# OpenText™ Documentum™ Connector for Core Share

## Deployment Guide

Deploy and configure OpenText™ Documentum™ Connector for Core Share.

EDCCOCDFS250400-IGD-EN-01

**OpenText™ Documentum™ Connector for Core Share**
**Deployment Guide**
EDCCOCDFS250400-IGD-EN-01
Rev.: 2025-Nov-25

**This documentation has been created for OpenText™ Documentum™ Connector for Core Share 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

# Table of Contents

Chapter 1

# Overview

Review the sections in this chapter before deploying Documentum Connector for Core Share.

## 1.1 Product overview

OpenText Documentum Connector for Core Share is a clip-on system to OpenText™ Documentum™ Content Management Server to expose authorized content from the OpenText™ Documentum™ Content Management repository to OpenText™ Core Share. When OpenText Documentum Connector for Core Share is deployed with Documentum CM Server, it enables the consumption of content by external collaborators, on multiple-channels of Web, Mobile, and Desktops through the cloud. In addition to automatically publish the content for collaboration, OpenText Documentum Connector for Core Share will also allow users of integrated OpenText Documentum Content Management (CM) client applications to manually choose content and share with preferred collaborators on OpenText Core Share. OpenText Documentum Connector for Core Share also includes two-way synchronization capabilities to ensure the changes that happen to a shared content on OpenText Core Share, can be synchronized back to your repository.

The Documentum Connector for Core Share is an application server instance that runs between the OpenText Core Share and the repositories. It publishes and synchronizes content from repositories to OpenText Core Share folders. Administrators can create or configure profiles and map them to folder synchronization points.

The Documentum Connector for Core Share consists of the following basic components, which are all packaged in the `syncagent.war` file:

- Connector Engine: This is a web application running on an application server, such as Apache® Tomcat™. The Connector Engine creates OpenText Core Share folders and stores profiles and credential information. It is independent of the underlying repository.
- Connector Agent for OpenText Documentum CM: The agent acts as a bridge between the Connector Engine and the repository. The Connector Agent performs the following operations:
  - Navigates the folder structure on a repository
  - Transfers content between a repository and OpenText Core Share
  - Records the synchronization status of each shared file and folder in the database

- Connector Administrator: The Administrator is a configuration tool that you use to manage the contents and folders synchronized between OpenText Documentum CM and OpenText Core Share, including their relevant credentials.

The Documentum Connector for Core Share consists of additional components packaged in the binary:

- Consul Health Check: `Consul.exe` helps in health check of the `micro-services`, `syncagent`, and `syncnshare-manual` services deployed for Documentum Connector for Core Share.
- MetadataService: Enables Documentum Connector for Core Share to publish OpenText Documentum CM metadata attribute values along with the documents to OpenText Core Share.
- CoreNotificationService: Enables Documentum Connector for Core Share (robotic and manual) to allow changes to the shared content on OpenText Core Share to be synchronized back to the repository.
- External Share: `syncnshare-manual-<version>.war` and `ExternalShare.dar` helps in configuring collaborative services for OpenText™ Documentum™ Content Management client.
- MailService: Enables configuration of mail server details. This configuration is optional.

## 1.2 Product behavior

Before deploying the product, be aware of the following product behaviors:

- The Documentum Connector for Core Share publishes only one content per object.
- The Documentum Connector for Core Share allows you to specify rules that limit what is published based on file size, object type, attributes, and formats (including renditions).
- If content fails to upload to OpenText Core Share because of issues in OpenText Core Share (such as a folder in OpenText Core Share failed to be created, OpenText Core Share quota was exceeded, and so on), the previously published content will not be affected.
- OpenText Core Share does not support the virtual document format. If a virtual document is published to OpenText Core Share, it is shown as a normal document.
- The Connector does not support a multiple repository configuration, where a document link from one repository folder is created in another repository folder.
- Special characters are not allowed in file names. If a OpenText Documentum CM document includes specific special characters, the Connector replaces them with an underscore (_) character. The Connector also prevents special characters from being used in the OpenText Core Share folder name.

- When using two-way synchronization, only updates to documents published from OpenText Documentum CM to Core Share are synchronized back to OpenText Documentum CM. New files added to the shared folder on OpenText Core Share are not synchronized back to OpenText Documentum CM.

## 1.3 Deployment overview

The following list is a high level overview of the Connector deployment procedure. The detailed steps are in the next chapter.

1. Check the system environment and verify all prerequisites, as described in "Prerequisites" on page 9.
2. Download and extract `OpenText Documentum CM - APIs and Dev Tools <version> Windows.zip` from My Support. The `OpenText Documentum CM - APIs and Dev Tools <version> Windows` folder includes the latest Documentum Connector for Core Share binaries. Extract the Documentum Connector for Core Share binaries to a folder on your local machine.
3. Deploy the `dccSyncUsers.dar` available in the Documentum Connector for Core Share binary to create the `dmc_sync_users` group in the OpenText Documentum CM.
4. Extract `syncagent.zip` on the application server and encrypt the database password with the `syncagent-encryption-<version>.jar` file.
5. Run the database SQL scripts for your database.
6. In the `syncagent.war` file, edit `database.properties` to configure the connection to the database, and `mail-details.properties` to configure the connection to the mail server.
7. If using Tomcat, copy `syncagent.war` into `$CATALINA_HOME/webapps`.
8. Deploy the Consul, Metadata, Core Notification, and Mail Service microservices.
9. Deploy `syncagent.war` to the application server and then start the instance of the application server.
10. Sign in to the Documentum Connector for Core Share Administrator and enter the credentials for the Core Service account and repository superuser account.
11. Create profiles and folders.

Chapter 2

# Deploying Documentum Connector for Core Share on Windows

## 2.1 Prerequisites

Before you start the deploying, review the following items:

- Review the product *Release Notes* to ensure that your environment meets the supported platforms and system requirements.
- The Connector does not support High Availability (HA) configurations. Ensure that you are not installing the Connector on systems in an HA configuration.
- Locate each Documentum CM Server instance where you plan to install the Connector, and ensure that you have administrator privileges for each.
- Identify the repositories that you will map to OpenText Core Share, and ensure that you have an account for each repository.
- Locate the application servers where you plan to deploy and configure the Connector credentials and database. The Documentum CM Server and application server are not required to be on the same system.
- Ensure that you have an OpenText Core Share Enterprise Edition account. It is recommended that the account has to be dedicated for use by the Documentum Connector for Core Share.
- Ensure you have created Confidential OAuth client credentials in OpenText Core Share. "Creating OAuth Client credentials" on page 27 provides more information.
- Ensure that you have a repository superuser account available.
- Choose a database to use with the Connector. Ensure that the database is configured for case-sensitive collation.
- Ensure to configure OpenText™ Documentum™ Content Management Foundation REST API to the preferred Documentum CM Server repository.
- Configure the following parameter in the `rest-api-runtime.properties` file located at `<application-server>\webapps\<dctm-rest>\WEB-INF\classes`:

```
rest.paging.default.size= 1000
```

```
rest.paging.max.size=10000
```

- Ensure the repository superuser account belongs to the group `dmc_sync_users`. If the account does not belong to the `dmc_sync_users` group, then add the user to the group using the Documentum Administrator client.

- Ensure to create user partition and OAuth Client ID to use OpenText™ Directory Services (OTDS) authentication. For more information, see *OpenText Directory Services - Installation and Administration Guide (OTDS-IWC)*.
- Ensure that you have a Microsoft 365 subscription with permission to add or modify the Microsoft Entra admin center and grant application or user permissions listed in the **App registrations**. Register the Mail service on Microsoft Entra admin center. For more information, see “Prerequisites:“ on page 18 step 3

## 2.2 Deploying Vault services

Documentum Secret Integration Service (DSIS) is a standalone program that acts as a mediator between OpenText Documentum CM components and HashiCorp Vault. It is packaged inside the Documentum CM Server installer. For more information, see *OpenText Documentum Content Management - Server and Server Extensions Installation Guide (EDCSY250400-IGD)*.

**Note:** Download the DSIS binaries from OpenText™ My Support.

### 2.2.1 Configuring Documentum Secret Integration Service

This procedure describes how to configure DSIS services on Documentum Connector for Core Share:

1. Copy the DSIS folder from the Documentum Connector for Core Share binaries and place it in *<DCC_HOME>* location, for example, `C:\Program Files\DCC\`.
2. In DSIS, run the following command to generate token:
   - For Windows:
     ```
     java -cp .;dsis.jar;lib/* com.dctm.vault.TokenGenerator
     ```
   - For Linux:
     ```
     java -cp .:dsis.jar:lib/* com.dctm.vault.TokenGenerator
     ```
3. In the `application.properties` file, set the value of `vaultEnabled` to `true` along with the other existing properties before starting the Vault daemon process.

   a. Configure one of the authentication methods:
      ```
      spring.cloud.vault.scheme=<http or https>
      spring.cloud.vault.host=<vault URL>
      spring.cloud.vault.port=<vault port>
      Token Based Authentication:
      spring.cloud.vault.token=<Token generated with READ and LIST access to secrets>
      Role Based Authentication:
      spring.cloud.vault.authentication=APPROLE
      spring.cloud.vault.app-role.role-id=<role_id>
      spring.cloud.vault.app-role.secret-id=<secret_id>
      spring.cloud.vault.app-role.role=<role>
      spring.cloud.vault.app-role.app-role-path=<role_path>
      spring.cloud.vault.namespace=<namespace>
      Kubernetes Token Authentication:
      ```

```
spring.cloud.vault.authentication=KUBERNETES
spring.cloud.vault.namespace=<namespace>
spring.cloud.vault.kubernetes.role=<role>
spring.cloud.vault.kubernetes.kubernetes-path=<path>
spring.cloud.vault.kubernetes.service-account-token-file=<token_file_location>
TLS Authentication:
<configuration to be added>

vaultEnabled=true
```

b. Configure the `dsis.dctm.kvpath` to the parent path if the secrets are configured under one key path. Otherwise, individual secret is configured with its corresponding key path. The Key-Value (KV) path can be configured to any secret that is used for validation during initialization. The validation can be turned off by setting dsis.dctm.enforceListDuringInit to `false`.

For example, `application.properties` file.

```
#Local Dev Vault - Working
spring.cloud.vault.token=hvs.SgbkRzILB3WGH34fCgyonFdd
spring.cloud.vault.scheme=http
spring.cloud.vault.host=10.194.37.188
spring.cloud.vault.port=8200

#spring.cloud.vault.scheme=https
#spring.cloud.vault.host=vault.otxlab.net
#spring.cloud.vault.port=443
#spring.cloud.vault.authentication=APPROLE
#spring.cloud.vault.app-role.role-id=218019e3-73d3-ed3d-278d-d4ef93607e94
#spring.cloud.vault.app-role.secret-id=2991c37f-2707-def6-28a7-05bc41d28bc5
#spring.cloud.vault.app-role.role=documentum
#spring.cloud.vault.app-role.app-role-path=approle
#spring.cloud.vault.namespace=private-cloud

#Must end with forward slash
#dsis.dctm.kvpath=/kv/data/lab/dhatchana2133-otad-local/prod-1/documentum/
dsis.dctm.kvpath=/dctm_secrets_dev/data/
#host must be localhost always
dsis.dctm.host=localhost
#port can be changed based on the availability
dsis.dctm.port=8203
dsis.dctm.executorThreadCount=10
#For additional security, though the daemon is listening only to localhost
dsis.dctm.token=9102201203969033446
#Making the below to false avoid token generation and configuration.
#It is recommended to have this flag to true, if the machine is used by
multiple users
dsis.dctm.tokenNeeded=true

vaultEnabled=true
```

**Notes**

- You need to add the `vaultEnabled` field in the `application.properties` file and set the value to `true` to use the Vault feature with DCC.
- The KV path and authentication process details must be passed correctly in the `application.properties` file.

c. Start DSIS with any of the following methods:

- For Windows:

- To start the service as part of the same window:

```
dsis_start.bat >> dsis.log
```

- To start the service as a background process:

```
dsis_start_background.bat
```

- For Linux: To start the service as background process:

```
dsis_start.sh
```

  The `dsis.log` is created in the same location for all the above startup.

To check if DSIS service is up and running, see `dsis.log`.

d. The DSIS service can be accessed by providing the following endpoints in the API testing application (for example, Postman):

- Select `GET` method and set `http://localhost:8200/dsis/checkstatus` as the location. The response is `200` if the DSIS is initialized successfully.
- Select `GET` method and set `http://localhost:8200/dsis/secret/`*`<secret_name>`*`/`*`<key_name>`* as the location - Will provide the password if the secrets with the key is found.

  In the preceding requests, `dsis-daemon-token` header has to be passed with the value generated in step 2.

  **Note:** For more information on the `application.properties` configuration, see *OpenText Documentum Content Management - Server and Server Extensions Installation Guide (EDCSY250400-IGD)*.

## 2.2.2 Storing key-value pair in Vault

The `secret_name`, `Key_name`, and `value` are stored on Vault by providing the following parameters in the API testing application (for example, Postman):

1. Select `POST` method and set `http://localhost:8200/dsis/secret/`*`<secret_name>`*`/`*`<key_name>`* as the location.
2. In the **Header** tab, add the following **Keys** and **Values**:
   - **dsis-daemon-token**: Specify the token generated in step 2.
   - **secret-value**: Specify the secret value.

   **Note:** For more information, see *OpenText Documentum Content Management - Server and Server Extensions Installation Guide (EDCSY250400-IGD)*.

   The following table provides the list of `secret_name` and `key_name`.

| `secret_name` | `key_name` | Default value | Description |
|---|---|---|---|
| DCC_DATABASE_PASSWORD | syncagent | *<encrypted db password for syncagent>* | The database password for DCC services. Keyname is the service name. |
| | syncnshareManual | *<encrypted db password for syncshareManual>* | |
| | metadata | *<encrypted db password for metadata>* | |
| | coreNotification | *<encrypted db password for coreNotification>* | |
| DCC_MAIL_PASSWORD | DCC_MAIL_PASSWORD | *<encrypted password for mailservice>* | Encrypted mail password used for sending mail. Keyname is the service name. |
| DCC_SOURCE_REPO_PASSWORD | testenv | *<encrypted source repo password for testenv>* | The password of repo used in Syncnshare-Manual v2 APIs. Keyname is the docbase name. |
| | testdoc | *<encrypted source repo password for testdoc>* | |
| DCC_TARGET_REPO_CLIENT_SECRET | DCCCORE_CLIENTID | *<coreshare encrypted client secret>* | OpenText Core Share encrypted Client Secret. Keyname is the `client_id`. |

**Notes**

- The `secret_name` and `key_name` must be same as mentioned in the above table. Only, the valid inputs to be set for the `Default_Values`.
- For Documentum Connector for Core Share services, all the secrets are passed as encrypted. The values must be first encrypted using `passwordutil.jar` provided with the binaries (`syncnshare-manual-encryption-`*`<version>`*`.xxxxx.jar` and `syncagent-encryption-`*`<version>`*`.xxxx.jar`).

### 2.2.3 Configuring Documentum Connector for Core Share for Hashicorp Vault

When the **vaultEnabled** field is set to `true` in the `application.properties` file located at *`<DCC_HOME>`*`\dsis`, perform the following configuration changes in the Documentum Connector for Core Share service:

- In all the `database.properties` file, **password** field must be kept as commented or empty (`syncagent`, `syncnshare-manual`, `metadata`, `coreNotification`).
- For mail service, if SMTP server requires credentials, **mailpassword** field must be kept as commented or empty.
- In the `set-up.properties` file (`syncnshare-manual`), the **sourceRepoPassword** and **targetRepoClientSecret** fields must be kept as commented or empty.

### 2.2.4 Starting Documentum Connector for Core Share service with DSIS service

This procedure describes how to start the Documentum Connector for Core Share service with the DSIS service.

1. Start the DSIS service.
2. Start Consul service by executing the `runconsul.bat` file.
3. Start the Documentum Connector for Core Share service by executing the `runservices.bat` file.
4. Start the `syncagent` and `synnshare-manual` service by starting Tomcat.

   Verify the `dsis.log` file to ensure that all the secret values are used from the Vault to the respective Documentum Connector for Core Share services.

## 2.3 Deploying DARs on Documentum CM Server

This procedure describes how to configure the Connector agent for each repository.

1. Download the Documentum Connector for Core Share binary, and from `\`*`<version>`*`\DccSyncUsers`, copy the `dccSyncUsers.dar` file to the Documentum CM Server machine.
2. Deploy `dccSyncUsers.dar` available in `\`*`<version>`*`\DccSyncUsers` to create the `dmc_sync_users` group on Documentum CM Server.
3. Deploy the `ExternalShare.dar` file on Documentum CM Server with the install parameter, `ExternalShare.installparam`, to create the required relation:

```
<build>\ExternalShare
-ExternalShare.dar
-ExternalShare.installparam
```

4. Sign in to the Documentum Administrator client, and click the group named `dmc_sync_users` under **User Management** > **Groups**.

5. Create a dedicated superuser for Documentum Connector for Core Share and ensure that it is part of the `dmc_sync_users` group.

## 2.4 Deploying on Application Server

This procedure describes how to set up credentials and configure the database on an application server.

1. Download and extract the contents of `syncagent-<version>.zip` to the application server.

2. In the resulting directory, go to the `\deployment\supported-databases\` folder, and then open the subfolder (oracle, sqlserver, or postgresql) that matches your database.

3. Run the following SQL scripts available in `\deployment\supported-databases\` folder accordingly:

   - For Oracle:
     `connector_tables_oracle.sql`
     `quartz_tables_oracle.sql`
     `DCC_Maintenance_oracle.sql`

   - For Microsoft SQL:
     `connector_tables_sqlserver.sql`
     `quartz_tables_sqlserver.sql`
     `DCC_Maintenance_sqlserver.sql`

   - For Amazon Aurora and on-premises PostgreSQL:
     `connector_tables_postgres.sql`
     `quartz_tables_postgres.sql`
     `DCC_Maintenance_postgres.sql`

   **Notes**

   You must run these SQL script files before starting the Connector.

   - These scripts create necessary tables required by the Connector.
   - When you run these scripts for Microsoft SQL, the `DCC_Syncagent` database is created by default.

4. In the top level syncagent directory that includes `syncagent-encryption-<version>.jar`, run the following command to encrypt a password for the database:

```
java -cp syncagent-encryption-<version>.jar
com.emc.syncplicity.connector.util.encryption.PasswordUtils <password>
```

   Where <*password*> is the password that you need to use in the step 5 to connect to the database.

Copy the encrypted value to use in the step 5.

5. Configure `syncagent.war` as follows:

   a. Extract `syncagent.war`.

   b. In the WAR file, go to `\WEB-INF\classes\config\` and edit the following:

      - `database.properties`

        – Set **jdbc.username** to any name you choose.

        – Set **jdbc.password** with the encrypted value from step 4.

        – Uncomment the **jdbc.dbSchemaName**, **jdbc.driverClassName**, and **jdbc.url** values associated with the required database.

        For example, if using Oracle:

```
#jdbc.dbSchemaName=oracle
#jdbc.driverClassName=oracle.jdbc.driver.OracleDriver
#jdbc.url=jdbc:oracle:thin:@//:1521/<db name>
#quartz.dbDelegateClass=org.quartz.impl.jdbcjobstore.oracle.OracleDele
gate
```

        For example, if using Microsoft SQL Server:

```
#jdbc.dbSchemaName=sqlserver
#jdbc.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDriver
#jdbc.url=jdbc:sqlserver://;databaseName=<db name>
#quartz.dbDelegateClass=org.quartz.impl.jdbcjobstore.MSSQLDelegate
```

        For example, if using Amazon Aurora or on-premises PostgreSQL Server:

```
#jdbc.dbSchemaName=postgres
#jdbc.driverClassName=org.postgresql.Driver
#jdbc.url=jdbc:postgresql://<DBHOST>:<PORT>/<DBNAME?
escapeSyntaxCallMode=callIfNoReturn
#quartz.dbDelegateClass=org.quartz.impl.jdbcjobstore.PostgreSQLDelegat
e
```

        For example, if SSL is enabled for PostgreSQL Server:

```
jdbc.url=jdbc:postgresql://<host>:<port>/<db_name>?
escapeSyntaxCallMode=callIfNoReturn&ssl=true&sslmode=verify-
ca&sslrootcert=<path to root-ca.crt>&sslcert=
<Path to server .crt file>&sslkey=<path to key file>
```

        **Notes**

        ○ The Key file must be in the `.der` format. Get the valid certificates and key file (`root-ca.crt`, `client.crt`, and `key.der`) for the PostgresSQL SSL.

        ○ The SSL modes are `prefer`, `require`, `verify-ca`, and `verify-full`.

        ○ The certificate `root-ca.crt` is mandatory for SSL modes `verify-ca` and `verify-full`, and optional for SSL modes `prefer` and `require`.

        ○ The certificates `client.crt` and `key.der` are required if the database is configured to require client certificate.

For example, if SSL is enabled for Oracle:

```
jdbc.url=jdbc:oracle:thin:@(description=(address=(protocol=tcps)
(host=<DB-HOST>)(port=<DB-PORT>))
(connect_data=(service_name=<SID>)) (SECURITY=(MY_WALLET_DIRECTORY =
 <Path of the wallet files>)))
```

– If necessary, change `jdbc.url` to a value appropriate for your database instance. For example, the URL will not include localhost if the database is on a different machine.

- `mail-details.properties`

```
#the details 'from','to' and 'subject' are mandatory fields if mail
service is being used.
#from=<sender email id>
#to=<recipient email ids separated by comma>
#bcc=<Bcc recipient list separated by comma>
#cc=<Cc recipient list separated by comma>
#subject=<Subject line for the email to be sent for Failed items>
```

c. Archive `syncagent.war`.

d. Navigate to `\webapps\syncagent\WEB-INF\classes\config\` and edit `service-registry-location.properties` as follows:

By default, the `http` protocol is enabled and the `serviceregistry-port` for `http` is `8500`. To use the `https` protocol, set `server-protocol` to `https`, `serviceregistry-port` as `8501`, and provide the host name of the server.

**Note:** When all the services are deployed on the same system, the service registry and server host share the same host name. If the services are deployed on distributed systems, the server host is the host name of the system on which Documentum Connector for Core Share is deployed.

For the `http` protocol, *<serviceregistry-fqdn>* is the same name as service registry and for the `https` protocol, the certificate CN name must be provided.

```
serviceregistry-fqdn = <serviceregistry>

#Server port
server-port=<server port>
server-protocol=<http/https>
server-host=<hostname of the server required only for https protocol>
serviceregistry-port=<valid port number>
```

6. If using Tomcat as the application server, perform the following tasks:

a. Copy `syncagent.war` into `$CATALINA_HOME\webapps\`.

b. Ensure that the application server is configured for UTF encoding by setting `URIEncoding="UTF-8"` as an attribute with the *<connector>* element in the `server.xml` configuration. If there are multiple *<connector>* elements, ensure to change the element used by `syncagent.war`.

**Tip:** The *<connector>* element used by `syncagent` would specify an `http` or `https` port, such as 8080 or 8443 respectively.

7. Do not start the application server.

8. Deploy microservices. See "Deploying microservices" on page 18. Deploying microservices is no longer optional.

9. After deploying microservices, proceed to "Completing the deployment" on page 26.

## 2.5 Deploying microservices

Ensure that you deploy consul, metadata, core notification, and mail service microservices in the same machine as `syncagent.war`.

**Prerequisites:**

1. Create the directory *`<DCC_HOME>`* and set the environment variable `DCC_HOME` entry to point to this path.

2. From the downloaded Documentum Connector for Core Share binaries, copy `MetadataService`, `Consul`, `CoreNotificationService`, `MailService`, `runConsulService.bat`, `runServices.bat`, `stopServices.bat`, and `uninstallServices.bat` from the extracted binary to *`<DCC_HOME>`*.

3. Mail service must be registered on Microsoft Entra admin center before deploying the Documentum Connector for Core Share application. Perform the following procedure to register the mail service on Microsoft Entra admin center.

   a. Sign in to Microsoft Entra admin center and click **App registrations**.

   b. On the **App registrations** page, perform the following:

      i. On the **New registration** tab, specify the name of the application.

      ii. Click **Register**.

   c. On the **App registrations** page, click the registered application.

   d. On the registered application page, perform the following:

      **Go to Manage > Authentication**

      i. Select **Accounts in this organizational directory only (Single tenant)**.

      ii. Click **Configure**.

      iii. Go to **Overview** from the pane and copy `Application (client) ID` and `Directory (tenant) ID`.

      **Go to Manage > Certificates & secrets**

      i. Click **New client secret**.

      ii. Specify the description in **Description**, and click **Add**.

      iii. Copy the added value of the secret and save.

**Go to Manage > API permissions**

i. Click **Add a permission**.

ii. Select **Microsoft Graph**.

iii. Select **Application permissions** and select the following permissions:

```
Mail.ReadWrite
Mail.send
```

iv. Select **Delegated permissions** and select the following delegated permissions:

```
User.Read
```

v. Select **Grant admin consent** to grant a permission.

**Note:** You must have a Microsoft 365 subscription with permission to add or modify Microsoft Entra admin center, and grant application or user permissions listed in the **App registrations**.

## 2.5.1 Deploying Consul

1. From the downloaded Documentum Connector for Core Share binaries, copy one of the following file to the *<DCC_HOME>*`\consul_config` location:

   - `basic_config.json`—to configure the `http` protocol.
   - `basic_config_https.json`—to configure the `https` protocol.

2. Configure the protocol-specific settings:

   - `http`: Open the *<DCC_HOME>*`\Consul\basic_config.json` file and provide the Documentum Connector for Core Share home path in the *`data_dir`* parameter and the server IP address of the virtual machine to the *`bind_addr`* parameter.

```
{
    "data_dir": "<Path to DCC_HOME>/Consul_data_dir",
    "log_level": "INFO",
    "ui":true,
    "bootstrap":true,
    "bind_addr":"IP ADDRESS TO BE FILLED HERE ",
    "client_addr":"0.0.0.0",
    "server": true
}
```

   - `https`: Open the *<DCC_HOME>*`\Consul\basic_config_https.json` file, and provide the Documentum Connector for Core Share home path in the *`data_dir`* parameter, the server IP address of the virtual machine to the *`bind_addr`* parameter, and enter the following *`tls`* parameters:

     – `key_file`: Path of the `crt.key` file. For example: `C:\\dcccert\\cert.key`

     – `cert_file`: Path of the `crt.pem` file. For example: `C:\\dcccert\\crt.pem`

  – `ca_file`: Path of the `ca.crt` file. For example: `C:\\dcccert\\ca.crt`

```
{
    "data_dir": "<Path to DCC_HOME>/Consul_data_dir",
    "log_level": "INFO",
    "ui":true,
    "bootstrap":true,
    "bind_addr":"IP ADDRESS TO BE FILLED HERE ",
    "client_addr":"0.0.0.0",
    "ports": {
      "http": -1,
      "https": 8501
    },
    "tls": {
      "defaults": {
        "key_file": "Path to cert.key file to be provided here"
         "cert_file": "Path to cert.pem file to be provided here"
        "ca_file": "Path to ca.crt file to be provided here"
        "verify_incoming": false,
        "verify_outgoing": false,
        "verify_server_hostname": false
      }
}
```

3. Open the *<DCC_HOME>*`\Consul\ConsulService` file and configure the `http` or `https` protocol.

   By default, the `http` protocol is selected. When using the `http` protocol, comment the `https` related parameters and alternatively, comment the `http` related parameters when using the `https` protocol.

   The parameters are as follows:

   - `http`: `<!--<arguments>agent -enable-script-checks -config-file="<Path to DCC_HOME>/Consul/basic_config.jason"</arguments>-->`
   - `https`: `<arguments>agent -enable-script-checks -config-file="<Path to DCC_HOME>/Consul/basic_config_https.jason"</arguments>`

```
<Service>
    <id>ConsulService</id>
    <name>ConsulService</name>
    <description>Consul windows service</description>
    <executable>consul</executable>
<!-- Please use the below line when enabling http-->
    <!--<arguments>agent -enable-script-checks -config-file="<Path to DCC_HOME>/
Consul/basic_config_https.jason"</arguments>-->
<!-- Please use the below line when enabling https-->
    <arguments>agent -enable-script-checks -config-file="<Path to DCC_HOME>/Consul/
basic_config_https.jason"</arguments>
    <logpath><Path to DCC_HOME>/Logs/consulService</logpath>
    <logmode>rotate</logmode>
<Service>
```

4. In the machine where you have deployed the service, add the following entries in the `C:\Windows\System32\drivers\etc\hosts` file:

```
localhost serviceRegistry
127.0.0.1 serviceRegistry
```

**Note:** If an error occurs, you can check the logs in *<DCC_HOME>*`\Logs`.

### 2.5.2 Deploying Metadata service

1. Go to the *<DCC_HOME>*`\MetadataService\deployment\`*<database>*`\` folder and run one of the following scripts:

   - For Oracle: `metadata_tables_oracle.sql`
   - For Microsoft SQL: `metadata_tables_sqlserver.sql`
   - For Amazon Aurora or on-premises PostgreSQL: `metadata_tables_postgres.sql`

   **Notes**

   - The script creates tables required by the metadata microservice.
   - For Microsoft SQL, the DCC_Metadata database is created by default when you run this script.

2. Go to the *<DCC_HOME>*`\MetadataService\Config\` folder and edit the `database.properties` file to connect to the required database as follows:

   - Set **jdbc.username** to valid database user.
   - Set **jdbc.password** with the encrypted value. For more information, see step 4
   - Uncomment the **jdbc.dbSchemaName**, **jdbc.driverClassName**, and **jdbc.url** values associated with the required database.

   For example, if using Oracle:

   ```
   #jdbc.dbSchemaName=oracle
   #jdbc.driverClassName=oracle.jdbc.driver.OracleDriver
   #jdbc.url=jdbc:oracle:thin:@//:1521/<db name>
   #quartz.dbDelegateClass=org.quartz.impl.jdbcjobstore.oracle.OracleDelegate
   ```

   For example, if using Microsoft SQL Server:

   ```
   #jdbc.dbSchemaName=sqlserver
   #jdbc.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDriver
   #jdbc.url=jdbc:sqlserver://;databaseName=<db name>
   #quartz.dbDelegateClass=org.quartz.impl.jdbcjobstore.MSSQLDelegate
   ```

   For example, if using Amazon Aurora or on-premises PostgreSQL Server:

   ```
   #jdbc.dbSchemaName=postgres
   #jdbc.driverClassName=org.postgresql.Driver
   #jdbc.url=jdbc:postgresql://<DBHOST>:<PORT>/<DBNAME?
   escapeSyntaxCallMode=callIfNoReturn
   #quartz.dbDelegateClass=org.quartz.impl.jdbcjobstore.PostgreSQLDelegate
   ```

3. Edit the `log4j2.properties` file in the same location to configure the Metadata service logs location and log level. By default, it is configured to create the `metadataService.log` in the *<DCC_HOME>*`\MetadataService\dcc-logs\` folder.

### 2.5.3 Deploying Core Notification service

1. Go to the `<DCC_HOME>\CoreNotificationService\deployment\<database>\` folder and run one of the following scripts:

   - For Oracle: `dcc_core_notifications_oracle.sql`
   - For Microsoft SQL: `dcc_core_notifications_sqlserver.sql`
   - For Amazon Aurora or on-premises PostgreSQL: `dcc_core_notifications_postgres.sql`

   **Notes**

   - The script creates necessary tables required by the Core notification microservice.
   - For Microsoft SQL, the `DCC_Notification` database is created by default when you run this script.

2. Go to the `<DCC_HOME>\CoreNotificationService\Config\` folder and edit the `database.properties` file to connect to the required database as follows:

   - Set **jdbc.username** to valid database user.
   - Set **jdbc.password** with the encrypted value. For more information, see step 4
   - Uncomment the **jdbc.dbSchemaName**, **jdbc.driverClassName**, and **jdbc.url** values associated with the required database.

   For example, if using Oracle:

   ```
   #jdbc.dbSchemaName=oracle
   #jdbc.driverClassName=oracle.jdbc.driver.OracleDriver
   #jdbc.url=jdbc:oracle:thin:@//:1521/<db name>
   #quartz.dbDelegateClass=org.quartz.impl.jdbcjobstore.oracle.OracleDelegate
   ```

   For example, if using Microsoft SQL Server:

   ```
   #jdbc.dbSchemaName=sqlserver
   #jdbc.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDriver
   #jdbc.url=jdbc:sqlserver://;databaseName=<db name>
   #quartz.dbDelegateClass=org.quartz.impl.jdbcjobstore.MSSQLDelegate
   ```

   For example, if using Amazon Aurora or on-premises PostgreSQL Server:

   ```
   #jdbc.dbSchemaName=postgres
   #jdbc.driverClassName=org.postgresql.Driver
   #jdbc.url=jdbc:postgresql://<DBHOST>:<PORT>/<DBNAME?
   escapeSyntaxCallMode=callIfNoReturn
   #quartz.dbDelegateClass=org.quartz.impl.jdbcjobstore.PostgreSQLDelegate
   ```

3. Edit the `log4j2.properties` file in the same location to configure the CoreNotificationService service logs location and log level. By default, it is configured to create `CoreNotificationService.log` in the `<DCC_HOME>\CoreNotificationService\dcc-logs\` folder.

4. Go to the `<DCC_HOME>\CoreNotificationService\Config\` folder and edit the `ExtApi.config` file.

- Set the robotic and/or manual service as follows:
  - To use only robotic service, comment the `dcc-manual-service` tag.

    **Example for `http` Robotic service:**

    `#dcc-manual-service = syncnshare-manual`

    `dcc-robot-service = syncagent`

    **Example for `https` Robotic service:**

    `dcc-secure-robot-service = syncagent`

    `#dcc-secure-manual-service = syncnshare-manual`
  - To use only manual service, comment the `dcc-robotic-service` tag.

    **Example for `http` Manual service:**

    `dcc-manual-service = syncnshare-manual`

    `#dcc-robot-service = syncagent`

    **Example for `https` Manual service:**

    `#dcc-secure-robot-service = syncagent`

    `dcc-secure-manual-service = syncnshare-manual`
  - To use both robotic and manual services, ensure that both tags are enabled.

    **Example for `http`:**

    `dcc-manual-service = syncnshare-manual`

    `dcc-robot-service = syncagent`

    **Example for `https`:**

    `dcc-secure-robot-service = syncagent`

    `dcc-secure-manual-service = syncnshare-manual`

    Where, `syncnshare-manual` is the name of the manual share application deployed in Tomcat and `syncagent` is the name of the robotic application deployed in Tomcat.

    **Notes**

    - When using the `http` protocol, ensure to comment the `https` related parameters and alternatively, comment the `http` related parameters when using the `https` protocol.
    - If the environment is SSL enabled, then enable the proxy and provide the `proxyhost`, `proxyport`, and `noproxy` parameters in the environment variable of the system.

      ```
      https.proxyHost
      https.proxyPort
      https.nonProxyHosts
      ```

      The `nonproxyHosts` inputs must be separated by pipe operator `|`.

- Provide the host name of the server for `serviceregistry`.

  ```
  #Externalizing serviceregistry
  serviceregistry = <Provide the hostname of the server>
  ```

  **Note:** For the `http` protocol, <*serviceregistry*> is the same name as service registry and for the `https` protocol, the certificate CN name must be provided.

- Set the `http` or `https` protocol in the environment variable of the system.

  By default, the `http` protocol is enabled and the `serviceregistry-port` for `http` is `8500`. To use the `https` protocol, set `httpsEnabled` to `true`, `serviceregistry-port` as `8501`, and provide the host name of the server.

  **Note:** When all the services are deployed on the same system, the service registry and server host share the same host name. If the services are deployed on distributed systems, the server host is the host name of the system on which Documentum Connector for Core Share is deployed. If you are using the `https` protocol, ensure the OpenText Core Share certificate is added to both the Java cacerts and truststore.

  ```
  #healthcheck properties
  httpsEnabled=false
  serviceregistry-port=<valid port number>
  server-host=<hostname of the server required only for https protocol>
  ```

## 2.5.4 Deploying Mail Service

1. From the downloaded Documentum Connector for Core Share binaries, copy the `MailService` folder to the *<DCC_HOME>* location, for example, `C:\Program Files\DCC\`.

2. Go to the *<DCC_HOME>*`\MailService\Config\` folder.

3. Edit the `mailservice.properties` file to configure the mail server details.

   Mail service supports both basic and OAuth authentication type. Provide the authentication type as basic or OAuth accordingly.

   ```
   authenticationType= <oauth/basic>
   ```

   - If authentication type is OAuth, then provide the following details:

     ```
     Below details are applicable for Microsoft Exchange oauth changes.
     # Provide the Service Provider name, default is Microsoft.
       serviceProvider= Microsoft
     # Provide the tenant ID from the Microsoft Entra admin center.
       tenantId = <tenant id>
     # Provide the client ID from the Microsoft Entra admin center.
       clientId = <client id>
     # If authenticationType is oauth, then provide the user's encrypted client
     secret from the Microsoft Entra admin center after completing the app
     registration.
       clientSecret= <client secret>
     # below 2 fields tokenUrl and graphApiUrl need not be changed
       tokenUrl= https://login.microsoftonline.com/{tenantId}/oauth2/v2.0/token
       graphApiUrl = https://graph.microsoft.com/v1.0/users/{sender-email}/sendMail
     ```

- If authentication type is basic, then provide the following details:

```
## Below details need to be provided if authenticationType is basic
#Provide the host, port, user email, and password for the SMTP server as
required. Uncomment the usermail and
 mailPassword fields if authentication is needed.
 smtpmailhost = <SMTP server Host when Authentication Mode is basic>
 smtpmailport = <SMTP server Port when Authentication Mode is basic>
#usermail=<Provide the usermail  if required for the SMTP server>
#mailpassword=<Provide the encrypted mail Password if required for the SMTP
server>
 starttls=false
```

**Notes**

- Run the following command to encrypt the client secret using `syncagent-encryption-<version>.jar` available in the Documentum Connector for Core Share binary.

```
java -cp syncagent-encryption-<version>.jar
com.emc.syncplicity.connector.util.encryption.PasswordUtils <client secret>
```

  Where <*client secret*> is the client secret generated in the Microsoft Entra admin center. For more information, see "Prerequisites:" on page 18 step 3.

4. Edit the `log4j2.xml` file in the same location to configure the Mail Service log location and log level. By default, it is configured to create `MailService.log` in the `<DCC_HOME>\MailService\dcc-logs\` folder.

## 2.6 Starting microservices

Ensure that you deploy consul, metadata, core notification, and mail service microservices in the same machine as `syncagent.war`.

1. Open a command prompt in administrator mode at *<DCC_HOME>* and run `runConsulService.bat`. The Consul service must now be running.
2. To verify, open the following URL in the browser: `http://serviceregistry:8500/ui` or `https://<hostname>:8501/ui`.
3. Open a command prompt in administrator mode at *<DCC_HOME>* and run `runServices.bat`. The Metadata, Core Notification, and MailService services must now be running.
4. To verify, open the following URL in the browser: `http://serviceregistry:8500/ui` or `https://<hostname>:8501/ui`.
5. Start the Tomcat application server.

**Note:** All the deployed Documentum Connector for Core Share services, such as Core Notification service, Metadata, MailService, Syncagent Robotic, Syncnshare-Manual, must be listed on the Consul UI.

To stop the Consul, Metadata, Core Notification, and MailService services, open a command prompt in administrator mode at *<DCC_HOME>* and run `stopServices.bat`.

## 2.7 Completing the deployment

1. If the machine uses a proxy to connect to Internet, perform the following:

   - Add the following system variables under the environment variables:

     ```
     http.proxyHost= <the host name/IP of the proxy server>
     http.proxyPort= <the port number>
     https.proxyHost=<the host name/IP of the proxy server>
     https.proxyPort=<the port number>
     https.nonProxyHosts=<the host name/the IP of the VM>|serviceRegistry
     http.nonProxyHosts=<the host name/the IP of the VM>|serviceRegistry
     ```

   - Configure the following settings in the Tomcat application server, `$CATALINA_HOME/conf/catalina.properties` file:

     ```
     http.proxyHost= <the host name/IP of the proxy server>
     http.proxyPort= <the port number>
     https.proxyHost=<the host name/IP of the proxy server>
     https.proxyPort=<the port number>
     http.nonProxyHosts=<the host name/the IP of the VM>|serviceRegistry
     https.nonProxyHosts=<the host name/the IP of the VM>|serviceRegistry
     ```

2. Start the instance of the application server.

3. When startup is complete, access the syncagent URL with `https://<host>:<port>/syncagent`. If the deployment is successful, the Documentum Connector for Core Share Administrator page appears.

   Before deploying the application, it is recommended to change the Administrator sign-in password. For details, see "Modifying the Administrator sign-in password" on page 26.

   **Note:** This step is applicable only for robotic folder synchronization and not applicable for client file synchronization.

## 2.8 Modifying the Administrator sign-in password

The Documentum Connector for Core Share contains the username and an encrypted password in the `OT_USER_DATA` table. The default username and password are `Administrator` and `Administrator` respectively. You can modify the Administrator sign-in password within the Documentum Connector for Core Share database.

1. In the top-level syncagent directory, which includes `syncagent-encryption-<version>.jar`, run the following command to encrypt a password for the Administrator:

   ```
   java -cp syncagent-encryption-<version>.jar
   com.emc.syncplicity.connector.util.encryption.PasswordUtils <password>
   ```

   Where <*password*> is the preferred Administrator password.

   Copy the encrypted value to use in the step 2.

2. Run the following query to update the `OT_USER_DATA` database table and to change the default password:

- For Microsoft SQL

  ```
  UPDATE OT_USER_DATA SET USER_PASSWORD = '<YOUR ENCRYPTED PASSWORD HERE>'
  GO
  ```

- For Oracle

  ```
  UPDATE OT_USER_DATA SET USER_PASSWORD = '<YOUR ENCRYPTED PASSWORD HERE>';
  commit;
  ```

- For PostgreSQL

  ```
  UPDATE OT_USER_DATA SET USER_PASSWORD = '<YOUR ENCRYPTED PASSWORD HERE>';
  COMMIT;
  ```

## 2.9 Creating OAuth Client credentials

You will need the OAuth client credentials to configure Documentum Connector for Core Share.

1. Sign in to the OpenText Core Share tenant account that will be used with Documentum Connector for Core Share.
2. Select **Security** > **OAuth Confidential Clients** > **Create Confidential OAuth client**.
3. Specify **Client Description**.
4. Specify the **Redirect URLs** in the following format:

   ```
   https://<host>:<port>/syncagent/oauthcallback
   ```

   The *<host>*:*<port>* refers to the location where `syncagent.war` is deployed, with *<port>* being the Connector port specified in the `server.xml` file when the service is added to the application server.
5. Click **Create**.
6. Copy the client secret that is displayed and save for later.

   **Note:** The client secret information will not be available after you click **OK**.

   The Client ID and Redirect URLs are listed with the Client description under **Existing OAuth Clients**.
7. Sign out from OpenText Core Share.

## 2.10 Configuring Documentum Connector for Core Share

This section describes how to start the Documentum Connector for Core Share Administrator and configure the OAuth authentication for OpenText Core Share.

The administrator must configure the Source and Target repositories on the Syncagent robotic UI. The Source repository is configured in OpenText Documentum CM and the Target repository is configured in OpenText Core Share.

1. Access the following URL, `https://<host>:<port>/syncagent` to start the Documentum Connector for Core Share Administrator:

   The *`<host>`*: *`<port>`* refers to the location where `syncagent.war` is deployed, with *<port>* being the Connector port specified in the `server.xml` file when the service is added to the application server.

2. In the Documentum Connector for Core Share **Sign In** window, provide the default credentials:

   - **Username**: Administrator
   - **Password**: Administrator

   You can change the Administrator sign-in password. For details, see “Modifying the Administrator sign-in password” on page 26.

3. On the **Global Settings** tab, enter the OpenText Documentum CM superuser credentials for Source repository. The Connector requires the superuser account to publish files from the repository. Documentum Connector for Core Share supports both Basic and OTDS authentication types to register the Source repository with Documentum CM Server.

   a. Click **Edit**.

   b. Based on the authentication type, specify the following information in the **Repository Account Credentials** dialog box:

      - Basic authentication type:

        – Documentum CM Server superuser **Sign-in name**

        – Documentum CM Server superuser **Password**

        – Basic **Auth Type** from the list

        – **Documentum URL**: Foundation REST API URL

        – **Repository** from the list (populated based on the specified OpenText Documentum CM URL)

        **Note:** The superuser account must belong to the group `dmc_sync_users`. If the account does not belong to the `dmc_sync_users` group, then add the user to the group using the Documentum Administrator client.

- OTDS authentication type: This is the default authentication type.
  - OTDS user **Sign-in name**
  - OTDS user **Password**
  - OTDS **Auth Type** from the list
  - **OTDS URL**: `http://`*`<host>`*`:`*`<port>`*`/otdsws/oauth2/token`
  - **Client ID** created in OTDS

    **Note:** Register Documentum Connector for Core Share on OTDS and generate Client ID. For example, `dcc_oauth_client`. Documentum Connector for Core Share supports grant type as `password`. To achieve this, clear the **Confidential** check box while creating Client ID on OTDS. For more information, see *OpenText Directory Services - Installation and Administration Guide (OTDS-IWC)*.
  - **Documentum URL**: OTDS-enabled Client REST API URL
  - **Repository** from the list (populated based on the specified OTDS-enabled Client REST API URL)

  **Note:** Client REST API must be pointing to same repository while upgrading from Basic to OTDS authentication type.

c. Click **Save**.

4. On the **Global Settings** tab, enter the **OpenText tenant account** information for the target repository.

   a. Specify the OpenText Core Share Endpoint URL, Client ID, and Client secret generated in OpenText Core Share. You can create the client credentials using "Creating OAuth Client credentials" on page 27.

   b. Click **Sign In**.

   c. Specify the OpenText Core Share tenant account credentials in the OpenText Core Share sign in screen.

   **Note:** Use the OpenText Core Share Enterprise Edition account. Ensure that it is an account dedicated for use by the Documentum Connector for Core Share. After you specify the details for OpenText Core Share, do not change the OpenText Core Share tenant account credentials. Else, Documentum Connector for Core Share will not be able to properly synchronize content from the repository to OpenText Core Share.

   d. Click **Sign In**. You will now be redirected to Documentum Connector for Core Share on successful sign in.

5. On the **Profiles** tab, create the profiles you need. The profiles determine what type of content will be published (maximum size, object types, formats and whether renditions are checked for format).

   For an explanation of how to create a profile, click the **Help** link and navigate to the Profiles topic.

   A profile can be associated with one or more folders.

6. On the **Folders** tab, click **New**. In the opened dialog box, select a repository folder, enter any name you choose to be the name of the OpenText Core Share folder and then assign a profile. The OpenText Core Share folder is created automatically by the Connector. For details on creating a folder, click the **Help** link and navigate to the Folders topic.

   Repeat this step for every repository folder to be shared with OpenText Core Share users.

## 2.11 Upgrading Documentum Connector for Core Share

If you have an earlier version of Documentum Connector for Core Share deployed on your system, follow this procedure to deploy the latest version of Documentum Connector for Core Share.

**Notes**

- Upgrade requires supported Foundation REST API and Documentum CM Server.
- Ensure that the following parameter `rest.paging.max.size=10000` is set in the `rest-api-runtime.properties` file located at `<application-server>\webapps\<dctm-rest>\WEB-INF\classes`.

1. Stop the application server where the Documentum Connector for Core Share is deployed.
2. Take a backup of the existing Documentum Connector for Core Share application, which is the `syncagent` folder with the `database.properties` file.
3. Take a backup of the existing Documentum Connector for Core Share database.
4. Download and extract the latest Documentum Connector for Core Share binaries.
5. From the resulting directory, extract the `syncagent-<version>.zip` file.
6. In the resulting directory, go to the `\deployment\supported-databases\` folder, and then open the subfolder (oracle, sqlserver, or postgresql) that matches your database.
7. Run the following SQL scripts available in `\deployment\supported-databases\` folder accordingly:

- For Oracle:
  `upgrade_quartz_tables_oracle.sql`
  `upgrade_connector_tables_oracle.sql`
  `DCC_Maintenance_oracle.sql`
- For Microsoft SQL:
  `upgrade_quartz_tables_sqlserver.sql`
  `upgrade_connector_tables_sqlserver.sql`
  `DCC_Maintenance_sqlserver.sql`
- For PostgreSQL:
  `upgrade_quartz_tables_postgre.sql`
  `upgrade_connector_tables_postgres.sql`
  `DCC_Maintenance_postgre.sql`

8. Extract `syncagent.war`.
9. In the WAR file, replace the `database.properties` file in the `\WEB-INF\classes\config\` folder with the backup taken in step 2.
10. Archive `syncagent.war`.
11. If you are using Tomcat application server:
    a. Remove the existing `syncagent` folder and `syncagent.war` within `$CATALINA_HOME\webapps\` folder.
    b. Copy the newly packaged `syncagent.war` into `$CATALINA_HOME\webapps\`.
12. To upgrade microservices, complete the instructions in "Upgrading microservices" on page 33.
13. Proceed to "Completing the deployment" on page 26.
14. On the **Global Settings** tab, enter the Documentum CM Server superuser credentials for Source repository. The Connector requires the superuser account to publish files from the repository. Documentum Connector for Core Share supports both Basic and OTDS authentication types to register source repository with Documentum CM Server.
    a. Click **Edit**.
    b. Based on the authentication type, specify the following information in the **Repository Account Credentials** dialog box:
       - Basic authentication type:
         – Documentum CM Server superuser **Sign-in name**
         – Documentum CM Server superuser **Password**
         – Basic **Auth Type** from the list
         – **Documentum URL**: Foundation REST API URL
         – **Repository** from the list (populated based on the specified Documentum URL)

**Note:** The superuser account must belong to the group `dmc_sync_users`. If the account does not belong to the `dmc_sync_users` group, then add the user to the group using the Documentum Administrator client.

- OTDS authentication type: This is the default authentication type.
  - OTDS user **Sign-in name**
  - OTDS user **Password**
  - OTDS **Auth Type** from the list
  - **OTDS URL**: `http://`*`<host>`*`:`*`<port>`*`/otdsws/oauth2/token`
  - **Client ID** created in OTDS

    **Note:** Register Documentum Connector for Core Share on OTDS and generate Client ID. For example, `dcc_oauth_client`. Documentum Connector for Core Share supports grant type as `password`. To achieve this, clear the **Confidential** check box while creating Client ID on OTDS. For more information, see *OpenText Directory Services - Installation and Administration Guide (OTDS-IWC)*.
  - **Documentum URL**: OTDS-enabled Client REST API URL
  - **Repository** from the list (populated based on the specified OTDS-enabled Client REST API URL)

  **Note:** Client REST API must be pointing to same repository while upgrading from Basic to OTDS authentication type.

c. Click **Save**.

**Note:** You must configure the repository in the **Global Settings** tab with the Foundation REST API URL of the preferred repository to avoid the repository unavailability error.

15. You can then create OAuth client credentials and configure connector. Refer to "Creating OAuth Client credentials" on page 27 and "Configuring Documentum Connector for Core Share" on page 28.

## 2.12 Upgrading microservices

1. Open a command prompt in administrator mode at *<DCC_HOME>* and run `stopServices.bat` to stop the current Consul, Metadata, and Core Notification microservices.
2. Take a backup of all contents within the folder *<DCC_HOME>*.

   **Note:** This data is required later in this procedure.

3. From the downloaded Documentum Connector for Core Share binaries, copy `MetadataService`, `Consul`, `CoreNotificationService`, `MailService`, `runConsulService.bat`, `runServices.bat`, `stopServices.bat`, and `uninstallServices.bat` from the extracted binary to *<DCC_HOME>*. Remove the old files from the directory.
4. Open the *<DCC_HOME>*`\Consul\basic_config.json` file for the `http` protocol or *<DCC_HOME>*`\Consul\basic_config_https.json` file for the `https` protocol and enter the IP address of the virtual machine to the *`bind_addr`* parameter.
5. Open a command prompt in administrator mode at *<DCC_HOME>* and run `runConsulService.bat`. The Consul service must now be running.
6. To verify, open the following URL in the browser: `http://serviceregistry:8500/ui` or `https://<hostname>:8501/ui`.

   **Note:** `http://serviceregistry:8500/ui` is only supported in Google Chrome.

   If an error occurs, you can check the logs in *<DCC_HOME>*`\Logs`.
7. Open a command prompt in administrator mode at *<DCC_HOME>* and run `runServices.bat`. The Consul service must now be running.
8. To verify, open the following URL in the browser: `http://serviceregistry:8500/ui` or `https://<hostname>:8501/ui`.

   **Note:** `http://serviceregistry:8500/ui` is only supported in Google Chrome.

   If an error occurs, you can check the logs in *<DCC_HOME>*`\Logs`.
9. Go to the `MetadataService\deployment\`*`<database>`* folder and run one of the following scripts:
   - For Oracle: `upgrade_metadata_tables_oracle.sql`
   - For Microsoft SQL: `upgrade_metadata_tables_sqlserver.sql`
10. Copy the `database.properties` file from your backup and replace it in the *<DCC_HOME>*`\MetadataService\Config\` folder.

11. Provide a valid value for `app.servicePort` in the `service-registry-location.properties` file.

    ```
    serviceregistry-fqdn = <serviceregistry>
    app.servicePort = <valid port number>
    serviceregistry-port=<valid port number>
    httpsenabled=false
    server-host=<hostname of the server required only for https protocol>
    ```

    **Notes**

    - The default port value is `9000`, which can be changed if required.
    - The value for `app.servicePort` must be different for all the applicable services.
    - For the `http` protocol, *<serviceregistry-fqdn>* is the same name as service registry and for the `https` protocol, the certificate CN name must be provided.
    - By default, the `http` protocol is enabled and the `serviceregistry-port` for `http` is `8500`. To use the `https` protocol, set `httpsEnabled` to `true`, `serviceregistry-port` as `8501`, and provide the host name of the server.
    - When all the services are deployed on the same system, the service registry and server host share the same host name. If the services are deployed on distributed systems, the server host is the host name of the system on which Documentum Connector for Core Share is deployed.

12. Edit the `log4j2.xml` file in the same location to configure the Metadata service logs location and log level. By default, it is configured to create `metadataService.log` in the *<DCC_HOME>*`\MetadataService\dcc-logs` folder.

13. Go to the `CoreNotificationService\deployment\`*<database>* folder and run one of the following scripts:

    - For Oracle: `upgrade_CoreNotification_tables_oracle.sql`
    - For Microsoft SQL: `upgrade_CoreNotification_tables_sqlserver.sql`
    - For PostgreSQL: `upgrade_core_notifications_postgres.sql`

14. Copy the `database.properties` file from your backup and replace it in the *<DCC_HOME>*`\CoreNotificationService\Config\` folder.

15. Go to the *<DCC_HOME>*`\CoreNotificationService\Config\` folder and edit the `ExtApi.config` file.

    - Set the robotic and/or manual service as follows:

      – To use only robotic service, comment the `dcc-manual-service` tag.

        **Example for `http` Robotic service:**

        `#dcc-manual-service = syncnshare-manual`

        `dcc-robot-service = syncagent`

        **Example for `https` Robotic service:**

`dcc-secure-robot-service = syncagent`

`#dcc-secure-manual-service = syncnshare-manual`

- To use only manual service, comment the `dcc-robotic-service` tag.

  **Example for `http` Manual service:**

  `dcc-manual-service = syncnshare-manual`

  `#dcc-robot-service = syncagent`

  **Example for `https` Manual service:**

  `#dcc-secure-robot-service = syncagent`

  `dcc-secure-manual-service = syncnshare-manual`

- To use both robotic and manual services, ensure that both tags are enabled.

  **Example for `http`:**

  `dcc-manual-service = syncnshare-manual`

  `dcc-robot-service = syncagent`

  **Example for `https`:**

  `dcc-secure-robot-service = syncagent`

  `dcc-secure-manual-service = syncnshare-manual`

  Where, `syncnshare-manual` is the name of the manual share application deployed in Tomcat and `syncagent` is the name of the robotic application deployed in Tomcat.

  **Notes**

  - When using the `http` protocol, ensure to comment the `https` related parameters and alternatively, comment the `http` related parameters when using the `https` protocol.
  - If the environment is SSL enabled, then enable the proxy and provide the `proxyhost`, `proxyport`, and `noproxy` parameters in the environment variable of the system.

    ```
    https.proxyHost
    https.proxyPort
    https.nonProxyHosts
    ```

    The `nonproxyHosts` inputs must be separated by pipe operator `|`.

- Provide the host name of the server for `serviceregistry`.

  ```
  #Externalizing serviceregistry
  serviceregistry = <Provide the hostname of the server>
  ```

  **Note:** For the `http` protocol, *<serviceregistry>* is the same name as service registry and for the `https` protocol, the certificate CN name must be provided.

- Set the `http` or `https` protocol in the environment variable of the system.

By default, the `http` protocol is enabled and the `serviceregistry-port` for `http` is `8500`. To use the `https` protocol, set `httpsEnabled` to `true`, `serviceregistry-port` as `8501`, and provide the host name of the server.

> **Note:** When all the services are deployed on the same system, the service registry and server host share the same host name. If the services are deployed on distributed systems, the server host is the host name of the system on which Documentum Connector for Core Share is deployed. If you are using the `https` protocol, ensure the OpenText Core Share certificate is added to both the Java cacerts and truststore.

```
#healthcheck properties
httpsEnabled=false
serviceregistry-port=<valid port number>
server-host=<hostname of the server required only for https protocol>
```

16. Edit the `service-registry-location.properties` file and provide the following inputs.

```
app.servicePort = <valid port number>
serviceregistry-port=<valid port number>
httpsEnabled=false
server-host=<hostname of the server required only for https protocol>
```

> **Notes**
>
> - By default, the port value is `9001`, which can be changed if required.
> - By default, the `http` protocol is enabled and the `serviceregistry-port` for `http` is `8500`. To use the `https` protocol, set `httpsEnabled` to `true`, `serviceregistry-port` as `8501`, and provide the host name of the server.
>
>   When all the services are deployed on the same system, the service registry and server host share the same host name. If the services are deployed on distributed systems, the server host is the host name of the system on which Documentum Connector for Core Share is deployed. If you are using the `https` protocol, ensure the OpenText Core Share certificate is added to both the Java cacerts and truststore.

17. Go to the *<DCC_HOME>*`\MailService\config\` folder and edit the `mailservice.properties` file to configure the mail server details.

    Mail service supports both basic and OAuth authentication type. Provide the authentication type as basic or OAuth accordingly.

```
authenticationType= <oauth/basic>
```

- If authentication type is OAuth, then provide the following details:

```
Below details are applicable for Microsoft Exchange oauth changes.
# Provide the Service Provider name, default is Microsoft.
  serviceProvider= Microsoft
# Provide the tenant ID from the Microsoft Entra admin center.
  tenantId = <tenant id>
# Provide the client ID from the Microsoft Entra admin center.
  clientId = <client id>
# If authenticationType is oauth, then provide the user's encrypted client
secret from the Microsoft Entra admin center after completing the app
```

```
registration.
  clientSecret= <client secret>
# below 2 fields tokenUrl and graphApiUrl need not be changed
  tokenUrl= https://login.microsoftonline.com/{tenantId}/oauth2/v2.0/token
  graphApiUrl = https://graph.microsoft.com/v1.0/users/{sender-email}/sendMail
```

- If authentication type is basic, then provide the following details:

```
## Below details need to be provided if authenticationType is basic
#Provide the host, port, user email, and password for the SMTP server as
required. Uncomment the usermail and
 mailPassword fields if authentication is needed.
 smtpmailhost = <SMTP server Host when Authentication Mode is basic>
 smtpmailport = <SMTP server Port when Authentication Mode is basic>
#usermail=<Provide the usermail  if required for the SMTP server>
#mailpassword=<Provide the encrypted mail Password if required for the SMTP
server>
 starttls=false
```

**Note:** Run the following command to encrypt the client secret using `syncagent-encryption-<version>.jar` available in the Documentum Connector for Core Share binary.

```
java -cp syncagent-encryption-<version>.jar
com.emc.syncplicity.connector.util.encryption.PasswordUtils <client secret>
```

Where <*client secret*> is the client secret generated in the Microsoft Entra admin center. For more information, see "Prerequisites:" on page 18 step 3.

18. Edit the `service-registry-location.properties` file and provide the following inputs.

```
serviceregistry-fqdn = <serviceregistry>
app.servicePort = <valid port number>
serviceregistry-port=<valid port number>
httpsenabled=false
server-host=<hostname of the server required only for https protocol>
```

**Notes**

- The default port value is `9000`, which can be changed if required.
- For the `http` protocol, <*serviceregistry-fqdn*> is the same name as service registry and for the `https` protocol, the certificate CN name must be provided.
- By default, the `http` protocol is enabled and the `serviceregistry-port` for `http` is `8500`. To use the `https` protocol, set `httpsEnabled` to `true`, `serviceregistry-port` as `8501`, and provide the host name of the server.
- When all the services are deployed on the same system, the service registry and server host share the same host name. If the services are deployed on distributed systems, the server host is the host name of the system on which Documentum Connector for Core Share is deployed.

19. Open the `mail-details.properties` file from the `Tomcat\webapps\syncagent\WEB-INF\classes\config\` location and edit the following:

```
#from=<email ID of the sender>

#to= <email id of the "to" recepient>

#bcc=<email id of the "bcc" recepient>

#cc=<emaild id of the "cc" recepient>

#subject= <Subject line for the email to be sent>
```

**Note:** If mail service is enabled, step 19 is mandatory to trigger the email on sync failure.

20. Open a command prompt in administrator mode at *<DCC_HOME>* and run `runServices.bat`. The Metadata and Core Notification microservices must now be running.

21. To verify, open the following URL in the browser: `http://serviceregistry:8500/ui` or `https://<hostname>:8501/ui`.

**Note:** `http://serviceregistry:8500/ui` is only supported in Google Chrome. If an error occurs, you can check the logs in *<DCC_HOME>*`\Logs`.

22. Start the Tomcat application server.

## 2.13 Configuring Collaboration Services for client

Prerequisite:

The tenant service user must be created for the Collaboration Services to work with Documentum Connector for Core Share.

### Creating tenant service user

1. Sign in to OpenText Core Share tenant account that will be used with Documentum Connector for Core Share.

2. Select **Security** > **OAuth Confidential Clients** > **Create Tenant Service User**.

3. Specify the following details:

   - **Tenant Service User Name**: Name visible to users on shared content.
   - **Tenant Service User Description**: Description of what this user is used for.
   - **OAuth Client Id**: Client ID retrieved while creating OAuth credentials. You can create the client credentials using "Creating OAuth Client credentials" on page 27.

4. Click **Create**.

### Deploying syncnshare-manual-<*version*>.war on application server

This procedure describes how to set up credentials and configure the database on an application server.

1. If not done already, download and extract the contents of `syncnshare-manual-<version>.zip` to the application server.
2. In the resulting directory, go to the `\deployment\supported-databases` folder, and then open the subfolder (oracle, sqlserver, or postgresql) that matches your database.
3. Run the following SQL scripts available in `\deployment\supported-databases\` folder accordingly:
   - For Oracle:
     `connector_tables_oracle.sql`
     `quartz_tables_oracle.sql`
   - For Microsoft SQL:
     `connector_tables_sqlserver.sql`
     `quartz_tables_sqlserver.sql`
   - For PostgreSQL:
     `connector_tables_postgres.sql`
     `quartz_tables_postgres.sql`

   **Notes**

   You must run these SQL script files before starting the Connector.

   - These scripts create necessary tables required by the Connector.
   - For Microsoft SQL, the `DCC_Syncagent_Manual` database is created by default when you run these scripts.
4. In the top-level `syncnshare-manual` directory, which includes `syncnshare-manual-encryption-<version>.jar`, run the following command to encrypt a password for the database:

```
java -cp syncnshare-manual-encryption-<version>.jar
com.emc.syncplicity.connector.util.encryption.PasswordUtils <password>
```

   Where **<password>** is the password that you will need to use in the step 5 to connect to the database.

   Copy the encrypted value to use in the step 5.
5. Configure `syncnshare-manual.war` as follows:
   a. Extract `syncnshare-manual.war`.
   b. In the WAR file, go to `\WEB-INF\classes\config` and edit `database.properties` as follows:
      - Set **jdbc.username** to a valid database user.

- Set **jdbc.password** with the encrypted value. For more information, see step 4
- Uncomment the **jdbc.dbSchemaName**, **jdbc.driverClassName**, and **jdbc.url** values associated with the required database.

For example, if using Oracle:

```
#jdbc.dbSchemaName=oracle
#jdbc.driverClassName=oracle.jdbc.driver.OracleDriver
#jdbc.url=jdbc:oracle:thin:@//:1521/<db name>
#quartz.dbDelegateClass=org.quartz.impl.jdbcjobstore.oracle.OracleDelegate
```

For example, if using Microsoft SQL Server:

```
#jdbc.dbSchemaName=sqlserver
#jdbc.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDriver
#jdbc.url=jdbc:sqlserver://;databaseName=<db name>
#quartz.dbDelegateClass=org.quartz.impl.jdbcjobstore.MSSQLDelegate
```

For example, if using Amazon Aurora or on-premises PostgreSQL Server:

```
#jdbc.dbSchemaName=postgres
#jdbc.driverClassName=org.postgresql.Driver
#jdbc.url=jdbc:postgresql://<DBHOST>:<PORT>/<DBNAME?
escapeSyntaxCallMode=callIfNoReturn
#quartz.dbDelegateClass=org.quartz.impl.jdbcjobstore.PostgreSQLDelegate
```

c. If necessary, change **jdbc.url** to a value appropriate for your database instance. For example, the URL must not include localhost if the database was on a different machine.

d. Create tenant service user for Collaboration Services to work with Documentum Connector for Core Share. “Creating tenant service user“ on page 38 provides more information.

e. In the WAR file, go to `\WEB-INF\classes\config` and specify the following parameters in the `setup.properties` file:

- `sourceRepoName`: Specify the repository name.
- `sourceRepoUser`: Specify the superuser repository user name.
- `sourceRepoPassword`: Specify the superuser repository encrypted password. To encrypt password, see step 4.
- `sourceRepoUrl`: Specify the Client REST API URL.
- `targetRepoUrl`: Specify the OpenText Core Share Endpoint URL.
- `targetRepoClientId`: Specify the Client ID created by the OpenText Core Share Admin. “Creating OAuth Client credentials” on page 27 provides more information.
- `targetRepoClientSecret`: Provide the OpenText Core Share encrypted Client Secret. To encrypt password, see step 4.

For example:

```
sourceRepoName = CS73
sourceRepoUser =  Administrator
sourceRepoPassword =  AAAAEKy3h8Qj3P7hdnI4+yOPoFdgKRbr+Qxvr4Ep11TOuzTY
sourceRepoUrl = http://localhost:8080/d2rest
```

```
targetRepoUrl = https://core.opentext.com
targetRepoClientId =  b495510e-fcbd-443d-b338-e795bf59bfea
targetRepoClientSecret =  AAAAENV8os1hYxWNLtU7D7dlo5tRKybzXmKOm/
2TKdnL9kZO2G5yF4tNpuUAxKFky1MDOB1VTDgMyRVQVO4RhwAhnGM=
```

f. Archive `syncnshare-manual.war`.

g. Navigate to `\webapps\syncnshare-manual\WEB-INF\classes\config` and edit the `service-registry-location.properties` file as follows:

By default, the `http` protocol is enabled and the `serviceregistry-port` for `http` is `8500`. To use the `https` protocol, set `httpsEnabled` to `true`, `serviceregistry-port` as `8501`, and provide the host name of the server.

**Note:** When all the services are deployed on the same system, the service registry and server host share the same host name. If the services are deployed on distributed systems, the server host is the host name of the system on which Documentum Connector for Core Share is deployed.

```
serviceregistry-fqdn = <serviceregistry>

#Server port
server-port=<server port>
server-protocol=<http/https>
server-host=<hostname of the server required only for https protocol>
serviceregistry-port=<valid port number>
```

**Note:** For the `http` protocol, <*serviceregistry-fqdn*> is the same name as service registry and for the `https` protocol, the certificate CN name must be provided.

6. If using Tomcat as the application server, perform the following tasks:

   a. Copy `syncnshare-manual.war` into `$CATALINA_HOME\webapps`.

   b. Ensure that the application server is configured for UTF encoding by setting **URIEncoding="UTF-8"** as an attribute with the <*connector*> element in the `server.xml` configuration. If there are multiple <*connector*> elements, ensure to change the element used by `syncnshare-manual.war`.

   **Tip:** The <*connector*> element used by syncagent would specify an `http` or `https` protocol, such as `8080` or `8443` respectively.

7. Deploy microservices. See “Deploying microservices” on page 18.

8. After deploying microservices, proceed to “Completing the deployment” on page 26.

### Syncnshare-manual service <version> API support for OpenText Directory Services enabled client

Documentum Connector for Core Share 24.2 and later versions API supports OpenText Directory Services (OTDS) enabled Client REST API and its Cross-Site Request Forgery (CSRF) component.

1. For OTDS and CSRF related configuration, set the following parameters for the `rest-api-runtime.properties` file located at `<TOMCAT>\webapps\d2-rest\WEB-INF\classes\`.

   - For OTDS:

     ```
     rest.security.auth.mode=ct-otds_token
     rest.security.otds.login.url=http://<otds_host>:<port>/otdsws/login?
     response_type=token&client_id=
     <oauth client created in otds>
     ```

   - For `Syncnshare-manual`:

     ```
     rest.d2.core.oauth2.callback.uri=/ui/core-login-cb
     rest.security.crypto.key.salt=abc
     rest.security.client.token.timeout=360000
     ```

   - For CSRF:

     ```
     rest.security.csrf.enabled=true
     ```

2. Optional Generate the token from OTDS by providing the following parameters in the API testing application (for example, Postman):

   **Note:** This step is optional if you are connecting through OpenText™ Documentum™ Content Management Smart View.

   a. Select `POST` method and set `http://<OTDS-HOST-IP>:<port>/otdsws/oauth2/token` as the location.

   b. In the **Body** tab, select **x-www-form-urlencoded**, and add the following **Keys** and **Values**:

      - **username**: Specify the OTDS user name.
      - **password**: Specify the OTDS user password.
      - **client-id**: Specify the OAuth Client ID created in OTDS. For more information, see *OpenText Directory Services - Installation and Administration Guide (OTDS-IWC)*.

        **Note:** Register Documentum Connector for Core Share on OTDS and generate Client ID. For example, `dcc_oauth_client`.

      - **grant_type**: Specify the OAuth Client ID secret. Documentum Connector for Core Share supports grant type as `password`. To achieve this, clear the **Confidential** check box while creating Client ID on OTDS. For more information, see *OpenText Directory Services - Installation and Administration Guide (OTDS-IWC)*.

      The response is as follows:

      ```
      {
                "access_token": “<Access Token>”,
           "refresh_token": "<Refresh Token>",
                "token_type": "Bearer",
           "expires_in": 3600
      }
      ```

3. `Syncnshare-manual` *<version>* APIs are in support with OTDS and CSRF.

   a. Deploy `Syncnshare-manual` on Tomcat. For more information on deploying `syncnshare-manual`, see "Configuring Collaboration Services for client" on page 38.

   b. Start Tomcat.

   c. In the API testing application (for example, Postman) select the `POST` method and set `http://`*`<Host Ip of syncnshare manual Machine>`*`:`*`<port>`*`/syncnshare-manual/v1/share` as the location.

   d. In the **Authorization** tab, set `Basic Auth` as the authentication type and provide the following parameter:

      - **Username**: Administrator
      - **Password**: *<Encrypted password for Administrator>*

   e. In the **Body** tab, select **raw**, **Json**.

```
{
  "sourceObjectId": "<object id>",
  "sourceRepoName": "<repository-name>",
  "sourceShareType": "Publish",
  "shareName":"<share-name>",
  "sourceRepoUser": "<OTDS user created in the partition of OTDS server >",
  "sourceAccessToken": "{{documentumToken}}",
  "targetAccessToken": "{{coreAccessToken}}",
  "csrfHeader": "{{csrfHeader}}",
  "csrfToken": "{{csrfToken}}"
  }
```

   Pre-script request:

```
pm.sendRequest({
    url: "<core Share url>/otdsws/oauth2/token",
    method: "POST",
    header: {
        "Content-Type": "application/x-www-form-urlencoded"
    },
    body: {
        mode: "urlencoded",
        urlencoded: [
            { key: "grant_type", value: "password" },
            { key: "client_id", value: "<Core Share Client ID> " },
            { key: "client_secret", value: "<Client Secret>" },
            { key: "username", value: "<Username>" },
            { key: "password", value: "<Password>" }
        ]
    }
},function (err, res) {
   pm.environment.set("coreAccessToken", res.json().access_token);
});
pm.sendRequest({
  url: "http://<Host>:<Port>/d2-rest/repositories/<Repo-name>",
  method: 'GET',
  header: {
    'Authorization': '<Bearer Access Token generated from OTDS user> ',
  }
},function (err, res) {
  var headerArray = JSON.parse(JSON.stringify(res))["header"];
  for (var i in headerArray) {
     if (headerArray[i].key == 'Set-Cookie') {
         pm.environment.set("documentumToken", headerArray[i].value);
         // break;
                 }
                 if (headerArray[i].key == 'DOCUMENTUM-CSRF-HEADER-NAME') {
                         pm.environment.set("csrfHeader",
headerArray[i].value);

                     }

                     if (headerArray[i].key == 'DOCUMENTUM-CSRF-TOKEN') {

                             pm.environment.set("csrfToken",
headerArray[i].value);

                        }
                     }
                 });
```

The response is as follows:

```
Status: 201 created
{
        "reason": "File shared successfully",
   "name": "test.docx.zip",
        "id": 2455017741585744002
}
```

Chapter 3

# Deploying Documentum Connector for Core Share on Oracle Linux

## 3.1 Prerequisites

1. Refer to the "Prerequisites" on page 9 section in "Deploying Documentum Connector for Core Share on Windows" on page 9 and ensure to adhere to all the points mentioned in this section.

2. Download and extract `OpenText Documentum CM - APIs and Dev Tools` *<version>* `linux.zip` from My Support. The `OpenText Documentum CM - APIs and Dev Tools` *<version>* `linux` folder includes the latest Documentum Connector for Core Share binaries. Extract the Documentum Connector for Core Share binaries to a folder on your local Oracle Linux machine.

3. Create the following directory where you want to deploy Documentum Connector for Core Share and grant full permissions (Read, Write, Execute):

   *<DCC_HOME>*

4. Add the following command to the `\etc\environment` file:

   `export DCC_HOME=`*<DCC_HOME>*

5. Add the following host entry at `\etc\hosts`:

   `localhost serviceRegistry`

   Example: `127.0.0.1 serviceRegistry`

6. Mail service must be registered on Microsoft Entra admin center before deploying the Documentum Connector for Core Share application. For more information, see "Prerequisites:" on page 18 step 3

## 3.2 Deploying microservices

Ensure that you deploy consul, metadata, core notification, and mail service microservices on the same machine as `syncagent.war`.

### 3.2.1 Deploying Consul

1. From the downloaded Documentum Connector for Core Share binaries, copy the `Consul` file located at `\Consul\Linux\` to the `\usr\bin` location in the Oracle Linux machine.
2. Run the following command to grant user permissions to the `consul` file:

   ```
   chmod +x \usr\bin\consul
   ```
3. Create the following directories at `usr\local`:

   *<DCC_HOME>*`\consul_Dir`

   *<DCC_HOME>*`\consul_config`

   *<DCC_HOME>*`\log`
4. From the downloaded Documentum Connector for Core Share binaries, copy one of the following file to the *<DCC_HOME>*`\consul_config` location:
   - `basic_config.json`—to configure the `http` protocol.
   - `basic_config_https.json`—to configure the `https` protocol.
5. Configure the protocol-specific settings:
   - `http`: Open the *<DCC_HOME>*`\Consul\basic_config.json` file and provide the Documentum Connector for Core Share home path in the *`data_dir`* parameter and the server IP address of the virtual machine to the *`bind_addr`* parameter.

     ```
     {
         "data_dir": "<Path to DCC_HOME>/Consul_data_dir",
         "log_level": "INFO",
         "ui":true,
         "bootstrap":true,
         "bind_addr":"IP ADDRESS TO BE FILLED HERE ",
         "client_addr":"0.0.0.0",
         "server": true
     }
     ```
   - `https`: Open the *<DCC_HOME>*`\Consul\basic_config_https.json` file, and provide the Documentum Connector for Core Share home path in the *`data_dir`* parameter, the server IP address of the virtual machine to the *`bind_addr`* parameter, and enter the following *`tls`* parameters:
     - `key_file`: Path of the `crt.key` file. For example: `\opt\cert\cert.key`
     - `cert_file`: Path of the `crt.pem` file. For example: `\opt\cert\crt.pem`
     - `ca_file`: Path of the `ca.crt` file. For example: `\opt\cert\ca.crt`

     ```
     {
         "data_dir": "<Path to DCC_HOME>/Consul_data_dir",
         "log_level": "INFO",
         "ui":true,
         "bootstrap":true,
         "bind_addr":"IP ADDRESS TO BE FILLED HERE ",
         "client_addr":"0.0.0.0",
     ```

```
    "ports": {
      "http": -1,
      "https": 8501
    },
    "tls": {
      "defaults": {
        "key_file": "Path to cert.key file to be provided here"
         "cert_file": "Path to cert.pem file to be provided here"
        "ca_file": "Path to ca.crt file to be provided here"
        "verify_incoming": false,
        "verify_outgoing": false,
        "verify_server_hostname": false
      }
}
```

6. Open the `<DCC_HOME>\Consul\ConsulService` file and configure the `http` or `https` protocol.

   By default, the `http` protocol is selected. When using the `http` protocol, comment the `https` related parameters and alternatively, comment the `http` related parameters when using the `https` protocol.

   The parameters are as follows:

   - `http`: `<!--<arguments>agent -enable-script-checks -config-file="<Path to DCC_HOME>/Consul/basic_config.jason"</arguments>-->`
   - `https`: `<arguments>agent -enable-script-checks -config-file="<Path to DCC_HOME>/Consul/basic_config_https.jason"</arguments>`

```
<Service>
    <id>ConsulService</id>
    <name>ConsulService</name>
    <description>Consul windows service</description>
    <executable>consul</executable>
<!-- Please use the below line when enabling http-->
    <!--<arguments>agent -enable-script-checks -config-file="<Path to DCC_HOME>/
Consul/basic_config_https.jason"</arguments>-->
<!-- Please use the below line when enabling https-->
    <arguments>agent -enable-script-checks -config-file="<Path to DCC_HOME>/Consul/
basic_config_https.jason"</arguments>
    <logpath><Path to DCC_HOME>/Logs/consulService</logpath>
    <logmode>rotate</logmode>
<Service>
```

7. Go to the `/usr/bin` location and run the following command to start consul:

   `consul agent -enable-script-checks -config-file=/<DCC_HOME>/consul_config/basic_config.json -log-file=/<DCC_HOME>/log/`

### 3.2.2 Deploying Metadata service

1. From the downloaded Documentum Connector for Core Share binaries, copy the `MetadataService` folder to the `\<DCC_HOME>\` location in the Oracle Linux machine.

2. Go to the `\<DCC_HOME>\MetadataService\config` folder.

3. Edit the `database.properties` file to connect to the required database as follows:

   - Set **jdbc.username** to valid database user.
   - Set **jdbc.password** with the encrypted value. For more information, see step 4
   - Uncomment the **jdbc.dbSchemaName**, **jdbc.driverClassName**, and **jdbc.url** values associated with the required database.

   For example, if using Oracle:

   ```
   #jdbc.dbSchemaName=oracle
   #jdbc.driverClassName=oracle.jdbc.driver.OracleDriver
   #jdbc.url=jdbc:oracle:thin:@//:1521/<db name>
   #quartz.dbDelegateClass=org.quartz.impl.jdbcjobstore.oracle.OracleDelegate
   ```

   For example, if using Microsoft SQL Server:

   ```
   #jdbc.dbSchemaName=sqlserver
   #jdbc.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDriver
   #jdbc.url=jdbc:sqlserver://;databaseName=<db name>
   #quartz.dbDelegateClass=org.quartz.impl.jdbcjobstore.MSSQLDelegate
   ```

   For example, if using Amazon Aurora or on-premises PostgreSQL Server:

   ```
   #jdbc.dbSchemaName=postgres
   #jdbc.driverClassName=org.postgresql.Driver
   #jdbc.url=jdbc:postgresql://<DBHOST>:<PORT>/<DBNAME?
   escapeSyntaxCallMode=callIfNoReturn
   #quartz.dbDelegateClass=org.quartz.impl.jdbcjobstore.PostgreSQLDelegate
   ```

4. Go to `\<DCC_HOME>\MetadataService` and run the following command to start MetadataService:

   ```
   java -Dlog4j.configurationFile=file:"./config/log4j2.xml" -jar
   metadata-1.0.0-SNAPSHOT.jar
   ```

### 3.2.3 Deploying Core Notification service

1. From the downloaded Documentum Connector for Core Share binaries, copy the `CoreNotificationService` folder to the `\<DCC_HOME>\` location in the Oracle Linux machine.

2. Go to the `\<DCC_HOME>\CoreNotificationService\config\` folder.

3. Edit the `database.properties` file to connect to the required database as follows.

   - Set **jdbc.username** to valid database user.
   - Set **jdbc.password** with the encrypted value. For more information, see step 4
   - Uncomment the **jdbc.dbSchemaName**, **jdbc.driverClassName**, and **jdbc.url** values associated with the required database.

   For example, if using Oracle:

   ```
   #jdbc.dbSchemaName=oracle
   #jdbc.driverClassName=oracle.jdbc.driver.OracleDriver
   #jdbc.url=jdbc:oracle:thin:@//:1521/<db name>
   #quartz.dbDelegateClass=org.quartz.impl.jdbcjobstore.oracle.OracleDelegate
   ```

   For example, if using Microsoft SQL Server:

   ```
   #jdbc.dbSchemaName=sqlserver
   #jdbc.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDriver
   #jdbc.url=jdbc:sqlserver://;databaseName=<db name>
   #quartz.dbDelegateClass=org.quartz.impl.jdbcjobstore.MSSQLDelegate
   ```

   For example, if using Amazon Aurora or on-premises PostgreSQL Server:

   ```
   #jdbc.dbSchemaName=postgres
   #jdbc.driverClassName=org.postgresql.Driver
   #jdbc.url=jdbc:postgresql://<DBHOST>:<PORT>/<DBNAME?
   escapeSyntaxCallMode=callIfNoReturn
   #quartz.dbDelegateClass=org.quartz.impl.jdbcjobstore.PostgreSQLDelegate
   ```

4. Edit the `ExtApi.config` file and provide the following inputs:

   - Set the robotic and/or manual service as follows:

     – To use only robotic service, comment the `dcc-manual-service` tag.

       **Example for `http` Robotic service:**

       `#dcc-manual-service = syncnshare-manual`

       `dcc-robot-service = syncagent`

       **Example for `https` Robotic service:**

       `dcc-secure-robot-service = syncagent`

       `#dcc-secure-manual-service = syncnshare-manual`

     – To use only manual service, comment the `dcc-robotic-service` tag.

       **Example for `http` Manual service:**

`dcc-manual-service = syncnshare-manual`

`#dcc-robot-service = syncagent`

**Example for `https` Manual service:**

`#dcc-secure-robot-service = syncagent`

`dcc-secure-manual-service = syncnshare-manual`

– To use both robotic and manual services, ensure that both tags are enabled.

  **Example for `http`:**

  `dcc-manual-service = syncnshare-manual`

  `dcc-robot-service = syncagent`

  **Example for `https`:**

  `dcc-secure-robot-service = syncagent`

  `dcc-secure-manual-service = syncnshare-manual`

  Where, `syncnshare-manual` is the name of the manual share application deployed in Tomcat and `syncagent` is the name of the robotic application deployed in Tomcat.

  **Notes**

  ○ When using the `http` protocol, ensure to comment the `https` related parameters and alternatively, comment the `http` related parameters when using the `https` protocol.

  ○ If the environment is SSL enabled, then enable the proxy and provide the `proxyhost`, `proxyport`, and `noproxy` parameters in the environment variable of the system.

  ```
  https.proxyHost
  https.proxyPort
  https.nonProxyHosts
  ```

  The `nonproxyHosts` inputs must be separated by pipe operator `|`.

• Provide the host name of the server for `serviceregistry`.

  ```
  #Externalizing serviceregistry
  serviceregistry = <Provide the hostname of the server>
  ```

  **Note:** For the `http` protocol, <*serviceregistry*> is the same name as service registry and for the `https` protocol, the certificate CN name must be provided.

• Set the `http` or `https` protocol in the environment variable of the system.

  By default, the `http` protocol is enabled and the `serviceregistry-port` for `http` is `8500`. To use the `https` protocol, set `httpsEnabled` to `true`, `serviceregistry-port` as `8501`, and provide the host name of the server.

  **Note:** When all the services are deployed on the same system, the service registry and server host share the same host name. If the services are deployed on distributed systems, the server host is the host name of the system on which Documentum Connector for Core Share is deployed. If you are using the `https` protocol, ensure the OpenText Core Share certificate is added to both the Java cacerts and truststore.

```
#healthcheck properties
httpsEnabled=false
serviceregistry-port=<valid port number>
server-host=<hostname of the server required only for https protocol>
```

5. Go to `\<DCC_HOME>\CoreNotificationService` and run the following command to start Core Notification:

   ```
   java -Dlog4j.configurationFile=.\config \log4j2.xml" -jar
   CoreNotificationService.jar
   ```

## 3.2.4 Deploying Mail Service

1. From the downloaded Documentum Connector for Core Share binaries, copy the `MailService` folder to the `\<DCC_HOME>\` location in the Oracle Linux machine.

2. Go to the `\<DCC_HOME>\MailService\config` folder.

3. Edit the `mailservice.properties` file to configure the mail server details.

   Mail service supports both basic and OAuth authentication type. Provide the authentication type as basic or OAuth accordingly.

   ```
   authenticationType= <oauth/basic>
   ```

   - If authentication type is OAuth, then provide the following details:

     ```
     Below details are applicable for Microsoft Exchange oauth changes.
     # Provide the Service Provider name, default is Microsoft.
       serviceProvider= Microsoft
     # Provide the tenant ID from the Microsoft Entra admin center.
       tenantId = <tenant id>
     # Provide the client ID from the Microsoft Entra admin center.
       clientId = <client id>
     # If authenticationType is oauth, then provide the user's encrypted client
     secret from the Microsoft Entra admin center after completing the app
     registration.
       clientSecret= <client secret>
     # below 2 fields tokenUrl and graphApiUrl need not be changed
       tokenUrl= https://login.microsoftonline.com/{tenantId}/oauth2/v2.0/token
       graphApiUrl = https://graph.microsoft.com/v1.0/users/{sender-email}/sendMail
     ```

   - If authentication type is basic, then provide the following details:

     ```
     ## Below details need to be provided if authenticationType is basic
     #Provide the host, port, user email, and password for the SMTP server as
     required. Uncomment the usermail and
      mailPassword fields if authentication is needed.
      smtpmailhost = <SMTP server Host when Authentication Mode is basic>
      smtpmailport = <SMTP server Port when Authentication Mode is basic>
     #usermail=<Provide the usermail  if required for the SMTP server>
     #mailpassword=<Provide the encrypted mail Password if required for the SMTP
     server>
      starttls=false
     ```

**Notes**

- Run the following command to encrypt the client secret using `syncagent-encryption-<version>.jar` available in the Documentum Connector for Core Share binary.

  ```
  java -cp syncagent-encryption-<version>.jar
  com.emc.syncplicity.connector.util.encryption.PasswordUtils <client secret>
  ```

  Where <*client secret*> is the client secret generated in the Microsoft Entra admin center. For more information, see "Prerequisites:" on page 18 step 3.

4. Go to `\<DCC_HOME>\MailService` and run the following command to start the Mail Service:

   ```
   java -Dlog4j.configurationFile=".\config\log4j2.xml" -jar
   mailservice-1.0.0-SNAPSHOT.jar
   ```

## 3.2.5 Deploying syncagent

For more information on deploying `syncagent`, refer to the "Deploying on Application Server" on page 15 section in the "Deploying Documentum Connector for Core Share on Windows" on page 9 chapter.

## 3.2.6 Deploying syncnshare-manual

For more information on deploying `syncnshare-manual`, refer to "Configuring Collaboration Services for client" on page 38.

Chapter 4

# Licensing OpenText Documentum CM

OpenText Documentum CM uses OpenText Directory Services (OTDS) to apply the licenses for all the OpenText Documentum CM components. For more information about procuring the license file and configuring OTDS and license, do the following:

- After completing the deployment, see *OpenText Documentum Content Management - Server and Server Extensions Installation Guide (EDCSY250400-IGD)*.
- After completing the upgrade, see *OpenText Documentum Content Management - On-Premises Upgrade and Migration Guide (EDCCS250400-UMD)*.

Chapter 5

# Managing the Documentum Connector for Core Share

This chapter provides the procedures to manage the Documentum Connector for Core Share for both Windows and Oracle Linux platforms.

For illustration, the locations mentioned in the following sections are applicable for Windows platforms (*<DCC_HOME>*).

For Oracle Linux platform, replace the Windows location with the Oracle Linux location: `\`*<DCC_HOME>*`\`

## 5.1 Checking the log

**Metadata services**

To enable the logging for metadata service, prior to deployment, modify `log4j2.xml`. The file can be found at *<DCC_HOME>*`\MetadataService\config`. Change the level of the `com.opentext.dcc.metadata` logger from **info** to **debug** if required. This enables all logging statements.

By default, `log4j2.xml` is configured to log to the file and the console. It is configured to create `metadataService.log` in *<DCC_HOME>*`\MetadataService\dcc-logs`. You can change the name and location of the file by changing the value of `fileName` and `filePattern` in the following line:

```
RollingFile name="RollingFile" fileName="<DCC_HOME>/Logs/metadataService/
metadataService.log" filePattern="<DCC_HOME>/Logs/metadataService/MetadataService%i.log">
```

**Core Notification services**

To enable the logging for Core Notification service, prior to deployment, modify `log4j2.xml`. The file can be found at *<DCC_HOME>*`\CoreNotificationService\config`. Change the level of the `com.opentext.dcc` logger from **info** to **debug** if required. This enables all logging statements.

By default, `log4j2.xml` is configured to log to the file and the console. It is configured to create `CoreNotificationService.log` in *<DCC_HOME>* `\CoreNotificationService\dcc-logs`. You can change the name and location of the file by changing the value of `fileName` and `filePattern` in the following line:

```
RollingFile name="RollingFile" fileName="<DCC_HOME>/Logs/CoreNotificationService/
CoreNotificationService.log" filePattern="<DCC_HOME>/Logs/CoreNotificationService/
CoreNotificationService%i.log">
```

**Mail Service**

To enable the logging for Mail Service, prior to deployment, modify `log4j2.xml`. The file can be found at `<DCC_HOME>\MailService\config`. Change the level of the `com.opentext.dcc` logger from **info** to **debug** if required. This enables all logging statements.

By default, `log4j2.xml` is configured to log to the file and the console. It is configured to create `MailService.log` in `<DCC_HOME>\MailService\dcc-logs`. You can change the name and location of the file by changing the value of `fileName` and `filePattern` in the following line:

```
RollingFile name="RollingFile" fileName="<DCC_HOME>/Logs/MailService/MailService.log"
filePattern="<DCC_HOME>/Logs/MailService/MailService%i.log">
```

**Documentum Connector for Core Share**

To enable signing in Documentum Connector for Core Share, prior to deployment:

- Modify `log4j2.xml` in `connector WAR`. By default, `log4j2.xml` is configured to log to the file and the console.

  It is configured to create `syncagent.log` in `Tomcat $CATALINA_HOME\bin\dcc-logs`. You can change the name and location of the file by changing the `fileName` and `filePattern` for the rolling file appender in the following line:

  ```
  <RollingFile name="RollingFile" fileName="C:/Documentum/logs/syncagent.log"
  filePattern="C:/Documentum/logs/syncagent%i.log">
  ```

- If using syncnshare-manual services, modify `log4j2.xml` in `syncnshare-manual.war`. By default, `log4j2.xml` is configured to log to the file and the console.

  It is configured to create `syncagent-manual.log` in `Tomcat $CATALINA_HOME\bin\dcc-logs`. You can change the name and location of the file by changing the `param name` and `value` for the rolling file appender in the following line:

  ```
  <RollingFile name="RollingFile" fileName="C:/Documentum/logs/syncagent-manual.log"
  filePattern="C:/Documentum/logs/syncagent-manual%i.log">
  ```

The files can be found at `WEB-INF/classes` in the WAR file. Change the level of the **com.emc.syncplicity** logger from **info** to **debug** if required. This enables all logging statements.

To enable the Quartz and third-party loggers, uncomment the related XML tags.

In addition to the `log4j2.xml` file, logging information is also stored in the database. There are two tables that store the synchronization history:

- `EMC_SYNC_SHARE_HISTORY` – stores general info about synchronization session of each Share.
- `EMC_SYNC_TASK_HISTORY` – stores tasks which were executed within particular synchronization session.

You can access log information using SQL scripts. The following lists a few examples:

- Get all activities for a particular object by repository object's chronicle ID (`REPO_OBJ_ID`):

```
SELECT
    T.REPO_OBJ_ID,
    T.REPO_OBJ_PATH,
    T.REPO_OBJ_NAME,
    T.SYNCP_OBJ_ID,
    T.SYNCP_OBJ_PATH,
    T.SYNCP_OBJ_NAME,
    T.OBJECT_TYPE,
    T.OPERATION,
    T.START_TIME,
    T.END_TIME,
    T.RESULT
FROM EMC_SYNC_TASK_HISTORY T WHERE T.REPO_OBJ_ID = '0901e24080010d34'
```

- Get all activities for a particular object in shared folder by repository object name (`T.REPO_OBJ_NAME`) and OpenText Core Share shared folder name (`S.SYNCPOINT_NAME`):

```
SELECT
    T.REPO_OBJ_ID,
    T.REPO_OBJ_PATH,
    T.REPO_OBJ_NAME,
    T.SYNCP_OBJ_ID,
    T.SYNCP_OBJ_PATH,
    T.SYNCP_OBJ_NAME,
    T.OBJECT_TYPE,
    T.OPERATION,
    T.START_TIME,
    T.END_TIME,
    T.RESULT
FROM EMC_SYNC_SHARE_HISTORY S, EMC_SYNC_TASK_HISTORY T
WHERE S.SYNC_SESSION_ID = T.SYNC_SESSION_ID AND
S.SYNCPOINT_NAME = 'MySharedFolder' AND T.REPO_OBJ_NAME = 'abc.txt'
```

- Get all activities in OpenText Documentum CM shared folder specified by path (`S.REPOSITORY_FOLDER_PATH`) and repository name (`S.REPOSITORY_NAME`):

```
SELECT
    T.REPO_OBJ_ID,
    T.REPO_OBJ_PATH,
    T.REPO_OBJ_NAME,
    T.SYNCP_OBJ_ID,
    T.SYNCP_OBJ_PATH,
    T.SYNCP_OBJ_NAME,
    T.OBJECT_TYPE,
    T.OPERATION,
    T.START_TIME,
    T.END_TIME,
    T.RESULT
FROM EMC_SYNC_SHARE_HISTORY S, EMC_SYNC_TASK_HISTORY T
WHERE S.SYNC_SESSION_ID = T.SYNC_SESSION_ID AND
S.REPOSITORY_FOLDER_PATH = '/Cabinet/MySharedFolder' AND
S.REPOSITORY_NAME = 'xcpc5repo'
```

- Get all activities in any synchronized OpenText Documentum CM folder by shared OpenText Documentum CM folder path (`S.REPOSITORY_FOLDER_PATH`), repository name (`S.REPOSITORY_NAME`) and relative OpenText Documentum CM folder path (it is relative to the shared folder):

```
SELECT
    T.REPO_OBJ_ID,
    T.REPO_OBJ_PATH,
    T.REPO_OBJ_NAME,
    T.SYNCP_OBJ_ID,
    T.SYNCP_OBJ_PATH,
    T.SYNCP_OBJ_NAME,
    T.OBJECT_TYPE,
    T.OPERATION,
    T.START_TIME,
    T.END_TIME,
    T.RESULT
FROM EMC_SYNC_SHARE_HISTORY S, EMC_SYNC_TASK_HISTORY T
WHERE S.SYNC_SESSION_ID = T.SYNC_SESSION_ID AND
S.REPOSITORY_FOLDER_PATH = '/Cabinet/MySharedFolder' AND
S.REPOSITORY_NAME = 'xcpc5repo'  AND T.REPO_OBJ_PATH = '\Subfolder'
```

- Get all activities for a shared folder within a specified time interval by OpenText Core Share shared folder name `S.SYNCPOINT_NAME`.

  If using an Oracle database:

```
Oracle SQL:
SELECT
    T.REPO_OBJ_ID,
    T.REPO_OBJ_PATH,
    T.REPO_OBJ_NAME,
    T.SYNCP_OBJ_ID,
    T.SYNCP_OBJ_PATH,
    T.SYNCP_OBJ_NAME,
    T.OBJECT_TYPE,
    T.OPERATION,
    T.START_TIME,
    T.END_TIME,
    T.RESULT
FROM EMC_SYNC_SHARE_HISTORY S, EMC_SYNC_TASK_HISTORY T
WHERE S.SYNC_SESSION_ID = T.SYNC_SESSION_ID AND
S.SYNCPOINT_NAME = 'MySharedFolder' AND
T.START_TIME >= TO_TIMESTAMP('2013-10-29 17:27:03', 'YYYY-MM-DD HH24:MI:SS') AND
T.END_TIME <= TO_TIMESTAMP('2013-11-06 16:20:24', 'YYYY-MM-DD HH24:MI:SS')
```

  If using a Microsoft SQL database:

```
SELECT
    T.REPO_OBJ_ID,
    T.REPO_OBJ_PATH,
    T.REPO_OBJ_NAME,
    T.SYNCP_OBJ_ID,
    T.SYNCP_OBJ_PATH,
    T.SYNCP_OBJ_NAME,
    T.OBJECT_TYPE,
    T.OPERATION,
    T.START_TIME,
    T.END_TIME,
    T.RESULT
FROM EMC_SYNC_SHARE_HISTORY S, EMC_SYNC_TASK_HISTORY T
WHERE S.SYNC_SESSION_ID = T.SYNC_SESSION_ID AND
S.SYNCPOINT_NAME = 'MySharedFolder' AND
T.START_TIME >= CAST('2013-10-29 17:27:03' AS DATETIME2) AND
T.END_TIME <= CAST('2013-11-05 11:20:24' AS DATETIME2)
```

## 5.2 Managing network connections

The Connector requires access to the OpenText Core Share cloud, a database, and OpenText Documentum CM components, including Documentum CM Server and Java Method Server. This section lists the networks connections used by the Connector and how to test for each connection.

The Connector Administrator requires a network connection to the repository. Ensure that the connection broker configured in `dfc.properties` is accessible.

Connector engine uses the CORE cloud network connection. In your web browser, enter https://core.opentext.com. You are redirected to the CORE home page.

## 5.3 Uninstalling Documentum Connector for Core Share

Perform the following to remove the Connector:

1. Use the Connector Administrator to delete all OpenText Core Share folders.
2. Wait until the Connector agent has a chance to clean up information in the repository.
3. Uninstall `syncagent.war` by uninstalling it from the application server.
4. Undo the application server configuration changes which were made to support `syncagent.war`. For example, delete the Tomcat instance.
5. On the Documentum CM Server, remove the `dmc_sync_users` group using the following DQL statement:

   ```
   Drop Group dmc_sync_users
   ```

6. To uninstall the Consul, metadata, and Core notification services, open a command prompt in administrator mode at *<DCC_HOME>* and run `uninstallServices.bat`.
7. Remove the `Documentum Connector for Core Share` folder (*<DCC_HOME>*) from the application server.

This completes the uninstallation.

Chapter 6

# Troubleshooting

| Symptom | Cause | Fix |
|---|---|---|
| When you click on the **New** button in the **Folders** tab of the Documentum Connector for Core Share Admin UI, the page keeps loading. | The DAR deployment was not successful. | 1. Sign in to the Documentum Administrator and create a new group named `dmc_sync_users` under the **User Management** > **Groups**.<br>2. Add the OpenText Documentum CM superuser configured in Documentum Connector for Core Share to the `dmc_sync_users` group. |

| Symptom | Cause | Fix |
|---|---|---|
| In OpenText Core Share, sharing of the intended documents fail if the storage quota has exceeded. | The storage quota in the administrator account has exceeded. | The following indications are received: • The mail notification is sent to the configured Email IDs in Documentum Connector for Core Share with the following: – Subject line: `Alert: Documentum Connector for Core Share Sync Failed - Quota Limit reached for Core Share` – Message: `Hi, your quota limit is reached for Core Share account.` • On exceeding the storage limit in the administrator account of OpenText Core Share, the following log is generated in : – `syncagent.log` `*ERROR: [TasksScheduler_Worker-2]: com.emc.syncplicity.httpclient.webapi.OTCoreHttpClient - 04 Apr 2024 13:33:01,297 - status code: 413, reason phrase: Request Entity Too Large org.apache.hc.client5.http.HttpResponseException: status code: 413, reason phrase: Request Entity Too Large*` – `syncnshareManual.log`: `1 INFO : [http-nio-8080-exec-67]: com.emc.syncplicity.httpclient.webapi.requests.HttpRequest - 04 Mar 2024 16:20:06,575 - HttpStatusCode : 413 2 ERROR: [http-nio-8080-exec-67]: com.emc.syncplicity.connector.share.impl.OTShareOperationsUtil -` |

| Symptom | Cause | Fix |
|---|---|---|
| | | `04 Mar 2024 16:20:06,609 - UploadDocumentRequest failed on CORE Wrong Server Response`<br>`3 ERROR: [http-nio-8080-exec-67]: com.emc.syncplicity.connector.controller.ShareController - 04 Mar 2024 16:20:06,610 - UploadDocumentRequest failed on CORE`<br><br>To resolve this error, you must procure additional storage quota through OpenText Core Share. |
| Consul does not start on Windows and the following error is logged:<br><br>`Error starting agent: error="Failed to start Consul server: Failed to start Raft: fail to open write-ahead-log: failed initializing meta DB: sync Consul_data_dir\raft\wal: Access is denied."` | The `Consul_data_dir` folder does not have `Write` permissions. | Perform the following:<br>1. Provide `Write` permission to `<DCC_HOME>\Consul_data_dir` folder along with all the subfolder present in the `Consul_data_dir` folder.<br>2. Restart Consul. |