients on OTDS and log in with the default administrator, `otadmin@otds.admin`:
`http://<otds_ip>:<otds_port>/otds-admin`
2. On OTDS, navigate to OAuth clients and click **Add**.
3. Type the Client ID.
4. Select Redirect URLs, add the Documentum Administrator URL and save your changes.

8.5.2.4 Configuring runtime properties

These authentication modes are supported in REST:

- `otds_token`: Supports OTDS integration with Documentum CM Server using token-based authentication.
- `otds_password`: Supports OTDS integration with Documentum CM Server using password-based authentication.

8.5.2.5 Configuring Documentum Administrator

1. In wdk/app.xml of Documentum Administrator, set OTDS SSO to true as follows:

```
<otds_sso>
    <enabled>true</enabled>
    <!-- <plainpassword>false</plainpassword> -->
    <repo_selection_page_required>true</repo_selection_page_required>
    <!-- renew otds token after the http session is invalidated-->
    <renew_token_after_logout>false</renew_token_after_logout>
    <!-- Enable login fall back to DocbaseLogin scheme -->
    <docbase_login_fallback>false</docbase_login_fallback>
</otds_sso>
```

2. You can set two types of authentication, either OAuth or plain password. If you are using OAuth authentication, set plainpassword to false in the app.xml file of Documentum Administrator.
3. In Documentum Administrator, provide the URL, Client ID, and Client secret used while creating the OAuth client in OTDS in the otds.oauth.properties file.

```
otds_url=<OTDS authentication URL>
client_id=<client ID configured in OTDS for Documentum Administrator>
scheme=<request scheme to support proxy conversion of https to http>
scope=<scope configuration support for tenancy 2.0>
```

8.5.3 Starting Documentum Administrator

Before you test the deployment, make sure that Documentum Administrator is started in the application server. The documentation on each web application server contains information on starting the application.

To verify Documentum Administrator deployment and configuration:

1. Open a browser window and type the following URL:

```
http://<host_name>:<port_number>/<virtual_folder>
```

where:

- <host_name> is the host where the application server is installed. If the browser is on the application server machine, substitute localhost for host_name. For example, `http://localhost`.
- <port_number> is the port where the application server listens for connections.
- <virtual_folder> is the virtual folder for your application.

For example, if the application server host is named `iris`, the port number is 8080, and the application virtual folder is `da`, then the URL is `http://iris:8080/da`.

2. Use Documentum Administrator application to log in to a repository.
If the login succeeds, the application is correctly deployed and configured.

To access the Documentum Administrator URL in an IPv6 environment, you must provide the IPv6 address within the square brackets in the URL.

8.5.4 Maintenance and procedures

After the installation, it is essential to follow a maintenance or procedure checklist for maximum system performance and stability.

Many of the maintenance procedures and jobs are configured or accessed through Documentum Administrator:

- Server and Repository configurations
- LDAP configuration
- Users, Groups, Roles
- Security (ACLs)
- Storage (Locations, Storage, and Filestores)
- Failed index list of Index Agent should be understood and resubmitted, if necessary

8.5.4.1 Logs to monitor

It is highly recommended to check all logs periodically for errors and warnings.

8.5.4.1.1 Application Server

- Name: stdout_yyyymmdd.log. For example, stdout_20090218.log.
- Location: Application Server logs folder.
- Purpose: Warnings and errors from Documentum Administrator and TBOs.

8.5.4.1.2 Documentum CM Server repository

- Name: DocbaseName.log
- Location: \$DOCUMENTUM\dba\log
- Purpose: Repository startup output and any warnings or errors

8.5.4.1.3 Java Method Server

- Name: ServerApps.log
 - Location: %DM_JMS_Home%\logs
 - Purpose: Access and status of the Java Method Server
- Name: DmMethodServerService.log
 - Location: %DM_JMS_HOME%\server\serviceConfig\MethodServer\ DmMethodServerService.log
 - Purpose: Access and status of the Java Method Server

8.5.4.1.4 Index Server



Note: This is applicable only for xPlore 20.2 and earlier versions.

- Name: access.log and DctmServer_IndexAgent.log
- Location: <WildFly_Home>\domains\DtcmDomain\servers\DtcmServer_IndexAgent\logs
- Purpose: Access and status of index agent

8.5.4.2 Disk space management

The Documentum CM Server has a state of the repository job, dm_StateOfDocbase, to monitor the disk space management. In addition, the data drive should be monitored.

Monitor the following:

- SQL Server transaction log
- Webtop cache files
- Index data drive
- Database maintenance and logs
- Disk space
- Transaction logs
- CPU and RAM usage patterns

8.5.4.3 Jobs

Some of the jobs discussed in this section are not active OOTB. They have to be set to active and started on a schedule. Make sure to set the run times so that they do not conflict other jobs and backup schedules.

- dm_ContentWarning: Provides warnings for low availability on DM content/fulltext disk devices.
- dm_LogPurge: Removes outdated server/session, and job/method logs method.
- dm_StateOfDocbase: Lists the repository configuration and status information. In addition, displays the number of documents and total size of content.
- dm_AuditMgt: Removes old audit trail entries A key parameter is the cutoff in days, basically how many days worth of audits to keep.
- dm_QueueMgt: Deletes dequeued items from dm_queue.
- dm_UpdateStats: Updates RDBMS statistics and reorganizes tables, if RDBMS supports.
- dm_ConsistencyChecker: Checks the consistency and integrity of objects in the repository.
- dm_DataDictionaryPublisher: Publishes the data dictionary information.
- dm_LDAPSynchronization: Used for one-way synchronization of LDAP users and groups to Docbase Method through OTDS.
- dm_FTStateOfIndex: State of Index dm_FTIndexAgentBoot Boot Index Agents Method.
- dm_GwmTask_Alert: Sends email alert if task duration exceeds.
- dm_GwmClean: Cleans all the orphan decision objects.

8.5.4.4 DQL queries

This section discusses the DQL queries to be run to check on audit trails and dm_i_queue_items.

The following statements are some of the DQLs to determine the number of audit trails and queue items that were in the repository:

```
Select count(*) from dm_i_queue_item  
Select count(*) from dm_audittrail
```

8.5.4.5 Network connectivity interruption

If any network interruption occurs, then service logs should be checked for compromised activity. The Documentum CM Server and Tomcat server may need to be restarted. The logs of the application and Documentum CM Servers should be periodically monitored for errors and warnings.

8.5.4.6 RAM and CPU Utilization maxed out

If RAM is filled or CPU utilization is maxed-out then the service responsible should be checked. If the service is a OpenText Documentum CM service, it should be restarted and root cause should be determined. Utilization should be monitored and any anticipated spikes in use or additional services need to be load tested and analyzed. If the application server's performance is slow and the concurrent users reach the set limit of 20, it is recommended to add a second application server.

8.5.4.7 Sessions to monitor

This section discusses the different ways to monitor sessions.

- Documentum Administrator: **Administration > User Management > Session**
- DQL:
 - execute show_sessions: To display all active and inactive sessions
 - execute list_sessions: To display active sessions
- DocBasic ebs script: Set this script at a command line prompt to output how many active and inactive sessions are current on the Documentum CM Server. Set the interval between output and how many loops to run.

8.5.4.8 Security and server access maintenance

You can perform the following for the security and server access maintenance:

- Test users and test content should be deleted out of production.
- The database schema owner account should be locked down.
- The OpenText Documentum CM install owner `dmadmin` should be locked down.
- Only scheduled, authorized access to the production should be allowed for all servers of the system.
- Repository audit trails should be configured for certain events, such as deleting of content.

8.5.5 Improving performance

This section lists the guidelines that can significantly improve performance of your web application.

- Improving search query performance
Set <showfolderpath> to false in the search component to speed queries.
- Disable tracing
Turn off tracing to improve performance. Navigate to the wdk/tracing.jsp page and deselect all tracing flags.
- Set the value of dfc.diagnostics.resources.enable to false in the dfc.properties file unless you are using the Foundation Java API diagnostics. This setting uses a significant amount of memory.

8.5.5.1 Java Enterprise Edition memory allocation

If the memory allocated to the Java Enterprise Edition (EE) server Java virtual machine (JVM) is not correctly set, the JVM spends a lot of time destroying Java objects, garbage collecting, and creating the new objects. To change the memory allocation, use a setting similar to the following in the Java arguments in the Java EE server start script that you use to start your application server:

```
-Xms512m -Xmx512m -verbose:gc
```

Element	Description
-Xms512m	Starting memory heap size, in megabytes. In general, increased heap size increases performance up until the point at which the heap begins swapping to disk.
-Xmx512m	Maximum Heap size. For a single JVM, Sun recommends that you set maximum heap size to 25% of total physical memory on the server host to avoid disk swapping. Increased heap size will increase the intervals between garbage collection (GC), which thus increases the pause time for GC.
-verbose:gc	Turns on output of garbage collection trace to standard output. Increased Java memory settings increases the amount of time before a major garbage collection takes and also increases the amount of time that garbage collection takes to execute. Garbage collection is the greatest bottleneck in the application, and all application work pauses during garbage collection.

Garbage collection tracing has the following syntax:

```
[GC 776527K->544591K(1040384K), 0.4283872 secs]
```

The trace can be interpreted as follows:

Element	Description
GC	GC indicates minor garbage collection event and Full GC indicates full garbage collection
776527K	Amount of total allocated memory at start of minor collection
544591K	Amount of total allocated memory at end of minor collection
1040384K	Amount of total memory on host
0.4283872 secs	Time in seconds to run garbage collection

Monitor memory usage by the Java process in the Windows task manager to determine whether your memory allocations are optimum. Allocated memory as shown in consecutive GC traces continues to grow until full garbage collection occurs. Full garbage collection takes much longer than minor garbage collection, often on the order of 10 times as long.

The following table describes some memory troubleshooting inferences that can be drawn from garbage collection:

Symptom	Reason
Frequent full GC, starting point higher after each full GC, decreasing number of GC between full GC	Total memory too small, or memory leak
Garbage collections take too long (GC 1 second, full GC 5 seconds), server cannot create the new threads	Too much memory allocated to JVM

Java EE servers also have configurable settings for thread management which can significantly affect performance. The symptom of insufficient threads is that, as the number of users increases, performance degrades without increased CPU usage. Some users get socket errors. In Tomcat, the catalina.log file shows that all threads up to maxProcessors have been started, and the new requests are rejected with Connection Reset By Peer.

The symptom of too many threads is excessive context switching between live threads and degraded response time.

Your application server documentation contains more information on these settings.

8.5.5.2 Preferences

User preferences are stored as cookies and written to the repository. Since cookies are passed back and forth with every request and response, there is a small increase in network traffic.

The configuration lookup methods `lookupString`, `lookupInteger`, and `lookupBoolean` have an optional parameter `consultPreference`. Set to `false` to look up a configuration value from the component definition and bypass a lookup of the user preference when the lookup is not needed.

8.5.5.3 Browser history

The number of history pages maintained on the server for each window or frame is set by the `<requestHistorySize>` flag in the `FormProcessorProp.properties` file, which is located in `WEB-INF/classes/com/documentum/web/form`. The default value is 3. If the value is empty or zero, then history is maintained indefinitely. This setting could significantly affect performance. Decrease the memory footprint per user by setting this to a lower value. If you set it higher, it consumes more memory.

Too many form history objects can use up memory. Set the upper limit for the number of objects as the value of `<maxNoOfFormHistoriesThreshold>` in `FormProcessorProp.properties`. The default value is 50. A message is displayed if the user tries to navigate past the maximum number of pages in history.

Memory that is allocated to maintaining browser history is managed more efficiently on the Java EE server if you generate framesets and frames using the `<dmf:frameset>` and `<dmf:frame>` tags. For more information about the browser history, see *OpenText Documentum Web Development Kit - Development Guide (EDCPKCLWT160709-PGD)*.

8.5.5.4 Value assistance

Performance is affected by the number of value assistance queries to be displayed in the properties component and in other components that display a set of properties. Do the following to enhance this performance:

- For each value assistance query, use Documentum Composer to turn on the option to allow caching.
- Turn on client persistent caching in the `dfc.properties` file located in `WEB-INF/classes` as follows:

```
dfc.cache.enable_persistence = T
```
- Index the associated attributes in Documentum CM Server.

8.5.5.5 Search query performance

Set `<displayresultspath>` to `false` in your custom search component definition to speed all queries. This suppresses the query for folder path of each object.

In advanced search, you can add a check box for case-sensitive search for non-indexed repositories. Set the `<casevisible>` attribute on the search controls to `true`. Set the default match case as the value of the element `<defaultmatchcase>` (`true` | `false`) in `wdk/config/advsearchex.xml`. Case-sensitive queries perform faster.

8.5.5.6 High latency and low bandwidth connections

Two filters are available to improve performance in high latency or low bandwidth networks. The filters are defined as servlet filters in `WEB-INF/web.xml`. They are turned on by default. The filters are as follows:

- Response compression filter (CompressionFilter)

Compresses text responses by mapping requests for `*.jsp`, `*.css`, `*.js`, `*.htm`, `*.html`, and the component dispatcher servlet. If the request `Accept-` header indicates that the browser accepts compression, the filter swaps the output stream for a compressed stream in either GZIP or deflate compression formats, depending on which format is accepted by the browser as indicated by the `Accept-` request header.

The configurable value for this filter, `init-param compressThreshold`, is a size in KB or MB that sets the threshold file size at which output is compressed. Compression does not decrease the size of the stream for small inputs. In addition, high-bandwidth networks may show improvement for only very large files (hundreds of KB). A value of `3KB` indicates that files 3 KB or larger is compressed.

Additionally, there are `init-params` for turning on compression filter debugging and excluding specific JSP pages from compression filtering.

A known limitation is that there is an unknown CPU cost for the compression.

- Cache control (ClientCacheControl)

Limits the number of requests by telling the client browser and any intermediary caches such as caching proxies to cache static elements such as `*.gif`, `*.js`, and `*.css` files, by adding a `Cache-Control` response header. After the browser has received a response with this header, it will not reget the content until the maximum age or until the content is cleared manually from the browser cache.

The configurable value for this filter, `init-param Cache-Control`, is the maximum age in seconds of the static content before revalidation. For example, `max-age=86400` (one day).

Add URL patterns to specify the file types that will be cached. In the following example, `*.gif` files are cached for up to two days:

```
<filter>
<filter-name>ClientCacheControl</filter-name>
<filter-class>com...ResponseHeaderControlFilter</filter-class>
<init-param>
```

```
<param-name>Cache-Control</param-name>
<param-value>max-age=172800</param-value>
</init-param>
</filter>
</filter>
<filter-mapping>
<filter-name>ClientCacheControl</filter-name>
<url-pattern>*.gif</url-pattern>
</filter-mapping>
```



Note: Safari browser does not apply this header.

Tracing for these filters can be enabled through the standard tracing mechanism (TraceProp.properties) or by adding the debug *<init-param>* element to the application deployment descriptor (WEB-INF/web.xml).

➡ Example 8-1: Enabling tracing of filters in WEB-INF/web.xml

```
<filter>
<filter-name>CompressionFilter</filter-name>
<filter-class>com.documentum.web.servlet.CompressionFilter</filter-class>
<init-param>
<param-name>compressThreshold</param-name>
<param-value>3kb</param-value>
</init-param>
<init-param>
<param-name>debug</param-name>
<param-value>true</param-value>
</init-param>
</filter>
```



8.5.5.7 Qualifiers and performance

Each qualifier that is defined in the application slows performance the first time a component is called. Navigation components must evaluate qualifiers for each action in the component JSP page. To improve performance, remove from your custom app.xml file the qualifiers that your application does not need. The application qualifier is required. In the following example from an app.xml file in the custom folder, only the type qualifier is used by a custom application. The app qualifier is required for all applications. No components or actions can be scoped to role in this example, because the role qualifier is not defined for the application.

```
<qualifiers>
<qualifier>com.documentum.web.formext.config.DocbaseTypeQualifier
</qualifier>
<qualifier>com.documentum.web.formext.config.AppQualifier
</qualifier>
</qualifiers>
```

For better performance, your qualifier should implement the IIInquisitiveQualifier interface. At startup, this interface is used to inform the qualifier of all relevant scopes defined in the action and component definitions. The qualifier can return an empty scope value that is cached, when the runtime context is not relevant.

8.5.5.8 Import performance

You can limit the number of files that can be imported by a user during a single import operation. This configuration setting is the `max-import-file-count` element with a default of 1000 in the `importcontainer` component. Extend this component definition to configure a different maximum value.

Certain environments have forward or reverse proxy web servers that do not support HTTP 1.1 chunking, which is used by UCF for content transfer. For those environments, you must configure UCF to use alternative chunking, and you can tune the chunk size for the web server. In general, the default chunk size works best for large file transfers. Smaller chunk sizes may enhance performance for small (less than 1 MB) files but degrade performance for large files. For more information about import performance, see *OpenText Documentum Web Development Kit - Development Guide (EDCPKCLWT160709-PGD)*.

8.5.5.9 Modal windows and performance

Modal windows provide a performance enhancement in web applications that use several frames. With a modal window, other frames do not need to refresh after the modal frame closes. For more information about modal windows and performance, see *OpenText Documentum Web Development Kit - Development Guide (EDCPKCLWT160709-PGD)*.

8.6 Troubleshooting

8.6.1 Wrong JRE used for application server

If the application server host has multiple JREs on the system, the application server can use the wrong JRE. Check your application server documentation for instructions to use the correct JRE with your application server. For example, the Apache Tomcat application server uses the `JAVA_HOME` environment variable. This variable value is specified in the application startup batch file `catalina.bat` or in the `service.bat` file for Windows services.

If the application server uses the wrong JRE, Apache Tomcat displays the following error:

```
ERROR [Thread-1]
org.apache.catalina.core.ContainerBase.[Catalina].[localhost].[/da]
- Error configuring application listener of
  classcom.documentum.web.env.NotificationManager
java.lang.UnsupportedClassVersionError:
  com/documentum/web/env/NotificationManager(Unsupported major.minor version 49.0)at
  java.lang.ClassLoader.defineClass0(Native Method)
```

8.6.2 No global registry or connection broker

Global registry information must be configured in the `dfc.properties` file. The application server must be able to download the required BOF modules from the global registry repository. If the information in the `dfc.properties` file is incorrect, the application server cannot download the appropriate BOF modules, and the following exception is thrown:

```
ERROR...Caused by:  
DfDocbrokerException:: THREAD: main; MSG:  
[DFC_DOCBROKER_REQUEST_FAILED] Request to Docbroker "10.8.3.21:1489" failed;  
ERRORCODE: ff; NEXT: null
```

To resolve this error, provide the correct BOF registry connection information in the `dfc.properties` file or do not provide any connection information. Make sure that you enable a repository as a global registry.

8.6.3 No connection to repository

If a connection broker is not specified in the `dfc.properties` file of the Documentum Administrator WAR file, the application server log contains the following error during application initialization:

```
at org.apache.catalina.startup.Bootstrap.main(Bootstrap.java:432)  
Caused by: DfDocbrokerException:: THREAD: main; MSG: [DFC_DOCBROKER_REQUEST_FAIL  
ED] Request to Docbroker "10.8.3.21:1489" failed; ERRORCODE: ff; NEXT: null
```

To establish a connection to repositories, Documentum Administrator must have information about the available connection broker. Enable the connection in the `dfc.properties` file.

If the repository that is specified as the global repository is down, the following message appears:

```
Caused by: DfNoServersException:: THREAD: main;  
MSG:[DM_DOCBROKER_E_NO_SERVERS_FOR_DOCBASE]error:  
"The DocBroker running on host (10.8.3.21:1489)  
does not know of a server for the specified docbase  
(wtD6winsql)"; ERRORCODE: 100; NEXT: null
```

8.6.4 Login page incorrectly displayed

If the login page displays several login buttons, the browser does not have the Java plug-in installed. Download and install the Java plug-in for the browser.

If the login page displays several controls with the same label, you have not turned off tag pooling in the application server.

8.6.5 Slow performance

For more information, see “[Improving performance](#)” on page 446.

8.6.6 Slow display first time

The application server must compile a JSP the first time it is accessed. It is much faster on subsequent accesses. If you have tracing turned on, or if you have a large log file (of several megabytes), the browser response time decreases dramatically.

8.6.7 Foundation Java API using the wrong directories on the application server

If you have not specified content transfer directories in the `dfc.properties` file, Foundation Java API looks first for global environment variables that set folder locations.

8.6.8 Tag pooling problem

If you have not properly disabled tag pooling in the application server, you see several instances of the same control on the login page.



Note: After you disable tag pooling, clear the cached JSP class files which can still contain pooled tags. Refer to your application server documentation to find the location of the generated class files. For example, Apache Tomcat displays the following error message:`com.documentum.web.form.control.TagPoolingEnabledException:JSP tag pooling is not supported.`

8.6.9 UCF client problems

If the `Compatible Java Run time environment is not installed` error message is displayed on a non-Windows client, verify that you have installed a certified version of the JRE on the client. UCF uses this version, which does not interfere with the browser VM. It is used for non-UCF applets.

If a UCF error is reported on the client, the following troubleshooting steps can help:

- For UCF timeouts, check whether anti-virus software on the application server is monitoring port 8080 or the application server port that is in use. Turn off monitoring of the application server port.
- For slow UCF downloads, make sure that virus scanning within ZIP files is not turned on.
- Make sure that the user has a supported JRE version on the machine to initiate UCF installation. To verify the presence and version of a JRE, you can point the client browser to a Java tester utility.
- Verify if the process from the launch command is running: Open the browser Java console look for invoked runtime: ... connected, uid:. A UID indicates successful connection to the UCF server.

- Check the application server console for errors on the UCF server.
- Restart the browser and retry the content transfer operation.
- Terminate the UCF launch process and retry the content transfer operation.
- If UCF operations still do not launch, delete the client UCF folder located in `USER_HOME/username/Documentum/ucf`.
- Search the client system for files that start with `ucfinit.jar-` and delete them.

8.6.10 Connection issues between a Federated Search Services server and IPv6 clients

Federated Search Services server uses the RMI protocol to communicate with the client applications. When the client application launches a request against the Federated Search Services server, it indicates the IP address that the Federated Search Services server must use to respond. However, it can happen that the client sends a link-local address instead of a global address. To avoid any connection issue, update the `catalina.bat` script that launches Documentum Administrator. The following setting forces the RMI IP to connect:

```
set JAVA_OPTS=%JAVA_OPTS% -Djava.rmi.server.hostname=
<global IPv6 address>
```

8.6.11 Max Sessions error

Before restarting the application server, use the Documentum Administrator to find the current active and inactive users sessions in the repository. Try reducing the session timeout value in the application server to see if the inactive sessions get cleared out faster.

Chapter 9

Thumbnail Server

This documentation describes the product and how it works to serve thumbnails to the clients.

Thumbnail Server retrieves low-resolution image files from a designated thumbnail file store on the Documentum CM Server host. These images are then used as visual cues in browsers and other web client applications.

Two types of thumbnails are served by Thumbnail Server:

- Preview images created by Transformation Services server when an applicable file type is imported or checked in to the repository. These thumbnails are saved in the thumbnail store on the Server.
- Default thumbnail images per type or format. These are icons that can be specified for folders or unknown file types. These thumbnails are stored in the repository, and can be viewed or updated when necessary.

Thumbnail Server uses Java servlets to manage thumbnail representations, and HTTP technology to accelerate the display of thumbnail images in web client applications. The file store is designated as a thumbnail file store when its *media_type* attribute is set to 1.

Applications retrieve thumbnails from thumbnail file stores by constructing a URL for the file. The URL consists of a main servlet which requires a path, store, and ticket argument, if applicable.

9.1 Introduction

All requests for the system to return thumbnails are processed by Thumbnail Server. The following sections describe how Thumbnail Server works when a thumbnail is requested from an application.

9.1.1 Thumbnails and Thumbnail Server

A thumbnail is an image that is used to visually represent a file in client applications. Thumbnail renditions provide a visual cue for browsing files, and they enable users to quickly identify objects at a glance, since the object name and/or additional attributes will not always provide enough information about an object.

After the registration of an object, the Transformation Services server automatically generates thumbnail renditions of the object by passing it to the appropriate plug-in. The plug-in extracts the object's media properties and creates a new object by transforming the original object into a predefined thumbnail format. The Transformation Services server sends the thumbnail back to Documentum CM

Server as a rendition of the original object and saves the thumbnail's properties as attributes of that rendition. Documentum CM Server stores thumbnails in a special file store that is shared with Thumbnail Server.

Thumbnail Server is a separate server (which must be downloaded and installed on the Documentum CM Server host) that interacts directly with a client browser. Thumbnail Server runs on a servlet engine, which itself runs on an HTTP server. This means that Java servlets are used to manage thumbnail representations and HTTP or HTTPS protocol is used to communicate with the web client applications.

For configurations that require SSL security, a special configuration is required to run Thumbnail Server with an HTTPS protocol. ["Configuring Thumbnail Server for SSL" on page 469](#) provides more information to configure Thumbnail Server for SSL.

9.1.2 Providing security for thumbnail requests

To make sure that thumbnail requests always come from a properly logged in client, Thumbnail Server provides security for all thumbnail requests, if you select.

There are two methods of providing security for thumbnail requests:

- Set security at the repository level. Thumbnail security is enabled for each repository by using the *dm_store.require_ticket* attribute of the thumbnail store object.
- Set security at the object level. If you decide not to use require_ticket security, then the standard ACL permissions security is applied to thumbnails.

It is highly recommended to turn on the require ticket feature of Thumbnail Server to disable thumbnail URL manipulation by the requesting clients.

Details for each method are provided in the following sections.

9.1.2.1 Using *require_ticket* attribute to provide security

Each repository includes a *dm_store.require_ticket* attribute for the thumbnail store object. This attribute can be used to provide security in addition to the security provided by the use of ACLs (as described in ["Using ACLs to provide security" on page 457](#)).

If the *require_ticket* attribute is set to TRUE for a storage area, Documentum CM Server generates a URL with an encrypted ticket that is valid for a period of five minutes. A ticket contains an encrypted path to the thumbnail and a timestamp. If the attribute flag is set to FALSE, Documentum CM Server generates the thumbnail URLs without a ticket.

Ticket encryption is based on a key generated by the genkey script. The script is provided with Thumbnail Server and is run during installation to generate a default key. The key is stored in this file:

```
%DM_HOME%\bin\thumbsrv.key
```

Thumbnail Server and Documentum CM Server share the same secret key.

When Thumbnail Server receives a URL containing a ticket, the server decrypts the ticket and compares the decrypted value to the URL, and compares the timestamp in the ticket to the current time. If the 5-minute limit has not expired, the server decrypts the path to the thumbnail. It then parses and processes the URL. The expiration time limit can be changed from the default of five minutes. [“Changing the ticket time-out value” on page 477](#) provides instructions to change the ticket time-out value. [“Requesting thumbnails” on page 458](#) describes how Thumbnail Server handles URLs.

Storage areas from which Thumbnail Server requests thumbnails can be components of distributed storage areas. In this case, the Documentum CM Server uses a different attribute to provide security. The server uses the value of the *require_ticket* attribute of the distributed storage area object, `dm_distributed_store`, rather than the value of the attribute for the component area.

9.1.2.2 Using ACLs to provide security

If ticket security is disabled for a repository, security for thumbnail renditions can be enforced using the ACL permissions on the object with which the thumbnail is associated. [“Using require_ticket attribute to provide security” on page 456](#) provides more information.

Thumbnails can be requested in two ways: SELECT statements and GET_FILE_URL administrative methods. The application of the permissions defined in ACLs to thumbnails depends on how the thumbnail is requested.

When a SELECT statement is used to request a thumbnail, the user must have at least *Browse* permission on the object of which the thumbnail is a rendition. The Documentum CM Server checks the permissions on the object when the SELECT statement is issued. If the user has the appropriate permission, the server returns the URL for the thumbnail and any requested attribute values.

When a GET_FILE_URL administrative method is used to request a thumbnail, the user must have either *Read* permission on the object of which the thumbnail is a rendition, or *Sysadmin* privileges. Otherwise, the Documentum CM Server does not return the URL for the thumbnail.

9.1.3 Requesting thumbnails

When an application requests a thumbnail file, it sends the request to the Documentum CM Server which sends the thumbnail URL back to the application. The application uses the thumbnail URL in a browser page. When the page is about to display, the client (browser) contacts Thumbnail Server (using the same URL) to obtain the desired thumbnail rendition.

Thumbnail Server consists of a main servlet which requires a path, store, and ticket argument, if applicable, in the URL.

- The path argument is the path of the thumbnail on the Documentum CM Server, relative to the root location of the thumbnail storage area. For example, 00232803/80/00/01/0b.jpg.
- The store argument is the name of the storage area where the file is stored, or the distributed store name if stored in a distributed store. For example, the store can be called thumbnail_store_01.
- If ticket security has been enabled, the third argument is a ticket argument, which is a string that encrypts the preceding parameters and a timestamp. For example, the ticket will be 76WR4QJ89X102.

With the preceding examples, a URL to request a thumbnail will appear as follows:

```
http://MyDocumentumServer:8081/thumbsrv/getThumbnail?  
path=00232803\80\00\01\0b.jpg&store=thumbnail_store_01&  
ticket=76WR4QJ89X102&format=jpeg_th&page_modifier=medium_jpeg_th&  
did=090004d280001c70
```

When the servlet is invoked, it extracts the required URL arguments. If the URL contains a ticket, Thumbnail Server decrypts it and matches it against the first two arguments of the URL (for example, path=00232803/80/00/01/0b.jpg&.store=thumbnail_store_01&ticket=76WR4QJ89X102). If it matches, the timestamp is checked. If the timestamp is less than the current time, then the request is rejected. This means that the five minute time limit has expired. “[Changing the ticket time-out value](#)” on page 477 contains instructions for changing the default ticket time-out.

After the URL has passed the security check, the servlet determines the repository where the thumbnail is stored. This is determined by the path argument which is the repository ID. The first time a request for a particular repository is made, the servlet connects to it and retrieve the storage area with the specified name. Then, it checks whether it is really a thumbnail storage area (attribute media_type=1). If not, an error is returned. Then, it checks if the storage area requires a ticket. If it does, and no ticket has been provided in the URL, the request is rejected. Otherwise, the root path of the storage area is retrieved and cached for subsequent requests. The complete path to the thumbnail is then constructed by concatenating the two paths. Finally, the thumbnail file is returned to the browser.

If the request fails any of the tests detailed in the preceding case, an HTTP 400 error (Bad Request) is returned to the client's browser. However, a detailed error message is written to the Thumbnail Server log file, localhost.xxxx-xx-xx.log.

If either the path or store argument is missing from the URL received by the server, the server uses the arguments that are present to select a default thumbnail to return to the application. “[Serving default thumbnails](#)” on page 459 describes how default thumbnails are served to the browser.

9.1.4 Serving default thumbnails

A default thumbnail is a thumbnail file used as a substitute for the requested thumbnail file, for documents that do not have an associated preview image. The thumbnail to be served can be based on three characteristics of a document: object type, format, and whether it is a virtual document. A default thumbnail usually appears as a generic icon representing the object's file type.

Sometimes an application requests a URL to a non-existent thumbnail rendition of an object. For example, when you request a thumbnail for an object that was just deleted by another user, or if the object was just registered, and OpenText™ Documentum™ Content Management Transformation Services - Media server or OpenText™ Documentum™ Content Management Transformation Services - Documents server has not yet had a chance to generate a thumbnail. When that occurs, the application can use the information returned by Documentum CM Server (by the `THUMBNAIL_URL` keyword or the `GET_FILE_URL` administrative method, depending on which security method you select) to build a URL that returns a default thumbnail. “[Providing security for thumbnail requests](#)” on page 456 provides more information on security when requesting thumbnails.

Default thumbnails are stored on the repositories configured for Thumbnail Server. Thumbnail Server determines which thumbnail to serve based on rules defined in the `default_thumbnails.xml` file. The `default_thumbnails.xml` file's elements describe a list of rules for matching parameters in a URL with a default thumbnail. The `default_thumbnails` rules file conforms to the `default_thumbnails.dtd`.

“[Understanding default thumbnails](#)” on page 475 provides instructions to obtain a default thumbnail. “[Adding default thumbnails](#)” on page 476 provides instructions to add a default thumbnail. “[Changing the ticket time-out value](#)” on page 477 provides instructions to change the encryption key.

9.2 Installation overview

Thumbnail Server serves graphic files from a thumbnail file store on the Documentum CM Server host. It provides these thumbnails to rich media-enabled client applications. Thumbnails are renditions created by the Transformation Services server when a file is imported or checked in to the repository.

Installing Thumbnail Server installs the Thumbnail Server software and sets the `base_url` attribute for file store objects that are defined for repositories residing on the host thumbnail storage areas.

The `base_url` is set to a URL that identifies Thumbnail Server. Its value is used by applications to construct complete URLs for files in the thumbnail storage area.

A file store is a thumbnail storage area when its *media_type* attribute is set to 1.

On Windows hosts, Thumbnail Server is installed as a service, set to start automatically after the system restart.



Caution

If you are upgrading Documentum CM Server from a release earlier than the latest version, then uninstall the previous version of Thumbnail Server first. Install and configure the latest version of Thumbnail Server after upgrading to the latest version of Documentum CM Server.

9.2.1 Prerequisites

Thumbnail Server must be installed on the same host as your Documentum CM Server. Thumbnail Server requires at least 32.5 MB of disk space.

The product *Release Notes* on My Support (support.opentext.com) provides the information on system requirements.

9.2.1.1 Setup verification

Before beginning the installation, verify the following:

- WinZip or a similar file-extraction utility is installed on the host machine(s).
- If you already have a `dfc.properties` file, make sure that it is pointing to the correct connection broker. *"Identifying a connection broker in dfc.properties"* on page 460 provides the instructions.
- The connection broker and repository services are running on your Documentum CM Server. *"Checking connection broker and repository services"* on page 461 provides more information.
- The Documentum CM Server is installed and a repository has been created.

9.2.1.1.1 Identifying a connection broker in `dfc.properties`

If you already have Foundation Java API installed on your Thumbnail Server host, a `dfc.properties` file already exists. Perform the following steps to make sure that the correct connection broker is identified:



Note: If you are installing Thumbnail Server on a clean host, then a `dfc.properties` file does not exist prior to installation, and therefore this procedure is not required.

1. Search for the `dfc.properties` file on your host. The file is usually located in the `$DOCUMENTUM/config/` folder.
2. Open the `dfc.properties` file in a text editor.

3. Find the line indicating the connection broker. Make sure that the specified connection broker is the one that connects to the repository. Change it, if necessary.
4. Save and close the `dfc.properties` file.

9.2.1.1.2 Checking connection broker and repository services

The connection broker and repository services must be of version 6 or later, and running properly on the Documentum CM Server host before you install Thumbnail Server.

9.2.1.2 Downloading installer

Before beginning the installation process, it is best to have the installer ready and available on your Thumbnail Server host. The Thumbnail Server installer can be found on My Support (support.opentext.com).

9.2.2 Installing, configuring, and uninstalling Thumbnail Server

This section explains how to install, configure, remove configuration, and uninstall the Thumbnail Server:

9.2.2.1 Required Thumbnail Server installation information

This section provides a table for recording the information you need to install Thumbnail Server. Having this information ready and available prior to installation ensures the accuracy and efficiency of your installation.

Installation information	Description
The host computer's Windows domain or the machine name	If you are installing Thumbnail Server on a Windows host, you must know the host's Windows domain. If the host is not part of a Windows domain, you must know the machine name.
Password for the installation owner	This is the same password as the one used to log in to the host.
Thumbnail Server and administration port numbers	These ports are used by Thumbnail Server and its administration tool. The default Thumbnail Server port is 8081. The default administration port is 8008. Leave these ports as default, whenever possible.

Installation information	Description
User names, passwords, and AEK passphrase (if it is not the default passphrase) for repositories you want to configure with Thumbnail Server	For each repository you select to configure with Thumbnail Server, a superuser name and password is required.

9.2.2.2 Installing and configuring Thumbnail Server on Windows

This section gives the instructions to install and configure Thumbnail Server on a Windows host. The installer installs the Thumbnail Server software. All running repositories on the host can be configured by setting the *base_url* for all file store objects that are also thumbnail storage areas. You can rerun the configurator at any time to configure additional repositories.

To configure Thumbnail Server to use HashiCorp Vault, you must store all secrets in the HashiCorp Vault server. For more information about storing secrets in HashiCorp Vault, see *HashiCorp* documentation.

After storing the secrets in HashiCorp Vault, the Thumbnail Server installer updates the secret and key name information in the <secret_name>/<key_name> format.

To install Thumbnail Server on a Windows host:

1. Log in to the Documentum CM Server host as the Documentum CM Server installation owner.
2. Remove any previous versions of Thumbnail Server using the instructions in “[Uninstalling Thumbnail Server from a Windows host](#)” on page 472.
3. Recall where the installation files are located.
4. Extract the Thumbnail Server installer bundle to a temporary location.
5. Double-click the thumbserverSetup.exe file to start the installer.
6. On the welcome page, click **Next**.
7. Accept the license agreement and click **Next**.
8. If you do not have the correct Foundation Java API installed on the Documentum CM Server host, the installer fails at this point.

You are prompted to select if Thumbnail Server must run in the HTTP mode or HTTPS (SSL) mode.

Select the appropriate option and click **Next**.



Note: If you have chosen to run in the HTTPS mode, then you must also configure Thumbnail Server for SSL. “[Configuring Thumbnail Server for SSL](#)” on page 469 provides the instructions.

9. Type the port number that is available for Thumbnail Server or click **Next** to accept the default port (8081).

If you type a port number already in use, you are prompted to type another port number.

10. Type the port number that is available for the Thumbnail Server administration or click **Next** to accept the default port (8008).

The installation summary page appears, indicating the applications that are to be installed.

- To confirm the installation, click **Next**.
- If you want to change a component of the installation, click **Back** to navigate back through the installer and change the installation information.

The installation proceeds, installing Thumbnail Server and its configurator. The installer creates the following folders in the %DM_HOME%\thumbsrv\ directory\:

- \conf\
- \configurator\
- \container\
- \install\
- \logs\

The post-installation notification page appears, informing you to run the Thumbnail Server configurator to add support to serviceable repositories.

11. Make note of the configurator location and click **Next**.
12. Click **Done** to close the installer.

To configure repositories for Thumbnail Server on a Windows host:

1. Navigate to the Thumbnail Server configurator in **Start > Programs > Documentum > Apply Thumbnail Server support to docbase**.
 **Note:** Your connection broker and any repositories requiring Thumbnail Server must be running to use the configurator.
2. If you want to configure the HashiCorp Vault secrets, on the **Documentum Thumbnail Server Vault Components** page, select the **Enable Vault** check box.
3. If you select the **Enable Vault** check box, provide the following information and click **Next**:
 - a. **DSIS URL:** Provide a value in the following format:
`http://localhost:<port mentioned in application.properties>/dsis`
 - b. **DSIS Token:** Provide the `<dsis_token>` token value as described in “Configuring Documentum Secret Integration Service” on page 49.

! **Important**

If you enabled the HashiCorp Vault configuration, the installer retrieves all the password information automatically from the HashiCorp Vault server. Make sure that you have stored the secret and key name information in HashiCorp Vault in the <secret_name>/<key_name> format. For example, INSTALL_OWNER_PASSWORD/testrepo.

4. Click **Next**.
5. Accept the license agreement and click **Next**.
6. The installer searches for all repositories configured for your Documentum CM Server host (in the %DOCUMENTUM%\dba\config\ folder). Select a repository from the list and click **Next**.

You are prompted to type the name of a repository superuser, their password, and the domain (optional). Thumbnail Server uses this information to communicate with the repository.

7. Type the required information and click **Next**.
If the default AEK passphrase is used in the repository configuration, then the Thumbnail Server configurator automatically uses the default AEK passphrase.
If the repository is configured with the custom AEK passphrase, then the Thumbnail Server configurator displays a page for the user to type the AEK passphrase.
8. Type the AEK passphrase.
If the repository you have chosen is part of a Distributed Documentum CM Server environment, a list of server config objects appear for the repository.
9. Select a server config object from the list and click **Next**.
10. Click **Done**. Thumbnail Server is started automatically.

Rerun the configurator for any other repositories you wish to configure for Thumbnail Server.

If you create a new repository on a host where Thumbnail Server is installed, you can run the Thumbnail Server configurator to search for new repositories and configure them automatically.

9.2.2.3 Installing and configuring Thumbnail Server on non-Windows

Use the instructions in this section to install and configure Thumbnail Server on a non-Windows Documentum CM Server and configure it for any repositories served by the connection broker(s) listed in your `dfc.properties` file. For these instructions, when we refer to non-Windows, we also mean the other supported Linux variations. Repositories are configured for Thumbnail Server by setting `base_url` for the file store objects associated with thumbnail storage areas.

You can rerun the Thumbnail Server configurator at a later time to configure additional repositories.

To install Thumbnail Server on non-Windows host:

1. Log in to the Documentum CM Server host as the Documentum CM Server installation owner.
2. Remove any previous versions of Thumbnail Server using the instructions in ["Uninstalling Thumbnail Server from a non-Windows host" on page 472](#).
3. With repository and connection broker running, add the following classpath where `$DOCUMENTUM_SHARED` is the path that contains the `dctm.jar` file and the `config` folder as follows:
`CLASSPATH=$DOCUMENTUM_SHARED/dctm.jar:$DOCUMENTUM_SHARED/config:$CLASSPATH`
4. Recall where the installation files are located.
5. Use the TAR utility to transfer the downloaded file to a temporary directory using the following command applicable for your operating system:
 - For Linux: `% tar -xvf ThumbnailServer_Linux_<version>.tar`

This command creates or extracts the following files:

- `thumbserverLinuxSetup.bin`
 - `thumbserver_Linux.zip`
 - `ts_install.properties`
 - `ts_config.properties`
6. Run `./thumbserverLinuxSetup.bin`.
 7. On the welcome screen, click **Next**.
 8. Accept the license agreement and click **Next**.
 9. You are prompted as to whether you want Thumbnail Server to run in the HTTP mode or HTTPS (SSL) mode. Select the appropriate option and click **Next**.



Note: If you have chosen to run in the HTTPS mode, then you must also configure Thumbnail Server for SSL. ["Configuring Thumbnail Server for](#)

"SSL" on page 469 provides the instructions. The installer log file also contains this information.

10. Type the port number that is available for Thumbnail Server or click **Next** to accept the default port (8081).

If you type a port number already in use, you are prompted to type another port number.

11. Type the port number that is available for the Thumbnail Server administration or click **Next** to accept the default port (8008).

The installation summary screen appears, indicating the applications that are to be installed.

12. Perform one of the following steps:

- To confirm the installation, click **Install**.
- If you want to change a component of the installation, click the **Back** button to navigate back through the installer and change the installation information.

The installation proceeds, installing Thumbnail Server and its configurator. The installer creates the following folders in the \$DM_HOME/thumbsrv/ directory:

- /conf/
- /container/
- /configurator/
- /install/
- /logs/

The post-installation notification screen appears, informing you to run the Thumbnail Server configurator to add support to serviceable repositories.

13. Make note of the configurator location and click **Next**.

14. Click **Done**.

To configure repositories for Thumbnail Server on a non-Windows host:

1. Navigate to the configurator folder identified in step 11 of the previous procedure (usually located in \$DM_HOME/thumbsrv/).
2. Run ./thumbServerLinuxConfigurator.bin.

The Thumbnail Server configurator splash screen is displayed, followed by the Welcome screen.



Note: Your connection broker and any repositories requiring Thumbnail Server must be running to use the configurator.

3. If you want to configure the HashiCorp Vault secrets, on the **Documentum thumbnail Server Components** page, select the **Enable Vault** check box.
4. If you select the **Enable Vault** check box, provide the following information and click **Next**:
 - a. **DSIS URL:** Provide a value in the following format:
`http://localhost:<port mentioned in application.properties>/dsis`
 - b. **DSIS Token:** Provide the `<dsis_token>` token value as described in “Configuring Documentum Secret Integration Service” on page 49.

! Important

If you enabled the HashiCorp Vault configuration, the installer retrieves all the password information automatically from the HashiCorp Vault server. Make sure that you have stored the secret and key name information in HashiCorp Vault in the `<secret_name>/<key_name>` format. For example, `INSTALL_OWNER_PASSWORD/testrepo`.

5. Click **Next**.
6. Accept the license agreement and click **Next**.
7. The installer searches for all repositories configured for your Documentum CM Server host (in the `$DOCUMENTUM/dba/config/` folder). Select a repository from the list and click **Next**.

You are prompted to type the name of a repository superuser, their password, and the domain (optional). Thumbnail Server uses this information to communicate with the repository.
8. Type the required information and click **Next**.

If the default AEK passphrase is used in the repository configuration, then the Thumbnail Server configurator automatically uses the default AEK passphrase. If the repository is configured with the custom AEK passphrase, then the Thumbnail Server configurator displays a screen for the user to type the AEK passphrase.
9. Type the AEK passphrase.

If the repository you have chosen is part of a Distributed Documentum CM Server environment, a list of server config objects appear for the repository.
10. Select a server config object from the list and click **Next**.

A message notifies you that the repository is successfully configured for Thumbnail Server.
11. Click **Done**. Thumbnail Server is started automatically.

If Thumbnail Server does not start automatically after configuration, then make sure that the mandatory environment variables are set in the Documentum CM Server. Then, start Thumbnail Server manually using the `$DM_HOME/thumbsrv/container/bin/startup.sh` file.

Rerun the configurator for any other repositories you wish to configure for Thumbnail Server.

If you create a new repository on a host where Thumbnail Server is installed, you can run the Thumbnail Server configurator to search for new repositories and configure them automatically.

9.2.2.4 Installing and configuring Thumbnail Server in silent mode

1. Extract the Thumbnail Server installer to a temporary folder.
2. Open the `ts_install.properties` file and verify if you want to change the default values in the file. This file is used for installing Thumbnail Server in silent mode.
3. Open the `ts_config.properties` file and update the values in the file. This file is used to run the Thumbnail Server configuration in silent mode.
4. Navigate to the temporary folder, where the installer is extracted, in the shell prompt.
5. To start the silent installation, type the following command:

- Windows:

```
start /wait thumbserverSetup.exe -f c:\temp\ts_install.properties
```



Note: Use the `start/wait` command to make the silent installation wait until the installation is complete.

- Linux:

```
./thumbserverLinuxSetup.bin -f /temp/ts_install.properties
```



Note: `C:\temp` in Windows and `/temp` in non-Windows mentioned in the preceding commands are the location of the properties files that are extracted from the installer.

6. Navigate to the `$DM_HOME/thumbsrv/configurator` folder in the shell prompt.



Note: If you are configuring Thumbnail Server with HashiCorp Vault, do not add the password in the `ts_config.properties` file but make sure to update the `ts_config.properties` file with the following entries:

```
IS_VAULT_ENABLED=true  
DSIS_URL=http://localhost:8200/dsis  
DSIS_TOKEN=<DSIS token value>
```

To run the Thumbnail Server configuration, type the following command:

- Windows:

```
start /wait thumbServerWinConfigurator.exe -f c:\temp\ts_config.properties
```



Note: Use the start/wait command to make the silent installation wait until the installation is complete.

- Linux:

```
./thumbServerLinuxConfigurator.bin -f /temp/ts_config.properties
```



Note: C:\temp in Windows and /temp in non-Windows mentioned in the preceding commands are the location of the properties files that are extracted from the installer.

9.2.2.5 Post-installation task

9.2.2.5.1 Licensing OpenText Documentum CM

OpenText Documentum CM uses OpenText Directory Services (OTDS) to apply licenses for all the OpenText Documentum CM components. For more information about procuring the license file and configuring OTDS and license, see “[Licensing OpenText Documentum CM](#)” on page 86.

9.2.2.6 Configuring Thumbnail Server for SSL

To enable a Thumbnail Server for Secure Sockets Layer (SSL) encryption, you must configure it to run over HyperText Transfer Protocol over SSL (HTTPS).

The following procedure provides steps for a default SSL configuration. There are cases where a certificate from the Certificate Authority is required. Tomcat website provides information to configure Tomcat for SSL and to use the keytool.

1. Create a certificate keystore using the following keytool command:

- Windows:

```
%JAVA_HOME%\bin\keytool -genkey -alias tomcat -keyalg RSA
```

- non-Windows:

```
$JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA
```



Notes

- Use changeit as the password (or add the *keypass* attribute). You can set the *keystore* and *keypass* parameters to change the default file name and location or the password, if required. The keystore is stored in the *.keystore* file in the user’s home directory, by default, as determined by the *user.home* system property.
- It is recommended to use JDK 11 to generate certificate keystore.

2. Open the *server.xml* file in a text editor. The file is located in the following directory:

- Windows:

```
<DM_HOME>\thumbsrv\container\conf
```

- non-Windows:

```
$DM_HOME/thumbsrv/container/conf
```

3. Locate the HTTPS connector and uncomment the <Connector> node. By default, the connector is on port 8443 and the value of the *SSLEnabled* and the *protocol* attributes are “true” and “HTTP/1.1” respectively. Change the *protocol* value to “org.apache.coyote.http11.Http11NioProtocol”. In addition, add the *sslImplementationName* and *type* attributes.

For example:

```
<Connector port="8443"
protocol="org.apache.coyote.http11.Http11NioProtocol"
sslImplementationName="org.apache.tomcat.util.net.jsse.JSSEImplementation"
SSLEnabled="true"
maxThreads="150">
<SSLHostConfig>
<Certificate certificateKeystoreFile="C:\Users\Administrator\.keystore"
type="RSA"/>
</SSLHostConfig>
</Connector>
```

4. To point the application server to the certificate keystore that contains the SSL certificate, add the following attribute:

```
certificateKeystoreFile="%keystore_location%"
```

Its value must point to the location where the keystore file is created in [step 1](#).
For example: C:\Documents and Settings\myAdminUser\.keystore.



Note: If you used a different password than the default password (changeit) in [step 1](#), add an additional attribute to the preceding Connector node:

```
keystorePass="${password}"
```

5. To avoid the listener conflict, comment the following line in the `server.xml` file:

```
<Listener className="org.apache.catalina.core.AprLifecycleListener"
SSLEngine="on"/>
```

6. Uncomment the following line in the `server.xml` file:

```
<Listener className=
"org.apache.catalina.security.SecurityListener" />
```

7. Make sure that the non-HTTPS default connector port is different from the HTTPS connector port in the `server.xml` file.

For example:

```
<Connector port="8088"
protocol="HTTP/1.1"
connectionTimeout="20000"
redirectPort="8443"/>
```

8. Type the appropriate port value in [step 7](#), if required (by default, the connector port is 8443).

9. Save and close the `server.xml` file.
10. If you ran Thumbnail Server in the HTTPS mode during the Thumbnail Server installation, skip to [step 11](#). Otherwise, update the repository's `thumbnail_url` using the following DQL statement:

If the name of the thumbnail store is `thumbnail_store_01`, then use the following query to update the `thumbnail_url`:

```
update dm_store object set base_url=
'https://<hostname>:<port>/thumbsrv/getThumbnail?'
where name='thumbnail_store_01'
```

Use the port number used in [step 8](#).

11. Restart the repository and Thumbnail Server.
12. To validate the installation, load the URL specified in the query in [step 10](#) from your browser.

On your first attempt to load the URL, you are prompted to accept the certificate.

9.2.2.7 Removing configuration of Thumbnail Server

Before uninstalling Thumbnail Server, remove the Thumbnail Server configuration.

To remove Thumbnail Server configuration manually:

1. Run the Thumbnail Server configurator.
2. Select **Remove Thumbnail Server support from a repository**. A list of configured repositories is displayed.
3. Select the repository from which you want to remove the configuration.
4. Click **Next** and provide the system administrator user name and password.
5. Click **Next**.

To remove Thumbnail Server configuration in silent mode:

1. Open the `ts_config.properties` file and update the value for `INSTALL_TYPE` as `REMOVE` instead of `ADD`.
2. Navigate to the `<DM_HOME>/thumbsrv/configurator` folder in the shell prompt.
3. To run the Thumbnail Server configuration, type the following command:
 - Windows:

```
start /wait thumbServerWinConfigurator.exe -f c:\temp\ts_config.properties
```



Note: Use the `start/wait` command to make the silent installation wait until the installation is complete.

- Linux:

```
./thumbServerLinuxConfigurator.bin -f /temp/ts_config.properties
```



Note: C:\temp in Windows and /temp in non-Windows mentioned in the preceding commands are the location of the properties files that are extracted from the installer.

9.2.2.8 Uninstalling Thumbnail Server from a Windows host

1. Connect to the Documentum CM Server host as the Documentum CM Server installation owner.
2. Make sure that you have removed the configuration for all the repositories that were configured with Thumbnail Server.
3. Stop Thumbnail Server.
4. Click **Start > Settings > Control Panel > Add/Remove Programs**.
5. In the **Change or Remove Programs** window, select **Documentum Thumbnail Server** and click **Change/Remove**.
6. On the welcome screen, click **Next**.
7. Click **Uninstall**.
8. Click **Done**.

If you are going to reinstall Thumbnail Server, you must restart the Thumbnail Server host before doing so.

9.2.2.9 Uninstalling Thumbnail Server from a non-Windows host

1. Connect to the Documentum CM Server host as the Documentum CM Server installation owner.
2. Make sure that you have removed all configuration from all the repositories that were configured with Thumbnail Server.
3. Stop Thumbnail Server.
4. Navigate to the \$DOCUMENTUM_SHARED/uninstall/thumbserver folder.
5. Run **./Uninstall**.
6. On the welcome screen, click **Next**.
7. Click **Uninstall**.
8. Click **Done**.

9.2.2.9.1 Stopping and starting Thumbnail Server on non-Windows

1. Open a command-line interface.
2. Navigate to \$DM_HOME/thumbsrv/container/bin/.
3. To stop Thumbnail Server, type the following command:

```
./shutdown.sh
```
4. Navigate to \$DM_HOME/thumbsrv/container/bin/startup.bat.
5. To start Thumbnail Server, type the following command:

```
./startup.sh
```

When you configure Thumbnail Server, the Thumbnail Server service Logon properties defaults to **Local Service** instead of **Local System**. This is the default behavior of Tomcat with version 9.x.

To run the Thumbnail Server service, perform the following steps:

1. Run services.msc from the command prompt.
2. Navigate to the Thumbnail Server service.
3. Stop the service.
4. Right-click the Thumbnail Server service and click **Properties**.
5. Click the **Log on** tab.
6. Select **Local System Account**.
7. Click **Apply**.
8. Click **OK**.
9. Start the service.
10. Verify the logs generated at <Documentum>\product\<release-version>\thumbsrv\container\logs\.
11. Verify the default URL for Thumbnail Server: <http://<Your Documentum CM Server>:8081/thumbsrv/getThumbnail?/>

9.3 Verifying Thumbnail Server installation

This section describes how to verify that you have successfully installed Thumbnail Server. The best way to verify installation is to import a test file to the repository using a client that uses Thumbnail Server. This assumes that at least one of OpenText™ Documentum™ Content Management Transformation Services - Documents, OpenText™ Documentum™ Content Management Transformation Services - Media server, or OpenText™ Documentum™ Content Management Transformation Services - Audio/Video server is installed, and is correctly generating thumbnail renditions. Consult the product documentation for those products.

Another verification method is to confirm that Thumbnail Server is running. To make sure your Thumbnail Server installation is working correctly, it is recommended that you complete both the tests.

After Thumbnail Server is started, you can verify that it is successfully serving default thumbnails through a web browser using the following steps:

To verify if Thumbnail Server is running:

1. Open a web browser.
2. Type the following address, based on your configuration: `http://<hostname>:<port>/thumbsrv/getThumbnail?/`
A default icon is displayed in the web browser.



Note: Make sure that the host name provided in the URL is accessible from the host where the browser is running.

9.4 Configuring Thumbnail Server in a trusted content store

1. Create a physical folder on the Documentum CM Server system.
2. Click **Administrator > Storage Management > Storage** and create a **Location** using Documentum Administrator and point to the physical folder created in [step 1](#).
3. Create a file store with the following values:
 - Select the location created in [step 2](#).
 - Select **Yes** to encrypt the file store.
 - Select the media type as **Thumbnail Content** for the file store.
4. Run the following DQL queries in the repository:

```
update dm_filestore object set base_url='(select base_url from dm_filestore  
where name = 'thumbnail_store_01')' where name = '<newly created file store>'
```

```
update dm_format object set default_storage = '<r_object_id of newly created
file store>' where name in ('jpeg_th','jpeg_lres','jpeg_story')
```

5. Restart the Documentum CM Server and Transformation Services server.
6. Verify if the thumbnails are stored in the newly created physical location of Documentum CM Server system and if the thumbnails are visible through any client.



Note: For an encrypted filestore, the filestore name and the folder name must be same.

9.5 Administration and configuration

This section describes how to perform administration tasks using Thumbnail Server.

9.5.1 Understanding default thumbnails

OpenText provides a set of default thumbnails, a default rules file, and the `default_thumbnails.dtd` file. When you install Thumbnail Server, the default rules file and DTD are stored in the following repository folder:

```
/System/ThumbnailServer
```

The default thumbnails are stored in the following repository folder:

```
System/ThumbnailServer/thumbnails
```

Thumbnail Server returns a default thumbnail rendition if it receives a URL from an application without a path or a store parameter. The server parses such URLs, and attempts to match the parameters it is given to a rule in the rules file. When a match is found, the associated default thumbnail is returned. If no match is found, the server returns a hard-coded default thumbnail. “Requesting thumbnails” on page 458 provides more information on how thumbnails are requested and returned.

If the URL contains a storage location and path, the proper thumbnail is returned. Otherwise, the default thumbnail for this format type (based on its own mapping in the rules file) is returned. If it is unable to find the mapping, the default thumbnail is returned.

The rules in the rules file reference four parameters:

- `object_type=<type_name>`

`<type_name>` identifies the object type of the returned object. This is the value in the object's `r_object_type` attribute.

- `format=<format_name>`

`<format_name>` identifies the format of the object's primary content object. This is the value of the format's `fdm_format` object.

- `is_vdm=<boolean>`

is_vdm indicates whether the object is a virtual document. The value is either true or false.

- *size=<64>*

If size is specified as 64, then 64x64 default images are served. Otherwise, the 32x32 default images are returned.

For everything else that is not included in the rules file, the default thumbnail is returned.

You can modify the rules file to include rules that reference other parameters.

[“Adding default thumbnails” on page 476](#) contains instructions for adding customized default thumbnails.

If an application uses the DQL THUMBNAIL_URL keyword to request a URL for an object that has no thumbnail rendition, Documentum CM Server returns the object type, format, and virtual document parameters for the object. The application then uses those parameters to build a URL that returns a default thumbnail.

The GET_FILE_URL administrative method does not return these parameters. If a thumbnail rendition does not exist, the method returns empty strings for the path and store argument. The application can send a URL without the path and store arguments, which causes Thumbnail Server to return a default thumbnail. The application can also query the repository to obtain values for the parameters and build a URL using those values.

9.5.2 Adding default thumbnails

If you have custom object types, you can add default thumbnails for those types.

Observe the following guidelines when editing the `default_thumbnails.xml` file:

- Rules are processed in order until a match between the format and the thumbnail is found.
- The file must conform to `default_thumbnails.dtd`.
- All file paths must be relative to:
`/System/ThumbnailServer-thumbnails`
- When a `<fixed_image>` or `<backup_image>` element includes the path to a file, be sure the file already exists.

When processing a URL, Thumbnail Server returns an error if it does not find a file specified in a `<fixed_image>` or `<backup_image>` element.

- File paths specified in a `<pattern>` element are not required to refer to an existing file.

When processing a URL, Thumbnail Server checks for the existence of a file specified in a `<pattern>` element. If the file is not found, the server returns the default thumbnail.

1. Import the thumbnail image to the following repository folder:
`/System/ThumbnailServer-thumbnails/folders`
2. Open the `default_thumbnails.xml` file (located in `/System/ThumbnailServer/`) in a text editor.
3. Scroll down to the appropriate location in the file, as indicated by the preceding guidelines.
4. Following the preceding guidelines, modify the `default_thumbnails.xml` file and add a rule for the new thumbnail.
5. Save and close the `default_thumbnails.xml` file.
6. Log in to Thumbnail Server.
7. Restart the Thumbnail Server service.

9.5.3 Changing the ticket time-out value

The time-out period for an encrypted ticket is set in the `web.xml` file. If required, you can change the value from the default of five minutes.

1. Log in to the Thumbnail Server host as an administrator.
2. Stop the Thumbnail Server service.
3. Navigate to the `WEB-INF` directory, located in:
`<DM_HOME>\thumbsrv\container\webapps\thumbsrv\WEB-INF`
4. Open the `web.xml` file in any text editor.
5. Scroll down to the `init-param` section:

```
<init-param>
    <param-name>ticket_timeout</param-name>
    <param-value>300</param-value>
</init-param>
```
6. Change the `<param-value>` element associated with the `ticket_timeout` parameter name. This becomes the ticket time-out value, in seconds. The value must be numerical.
7. Save and close the `web.xml` file.
8. Restart the Thumbnail Server service.

9.5.4 Activating thumbnail logging

You can configure Thumbnail Server to log thumbnail requests. The logging feature is not activated automatically after the installation of Thumbnail Server and therefore it must be done manually. OpenText strongly recommends that this feature be activated.

Logging will be useful for you for a number of reasons. It will help you to troubleshoot any errors occurring with Thumbnail Server. For example, if you are seeing some thumbnails but not others, examining the Thumbnail Server log will indicate a problem with Thumbnail Server. The log file also indicates errors relating to invalid or expired tickets.

The Thumbnail Server log file, `localhost<yyyy-mm-dd>.log`, is located in the `<DM_HOME>\thumbsrv\container\logs\` directory.

1. Log in to the Thumbnail Server host as an administrator.
2. Stop Thumbnail Server.
3. Navigate to the following directory:

Windows:

```
<DM_HOME>\thumbsrv\container\webapps\thumbsrv\WEB-INF
```

non-Windows:

```
$DM_HOME/thumbsrv/container/webapps/thumbsrv/WEB-INF
```

4. Open the `web.xml` file in any text editor.
5. Set the value of `param-name` to `debug` and the value of `param-value` to `true` as follows:

```
<param-name>debug</param-name>
<param-value>true</param-value>
```
6. Save and close the `web.xml` file.
7. Restart the Thumbnail Server service.

Chapter 10

XML Store

This documentation describes how to deploy XML Store.

To deploy XML Store, you need:

- Administrative privileges on the computer where you are deploying XML Store.
- Working knowledge of Microsoft Windows or Linux.
- Working knowledge of Server administration and configuration, including high-availability (HA) configurations.

10.1 Introduction

XML Store gives Documentum CM Server extended capabilities to store and process XML data in the repository by integrating it with BaseX, a highly scalable native XML database.

XML Store adds standards-based XQuery to the XML capabilities of Documentum CM Server. XML Store enables richer searches and reusing and composing of content in a more flexible manner.

XML Store offers fine-grained access to any content fragment without requiring the content be chunked or burst into individual content objects. This functionality means users can conduct richer searches throughout the enterprise and achieve more flexible reuse and content composition.

XML Store preserves XML content as is, without mapping XML to RDBMS table rows and columns. The XML structure is preserved, allowing users to efficiently and accurately query content at any level of detail. For example, individual elements, attributes, content objects, or metadata attributes, even on large information sets. As a native XML repository, XML Store provides performance advantages over relational databases and file systems through specialized XML indexing methods, caching, and architecture optimized for XML.

XML Store optimizes performance for XML content files and handles access to XML content using XQuery. XML Store works with all other Documentum CM Server features such as versioning, security, and lifecycles, including XML applications.

XML Store provides the following features:

- XML-specific storage type for storing XML content.
- XQuery interface for searching and retrieving XML content and OpenText Documentum CM attributes through XQuery syntax.
- XML Store administrator interface to improve search performance by creating XML indexes, controlling the generation of folders and file or folder mappings.

10.1.1 XML Store-enabled repository

XML Store is an optional module that adds a native XML database server to the Documentum CM Server repository. When you deploy XML Store for a Documentum CM Server, a BaseX instance needs to be deployed earlier, which is a container of one or more BaseX databases. A Documentum CM Server instance is associated with BaseX database that maps to a repository in the Documentum CM Server. When you enable XML Store for a Documentum CM Server repository, a corresponding BaseX database is created.

The following diagram is an illustration of the XML Store-enabled repository:

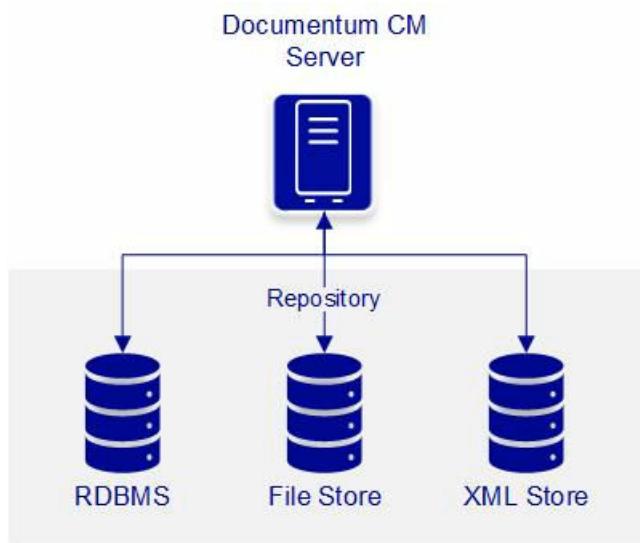


Figure 10-1: XML Store—enabled repository

10.1.2 XML Store deployment mode

Documentum CM Server is designed to use an external XML database for high availability Documentum CM Server configuration.

Select the External XML database deployment mode to have the XML Store failover to a secondary Documentum CM Server in case the primary Documentum CM Server is unavailable. In this mode, you install a standalone BaseX instance before deploying XML Store. [“Java Method Server for High-Availability” on page 116](#) provides the detailed information about high-availability. XML Store uses Java Method Server failover mechanisms that allow methods to run on one or more servers when the Java Method Server is not available for any one Documentum CM Server. When a Java Method Server or Documentum CM Server is down, XML Store can still be accessed and managed through other Documentum CM Servers. The configuration removes a single point of failure from any one Documentum CM Server node for XML Store, but will not provide failover for BaseX itself.

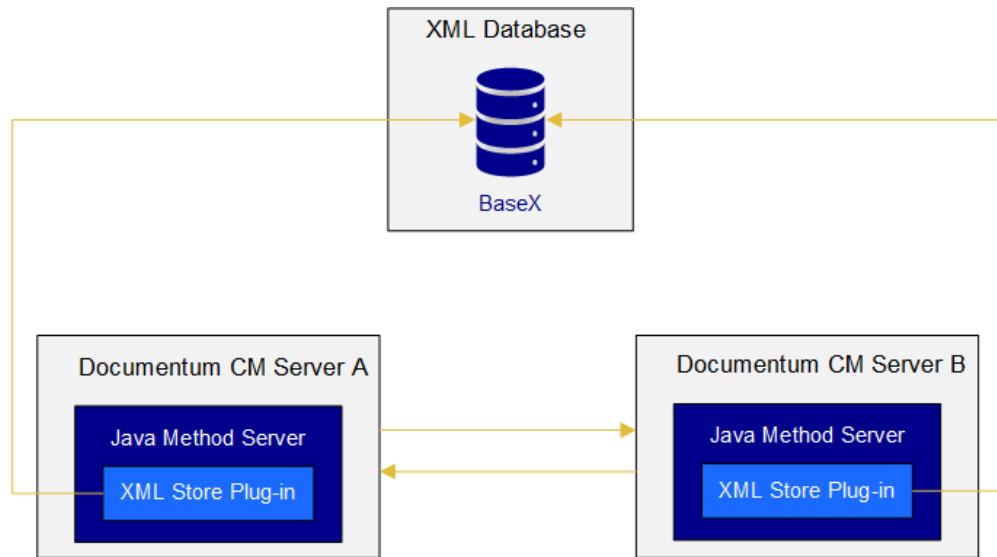


Figure 10-2: External XML database (BaseX)

10.1.3 Prerequisites

Before you deploy XML Store, perform the following tasks:

- Make sure that you have XML Store enabled on the repository to retrieve or delete documents from XML Store.
(Optional) Administrators must have the Content Storage Services to configure assignment policies in the repository.
- Install a standalone BaseX instance and then run the BaseX instance.
If you plan to deploy XML Store in a high-availability (HA) environment using an external XML database, you must install a standalone BaseX instance first before deploying XML Store.

When you install BaseX:

- Note down the BaseX host name, port number, administrator or superuser password, and data file path during the installation. You need to provide the same information when you deploy XML Store.
- If HashiCorp Vault is enabled, the XML Store administrator or the superuser password must be provided in the HashiCorp Vault server in the format, `XML_STORE_PASSWORD/<repository_name>`. For example, `XML_STORE_PASSWORD/testrepo`.
- Make sure that the BaseX client and BaseX server have the BaseX libraries of the same version and the BaseX server has started.

The *BaseX* documentation provides the detailed instructions to install BaseX.

The installation program maintains an error log, which it writes to a file called `setupError.log` in the current working directory. If it cannot write into the working directory, it writes to the home directory of the user who initiated the installation. Reading this file will help you to see what went wrong. If not, it can help OpenText Global Technical Services to help you. Everything in this file is important. Send the entire unedited file if you need to make a support call. The file will not contain passwords or other secure information.

10.2 Deploying XML Store

10.2.1 XML Store deployment

You can enable XML Store, if required, either during Documentum CM Server configuration or as part of an upgrade. “[Documentum CM Server](#)” on page 9 provides the information about Documentum CM Server installation and configuration steps.

10.3 Enabling XML Store for a repository

Whether you are creating a new repository, upgrading a repository from an earlier version, or updating an existing repository, the Documentum CM Server configuration program asks you whether to enable XML Store for the repository if it is not enabled yet.

1. In the Documentum CM Server configuration program, when prompted to enable XML Store for the repository, select **XML Store** and click **Next**.



Note: If you do want to configure XML Store now, you can run `Server_Configuration_Program.exe` on Windows or `dm_launch_server_config_program.sh` on Linux later to launch the configuration program and then activate XML Store.

2. Specify a storage file path for XML Store.
3. Specify the following BaseX information:
 - **XML Store Host:** Host name of the machine where BaseX server is installed and is running.
 - **XML Store Port:** Port on which BaseX server is running.
 - **XML Store Administrator/Superuser Password:** Password of the admin user in BaseX. This password is set when you install BaseX and is used to create BaseX databases in XML Store. If HashiCorp Vault is enabled during Documentum CM Server installation, then the installer retrieves the password automatically from the HashiCorp Vault server.
 - **XML Store Location:** Valid path on the external BaseX host where you want to store the BaseX data. This must be the same path specified for BaseX during its installation.

If you select this option, you must also perform additional configuration to deploy XML Store for multiple Documentum CM Servers. “[Enabling XML Store to work with multiple Documentum CM Servers](#)” on page 483 provides more details.

4. Enter the fully qualified domain name (FQDN) of the Documentum CM Server host.
5. The deployment process begins. When complete, exit the Documentum CM Server configuration program.

10.4 Enabling XML Store to work with multiple Documentum CM Servers

To deploy XML Store to work with multiple Documentum CM Servers, after you complete the deployment process for the first (primary) Documentum CM Server with an external XML database, you must perform some additional steps. “[XML Store deployment mode](#)” on page 480 provides information about XML Store deployment mode with an external BaseX. If you deployed XML Store with an embedded XML database, you need to migrate data in the embedded XML database to an external XML database first and then perform additional steps. For more information, see *OpenText Documentum Content Management - XML Store Administration Guide* (EDCCFMTXML250400-AGD).

To deploy XML Store to work with additional Documentum CM Server nodes:

1. Make sure that the Documentum CM Server and Java Method Server instances have been properly configured for failover in a high-availability environment.
Install additional Documentum CM Server nodes including the Documentum CM Server and Tomcat components.
2. Copy the XML Store plug-in, `XMLStoreService`, from the primary Java Method Server to additional Java Method Server.
 - a. From the primary Documentum CM Server node, find the `XMLStoreService` folder located by default in `$DM_JMS_HOME\webapps\`.
 - b. Copy the entire `XMLStoreService` folder to the same file path in the Tomcat on additional Documentum CM Server nodes.
 - c. If Tomcat is located in different file paths on the additional Documentum CM Server node, verify the entries in the `dfc.properties` and `log4j2.properties` files are correct for the additional Documentum CM Servers. The paths in these files must match the path for the additional Documentum CM Server nodes.
The `dfc.properties` and `log4j2.properties` files are located by default in the following file path: `$DM_JMS_HOME\webapps\XMLStoreService\WEB-INF\classes\`
 - `dfc.properties`: The file has only one line:

```
#include E:\Documentum\config\dfc.properties
```

You must make sure that this path is correct for the additional Documentum CM Servers.

- **log4j2.properties:** Evaluate all paths in this file to make sure that they are accurate for the additional Documentum CM Server nodes. This entry in the file is similar to the following:

```
log4j.appenders.F1.File=$DM_HOME/webapps/logs/XMLStoreService.log
```

- Restart Java Method Server.
- On additional Documentum CM Servers, add `XMLStoreService` to the newly created `jms_config` object, using Documentum Administrator or by running a custom script.
 - Launch Documentum CM Server Manager and start the IDQL utility; then use the following DQL query to obtain the server config ID, `r_object_id`:

```
1> ?,c,select r_object_id, object_name,
server_config_id from dm_jms_config 2> go
```

The results returned are similar to the following:

```
r_object_id object_name 3d015b3880000102
repo 3d015b3880000c57 xmlharcs_repo
```

Determine which server ID to use by examining the `object_name` that matches that of the `dm_jms_config` object on the additional node.

- Navigate to the `$DM_HOME\bin\` file path and run the `dm_jms_admin` tool with the following parameters:

```
dm_jms_admin -docbase docbasename -username installowner -password password
-action add -jms_host_name hostname -jms_port port -servlet_name XhiveConnector
-base_uri /XhiveConnector/servlet/XhiveConnectorServlet -server_config_id
serverconfigID
-proximity 1
```

- Navigate to the `$DM_HOME\bin\` file path and run the `dm_jms_admin` tool with the following parameters:

```
dm_jms_admin -docbase docbasename -username installowner -password password
-action add -jms_host_name hostname -jms_port port -servlet_name
XMLStoreService
-base_uri /XMLStoreService/servlet/XMLStoreServiceServlet
-server_config_id serverconfigID -proximity 1
```

For example:

```
dm_jms_admin -docbase repo -username Administrator -password Pass123
-action add -jms_host_name secondaryCS -jms_port 9080 -servlet_name
XMLStoreService
-base_uri /XMLStoreService/servlet/XMLStoreServiceServlet
-server_config_id 3d015b3880000c57
-proximity 1
```

- Enable `XMLStoreService` on any additional Documentum CM Servers.

- The Documentum CM Server invokes a shared library or DLL to handle content stored in XML Store. The shared library or DLL is represented in the repository by a `dm_plugin` object. The shared library or DLL is stored as the

content of the plug-in object, and is created, by default, in filestore_01 during the Documentum CM Server installation.

- View information for this plug-in object using the following IAPI command:

```
API> retrieve,c,dm_plugin where object_name like '%xmlstore%' dump,c,1
```

For XML Store to function properly on additional Documentum CM Servers in a distributed configuration, this plug-in must be present in a distributed storage area accessible by all Documentum CM Servers.

- Use the DQL statement and information provided to make sure that each Documentum CM Server node has a properly configured file store component in a distributed storage environment. This step is required when storage is distributed. If filestore_01 is accessible by all Documentum CM Servers, for example, through a UNC path, then a distributed store is not needed.
- “[Implementing single-repository models](#)” on page 276 provides more information to complete the necessary steps. Specifically, refer to the step for moving the objects currently in filestore_01 to the distributed store.

5. Verify the **Proximity** value for each **Target Host** on each Documentum CM Server. Because Java Method Server failover will not support remote Content Server (RCS) configuration, the projection target range for each Documentum CM Server must be outside the range of 9000 to 9999. If the projection target falls in this range, Java Method Server failover will not work as expected.

Check both the `server.ini` file and the `server config` object for each Documentum CM Server:

- a. Examine the `server.ini` file for each server, located by default in the `.../dba/config/repositoryname/server.ini/` directory, for the primary Documentum CM Server. Additional Documentum CM Servers .ini file is in the same file path as `server_machine_name_service_name.ini`. For example:

```
[DOCBROKER_PROJECTION_TARGET] host = cshost1 port = 1489
#proximity=9001 [DOCBROKER_PROJECTION_TARGET_1]
#host = #port = #proximity = #host=cshost1
#port=1489 #proximity=9010
```

- b. Examine the `server config` object for each server in Documentum Administrator:

- i. Select **Administration > Basic Configuration > Documentum Servers**, right-click **Server configuration** and then select **Properties**.
- ii. Select **Connection Brokers** and make sure that the **Proximity** value for each **Target Host** falls outside the range of 9000 to 9999.

6. **Optional** If you want to enable the XML Store service for secondary Documentum CM Server, then extract the `XMLStoreService.zip` file located at `<DM_HOME>\install\`.

- a. Copy the contents of the `lib` folder to the `<DOCUMENTUM>\tomcat\shared\lib\` folder.

- b. Update the `xmlstore.properties` file available in the extracted `XMLStoreService` folder with appropriate values.
- c. Move the `XMLStoreService` folder to `<DOCUMENTUM>\tomcat\webapps\`.
- d. Copy the XML Store password file from primary Documentum CM Server to the secondary Documentum CM Server.
- e. Restart Java Method Server.