OpenText™ Documentum™ Content Management

Workflow Designer Deployment Guide

Deploy and configure Workflow Designer.

EDCPKL250400-IWD-EN-01

OpenText™ Documentum™ Content Management
Workflow Designer Deployment Guide
EDCPKL250400-IWD-EN-01
Rev.: 2025-Nov-24

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

Open Text Corporation

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

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

Table of Contents

Chapter 1

Getting Started

1.1 Overview

Workflow Designer is used to create workflows for an application. A workflow defines a business process that can be used repeatedly within an application.

You can create and run multiple instances of a workflow process within an application at runtime. A workflow consists of one or more *activities* linked together by *flows*. An activity represents a task that a user should perform on the business process objects to pass the document to the next activity in the workflow. For example, reviewing a document, checking the document into the repository, or approving the document. A flow connects two activities and defines the sequence of activities in a business process. It also defines the packages that are exchanged between the activities. A *package* is an object (Sysobject, Content, Folder, or a Cabinet) that is processed during an activity and then passed to the next activity in the business process.

Note: To access the Workflow Designer application, the user should have administrative privileges (`documentum_workflow_designer` role).

1.2 Prerequisites

You should have the following components installed before you install Workflow Designer:

- Application server (third-party component)
- JDK
- Documentum Administrator
- OpenText™ Documentum™ Content Management Server
- OpenText™ Documentum™ Process Engine(PE)
- OTDS

1.3 System requirements

This section provides information about hardware, software, and web browser requirements for Workflow Designer.

Note: For detailed information on hardware requirements, operating system and application server requirements, and web browser requirements, see *OpenText™ Documentum™ Content Management Server Release Notes*.

1.3.1 Updated attributes for REST API properties

Starting with the Workflow Designer 21.4 release, the following attribute names are renamed:

Old attribute name	New attribute name
xcp.security.logon.secured	rest.security.logon.secured
xcp.signon.logout.url	rest.signon.logout.url
xcp.security.logon.audit	rest.security.logon.audit
xcp.security.kerberos.enable.dt.support	rest.security.kerberos.enable.dt.support
xcp.signon.ticket.timeout	rest.signon.ticket.timeout
xcp.signon.logout.url	rest.signon.logout.url

Notes

- The attribute, `xcp.security.logon.challenge.suffix`, has been removed from the Workflow Designer 21.4 release.
- Starting with Workflow Designer release 25.4, the following new attributes are added in the `rest-api-runtime.properties` file:
  - `ui.session.idle.time`: Time in seconds after which the session timeout warning message is shown to the user.
  - `ui.session.timeout`: Time in seconds after which the user is logged out. The user is logged out after the combined duration of *`ui.session.idle.time`* and *`ui.session.timeout`*.
  - `ui.session.heartbeat.interval`: Time interval in seconds after which the heartbeat signal is sent to the application server. For example, if set to 30 seconds (default), the signal is sent every 30 seconds.
  - `offline.help.server.url`: URL for offline Help server.

Chapter 2

Installing Workflow Designer

2.1 Installation overview

Workflow Designer installation involves deploying:

- &g-dcm-pe;: You can install it on OpenText Documentum Content Management (CM) Server.
- Workflow Designer WAR: You should deploy the Workflow Designer WAR file to the Application Server.

2.2 Installing Process Engine

Before installing Process Engine:

- Ensure that the Java Method Server (JMS) service is stopped.
- Ensure that the Documentum CM Server repository is running.

2.2.1 Starting the connection broker and all repositories

1. Select **Start** > **Programs** > **Documentum Server Manager**.
2. In the **Documentum Server Manager** dialog box, click **DocBroker**, select a connection broker, and click **Start**.
3. Click **Docbase**, select a repository, and click **Start**.
4. Click **OK**.

2.2.2 Preparing non-Windows Platforms for Process Engine installation

Before you install Process Engine on Linux platforms, source the script dm_set_server_env.csh or script dm_set_server_env.sh (depending on the shell in which you run), in the owner environment for the Documentum CM Server installation. This sets the environment variables required by the installer. The script is located at `$DM_HOME/bin`. The following example shows the script for a C shell:

source ./dm_set_server_env.csh

The *OpenText Documentum Content Management - Platform and Platform Extensions Installation Guide (EDCSY240400-IGD)* provides more information on manually configuring environment variables.

2.2.3 Installing Process Engine

2.2.3.1 Prerequisites

If you are upgrading Process Engine from previous releases to up to 24.4 release, complete the following steps before proceeding with Process Engine upgrade:

Note: This section is not applicable if you are upgrading the Process Engine to release 25.2 or later.

1. Stop the Java Method Server.
2. Back up the existing `\\Documentum\tomcat\webapps\bpm` folder for future reference.
3. Install the Process Engine.
4. Restart the Java Method Server.

2.2.3.2 Installing Process Engine with backward compatibility

To ensure backward compatibility with Workflow Designer versions 24.4 and 25.2, complete the following steps:

1. Go to `\\Documentum\tomcat x.x.x\webapps\` folder and back up the existing `bpm` folder to a temporary location.
2. Delete the `\\Documentum\tomcat x.x.x\webapps\bpm` folder.
3. Proceed to install the Process Engine version 25.4.

2.2.3.3 Installing Process Engine

1. Download the Process Engine software from the OpenText MySupport (https://support.opentext.com) site.
2. Locate the file for Windows, `Process_Engine_win.zip` and extract the file.

   Extracting the file creates a new folder that stores the installer files. For example, unpacking the `Process_Engine_win.zip` creates a folder named `Process_Engine_win`.
3. Browse to the new folder. For example, open the `Process_Engine_win` folder.
4. Run the installation file, `peSetup.exe`.
5. Read the Software License and Maintenance agreement page.
6. To continue with the installation, click **I accept the terms of the license agreement** and click **Next**.
7. The installer displays a list of repositories where you want to install Process Engine. Click **Next**.

Make sure you have started all the repositories where you want to install Process Engine. If you do not start the repositories, the installation fails when the installer unpacks the DAR files and tries to install them.

8. The system prompts you for the application server credentials. Enter your credentials, confirm, and click **Next**.

9. The installer displays the installation location. To continue with the installation, click **Install**.

   The installer installs Process Engine on all repositories that are served by Documentum CM Server. This installs the `bpm.war` file on Java Method Server (JMS) and installs the DAR files on each repository.

   The Process Engine installer extracts the contents of the `bpm.war` file to `%DOCUMENTUM%\tomcat`*`<version>`*`\webapps\` on the Microsoft Windows platform and creates the following folder structure:

   - `bpm.war` folder containing the web application files
   - `WEB-INF` folder
   - `META-INF` folder

   On the Windows platform, the installer creates the `PE` folder at `%DOCUMENTUM%\`.

10. Click **Done** to close the Process Engine Installation wizard.

11. Restart the Java Method Server.

12. To verify that the Process Engine installation is successful:

    a. To verify whether JMS started successfully, type the following URL into a browser:

       `http://<host>:<port>/bpm/servlet/DoMethod`

    b. To check the Process Engine version, type the following URL into a browser:

       `http://<host>:<port>/bpm/modules.jsp`

       where *<host>* is the host name or IP address of the JMS and *<port>* is the port number of JMS.

    If the installation fails, view the installation error details in the `logs` folder created inside the PE installation folder in the Process Engine installation folder.

Note: If you are installing Process Engine with Vault enabled, use the silent mode installation process. For more information, see "Installing Process Engine in silent mode" on page 10.

2.2.3.4 Installing Process Engine in silent mode

To install the Process Engine in silent mode, perform these steps:

1. Specify the following parameters—username, password, and domain name in the `test.properties` file.

   ```
   INSTALLER_UI=silent
   PROCESS_ENGINE.GLOBAL_REGISTRY_ADMIN_USER_NAME=AdminUserName
   PROCESS_ENGINE.GLOBAL_REGISTRY_ADMIN_DOMAIN=domainName
   PROCESS_ENGINE.SECURE.GLOBAL_REGISTRY_ADMIN_PASSWORD=AdminPassword
   ```

2. Specify the port number and password for the application host server.

   ```
   APPSERVER.SERVER_HTTP_PORT=9080
   APPSERVER.SECURE.PASSWORD=jboss
   ```

3. Specify the location of OpenText Documentum CM or IJMS, domain name, administrator name and password:

   ```
   \\Used for Default JMS / Independent Java Method Server \\ Location of DCTM
   PE.INSTALL_TARGET= C:\\Documentum
   \\Location of IJMS
   PE.INSTALL_TARGET= C:\\IJMS
   PE.FQDN= fully_qualified_domain_name
   PE.DOCBASES_ADMIN_USER_NAME=username
   PE.DOCBASES_ADMIN_USER_PASSWORD=password
   ```

4. If the vault service is enabled, add the following properties:

   ```
   \\ DSIS_URL consists of the ip and port on which the dsis service is running.
   \\ DSIS_TOKEN is the token generated from dsis service
   IS_VAULT_ENABLED=true
   DSIS_URL=http://localhost:8200/dsis
   DSIS_TOKEN=<vault_token>
   ```

5. Invoke the Process Engine silent installation using this command:

   ```


   //For Windows
   peSetup.exe -f c:\test.properties
   //For Linux
   peSetup.bin -f /home/test.properties
   ```

2.2.4 Adding Process Engine to a new repository

If you configure a new repository on Documentum CM Server after installing Process Engine, the repository configuration program automatically installs the DAR files. You do not have to install Process Engine for this new repository.

2.2.5 Installing Process Engine in an environment with multiple Documentum CM Servers

1. Make sure all the connection brokers, Documentum CM Servers, and repositories for which Process Engine should be installed are in the running state.

   "Starting the connection broker and all repositories" on page 7 and the *OpenText Documentum Content Management - Server Administration and Configuration Guide (EDCCS240400-AGD)* explain how to start connection brokers and all the repositories.

2. Perform the following for each Documentum CM Server:

   a. Install Process Engine on all the Documentum CM Servers.

      "Installing Process Engine" on page 8 provides more information.

   b. Configure the Workflow Agent threads according to your sizing requirements.

      Set the Workflow Agent threads to zero for other Documentum CM Servers that are not participating in processing activities.

   c. Configure the Workflow Agents.

      The Workflow Agent controls the execution of automatic activities in a workflow. The *OpenText Documentum Content Management - Server Administration and Configuration Guide (EDCCS240400-AGD)* explains how to configure Workflow Agents.

      For the distributed Documentum CM Server setup, manually approve the OpenText™ Documentum™ Content Management Foundation Java API privileged client on the secondary Documentum CM Server.

2.2.6 Deploying Workflow Designer on application server

You should deploy the Workflow Designer WAR (`DocumentumWorkflowDesigner.war`) file to the application server to complete the Workflow Designer installation. Download the `DocumentumWorkflowDesigner.war` software file from the OpenText MySupport (https://support.opentext.com) site.

2.2.6.1 Prerequisites for deploying on application server

Before deploying Workflow Designer on an application server, complete the following steps:

- Uncomment the following lines in the `DocumentumWorkflowDesigner.war\WEB-INF\classes\rest-api-runtime.properties` file:

```
# OTDS SSO authentication settings
rest.security.auth.mode=otds-basic
rest.security.otds.login.url={otds-idp}/otdsws/login?response_type=token
&client_id={client_id} &client_secret={client_secret}&logon_appname={app name to
display in log on screen}
rest.security.otds.logout.url={otds-idp}/otdsws/logout?client_id={client_secret}
```

- Update the repository details in the `DocumentumWorkflowDesigner.war\WEB-INF\classes\dfc.properties` file.
- If the vault service is enabled, add the following properties to `dfc.properties`:

```
dfc.dsis.enabled=true
dfc.dsis.daemon.url=http://localhost:8200/dsis
dfc.dsis.daemon.token=<VAULT TOKEN ID>
dfc.globalregistry.repository=<docbasename>
dfc.globalregistry.username=<dm_bof_registry>
dfc.globalregistry.password=<VAULT GLOBALREGISTRY PASSWORD/docbasename>
dfc.session.allow_trusted_login=false
```

2.2.6.2 Configuring the WildFly application server

1. Install the WildFly Enterprise application platform server.
2. In the *`<application_server_home>`*`\standalone\configuration\` folder, edit the port information in the `standalone.xml` file and save the `standalone.xml` file.

   Notes

   - Specify the `jboss.management.native.port` port as the port for the WildFly. The default port is 9990.
   - Specify the `jboss.management.http.port` port as the port for the WildFly. The default port is 9990.

3. At the `application_server_home>\bin\` command prompt, run the `adduser.bat` file to create a management user.

```
What type of user do you wish to add?
a) Management User (mgmt-users.properties)
b) Application User (application-users.properties)
(a): a
Enter the details of the new user to add.
Realm (ManagementRealm) :<Enter>
Username : <WildFly_Admin_User_Name>
Password requirements are listed below. To modify these restrictions edit the
adduser.
properties configuration file.
- The password must not be one of the following restricted values {root, admin,
administrator}
- The password must contain at least 8 characters, 1 alphabetic character(s),
1 digit(s), 1 nonalphanumeric
symbol(s)
```

```
- The password must be different from the username
Password :
Re-enter Password :
What groups do you want this user to belong to? (Please enter a comma separated
list, or leave blank for none)
[ ]:
About to add user 'test' for realm 'ManagementRealm'
Is this correct yes/no? yes
Added user 'WildFly_User_Name' to file 'C:\Node2\standalone\configuration\mgmt-
users.properties'
Added user 'WildFly_User_Name' to file 'C:\Node2\domain\configuration
\mgmtusers.properties'
Added user 'WildFly_User_Name' with groups to file 'C:\Node2\standalone
\configuration\mgmt-groups.properties'
Added user 'WildFly_User_Name' with groups to file 'C:\Node2\domain\configuration
\mgmt-groups.properties'
Is this new user going to be used for one AS process to
connect to another AS process?
e.g. for a slave host controller connecting to the master or
for a Remoting connection for server to server EJB calls.
yes/no? yes
To represent the user add the following to the <server-identities definition secret
value="cGFzc3dvcmQ=" />
[dmadmin@APPVM bin]$ ./add-user.sh
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by __redirected.__SAXParserFactory
(file :C:\Node2\standalone\configuration\jboss-modules.jar) to constructor
com.sun.org.apache.xerces.internal.jaxp.SAXParserFactoryImpl()
WARNING: Please consider reporting this to the maintainers of
__redirected.__SAXParserFactory
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective
access operations
WARNING: All illegal access operations will be denied in a future release
What type of user do you wish to add?
a) Management User (mgmt-users.properties)
b) Application User (application-users.properties)
(a): b
Enter the details of the new user to add.
Using realm 'ApplicationRealm' as discovered from the existing property files.
Username : <Wildfly_Admin_User_Name>
Password recommendations are listed below. To modify these restrictions edit the
adduser.
properties configuration file.
- The password should be different from the username
- The password should not be one of the following restricted values {root, admin,
administrator}
- The password should contain at least 8 characters, 1 alphabetic character(s), 1
digit(s), 1 nonalphanumeric symbol(s)
Password :
WFLYDM0101: Password should have at least 1 digit.
Are you sure you want to use the password entered yes/no? yes
Re-enter Password :
What groups do you want this user to belong to? (Please enter a comma separated
list, or leave blank for none)[ ]:
About to add user 'Wildfly_User_Name' for realm 'ApplicationRealm'
Is this correct yes/no? yes
Added user 'Wildfly_User_Name' to file 'C:\Node2\standalone\configuration
\application-users.properties'
Added user 'Wildfly_User_Name' to 'C:\Node2\standalone\configuration
\applicationusers.properties'
Added user 'Wildfly_User_Name' with groups to file 'C:\Node2\standalone
\configuration\application-roles.properties'
Added user 'Wildfly_User_Name' with groups to file 'C:\Node2\standalone
\configuration\application-roles.properties'
Is this new user going to be used for one AS process to connectto another AS
process?

e.g. for a slave host controller connecting to the master or for a
Remoting connection for server to server EJB calls. yes/no? yes
```

```
To represent the user add the following to the <server-identities definition secret
value="cGFzc3dvcmQ="/>
```

4. Before starting the WildFly application in a Windows environment, add the following Java Options properties in the `standalone.conf.bat` file. For Linux, update the `standalone.conf` file.

   a. Configure the Java home variable.

   ```
   JAVA_HOME=<java_21_home>
   ```

   b. Set Java Options.

   ```
   set "JAVA_OPTS=%JAVA_OPTS% -Djboss.modules.system.pkgs=
   $JBOSS_MODULES_SYSTEM_PKGS -Djava.awt.headless=true
   --add-opens=java.base/java.lang=ALL-UNNAMED
   --add-opens=java.base/java.lang.invoke=ALL-UNNAMED
   --add-exports=java.base/sun.security.provider=ALL-UNNAMED
   --add-exports=java.base/sun.security.pkcs=ALL-UNNAMED
   --add-exports=java.base/sun.security.x509=ALL-UNNAMED
   --add-exports=java.base/sun.security.util=ALL-UNNAMED
   --add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED
   -Dprogram.name=$PROGNAME -Xms3072m -Xmx3072m -XX:MetaspaceSize=1024M -
   XX:MaxMetaspaceSize=1024M
   -Djava.locale.providers=COMPAT,SPI --illegal-access=permit"
   ```

5. At the *`<application_server_home>`*`\bin\` command prompt, start the WildFly server with these options:

   ```
   standalone.bat -b <WildFly_HOSTNAME> -bmanagement <WildFly_HOSTNAME>
   where,
   <WildFly_HOSTNAME> refers to machines' Hostname.
   ```

   This option binds the WildFly interfaces to the host name of the machine rather than the IP address of the machine.

6. Log into the admin console of the WildFly server and deploy the `DocumentumWorkflowDesigner.war` file.

2.2.6.3 Configuring the WebSphere Liberty Application Server

1. Install the WebSphere Liberty Application Server.

2. Use the `C:\Program Files\IBM\WebSphere\Liberty_1\bin\server.bat` command to create a server.

3. Update the `server.bat` file.

```
  set JAVA_HOME=<java_21_home>
  set JAVA_ARGS="-Dcom.ibm.jsse2.overrideDefaultTLS=true"
  set "JAVA_OPTS=-Dprogram.name=%PROGNAME% %JAVA_OPTS%"
  set "JDK_JAVA_OPTIONS=-Dprogram.name=%PROGNAME% %JDK_JAVA_OPTIONS%"
  set "JAVA_OPTS=%JAVA_OPTS% --add-opens=java.base/java.lang=ALL-UNNAMED"
  set "JAVA_OPTS=%JAVA_OPTS% --add-opens=java.base/java.io=ALL-UNNAMED"
  set "JAVA_OPTS=%JAVA_OPTS% --add-opens=java.base/java.util=ALL-UNNAMED"
  set "JAVA_OPTS=%JAVA_OPTS% --add-opens=java.base/java.util.concurrent=ALL-UNNAMED"
  set "JAVA_OPTS=%JAVA_OPTS% --add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED"
  set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS% --add-opens=java.base/java.net=ALL-
UNNAMED"
  set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS% --add-opens=java.base/java.lang.ref=ALL-
UNNAMED"
```

```
  set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS% --add-opens=java.naming/
com.sun.jndi.toolkit.url=ALL-UNNAMED"
  set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS% --add-exports=java.base/
sun.security.provider=ALL-UNNAMED"
  set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS% --add-exports=java.base/
sun.security.pkcs=ALL-UNNAMED"
  set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS% --add-exports=java.base/
sun.security.x509=ALL-UNNAMED"
  set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS% --add-exports=java.base/
sun.security.util=ALL-UNNAMED"
  set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS% --add-exports=java.base/
sun.security.tools.keytool=ALL-UNNAMED
  set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS% --add-opens=java.base/java.lang=ALL-
UNNAMED
```

4. Update the `C:\Program Files\IBM\WebSphere\Liberty_1\usr\servers\LibertyWFD\server.xml`. Replace the feature tags as following:

```
//Updated for WFD:
    <featureManager>
              <feature>jakartaee-10.0</feature>
                      <feature>webProfile-10.0</feature>
                        <feature>restConnector-2.0</feature>
    </featureManager>
```

Note: To update the port, change the *httpPort* attribute in the `server.xml` file.

Sample `server.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?><server description="new server">
<!-- Enable features -->
<featureManager>
<feature>jakartaee-10.0</feature>
</featureManager>
<!-- To access this server from a remote client add a host attribute to the
following element, e.g. host="*" -->
<httpEndpoint id="defaultHttpEndpoint" httpPort="8010" httpsPort="9543"
host="csxcpvm"/>
<!--  Automatically expand WAR files and EAR files  -->
<applicationManager autoExpand="true"/>
</server>
```

5. Place the `DocumentumWorkflowDesigner.war` in `C:\Program Files\IBM\WebSphere\Liberty_1\usr\servers\LibertyWFD\dropins`.

6. Log in to Workflow Designer using `http://hostip:port/DocumentumWorkflowDesigner`.

2.2.6.4 Configuring the Tomcat server

1. To configure the Tomcat server as a standalone server, add the following settings in the `<tomcat_home>\bin\catalina.bat`:

```
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS% --add-opens=java.base/java.net=ALL-
UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS% --add-opens=java.base/java.lang.ref=ALL-
UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS% --add-opens=java.naming/
com.sun.jndi.toolkit.url=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS% --add-exports java.base/
sun.security.provider=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS% --add-exports java.base/
sun.security.pkcs=ALL-UNNAMED"
```

```
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS% --add-exports java.base/
sun.security.x509=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS% --add-exports java.base/
sun.security.util=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS% --add-exports java.base/
sun.security.tools.keytool=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS% --add-opens=java.xml.crypto/
com.sun.org.apache.xml.internal.security=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS% --add-opens=java.base/java.lang=ALL-UNNAMED
```

2. To configure the Tomcat server as a service, complete the following steps:

   a. Locate the `<tomcat_home>\bin\Tomcatw.exe`.

   b. Right-click the file, `Tomcatw.exe`, and select **Run as administrator** from the pop-up menu to open the **Properties** dialog box.

   c. Click **Java** tab and enter the following code under the **Java Options** field:

   ```
   --add-opens=java.base/java.net=ALL-UNNAMED
   --add-opens=java.base/java.lang.ref=ALL-UNNAMED
   --add-opens=java.naming/com.sun.jndi.toolkit.url=ALL-UNNAMED
   --add-exports=java.base/sun.security.provider=ALL-UNNAMED
   --add-exports=java.base/sun.security.pkcs=ALL-UNNAMED
   --add-exports=java.base/sun.security.x509=ALL-UNNAMED
   --add-exports=java.base/sun.security.util=ALL-UNNAMED
   --add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED
   --add-opens=java.xml.crypto/com.sun.org.apache.xml.internal.security=ALL-
   UNNAMED
   --add-opens=java.base/java.lang=ALL-UNNAMED
   ```

3. Copy the `DocumentumWorkflowDesigner.war` file to the *`<application_server_home>`*`\webapps\` folder.

2.2.7 Licensing OpenText Documentum CM

OpenText Documentum CM uses OpenText Directory Services (OTDS) to apply licenses for all the OpenText Documentum CM components. For more information about procuring the license file and configuring OTDS and license, see *OpenText Documentum Content Management - Cloud Deployment Guide (EDCSYCD-IGD)*.

2.2.8 Verifying Workflow Designer installation

To verify whether Workflow Designer is installed successfully in the application server, open the following link in a web browser:

```
http://FQDN:<port_number>/DocumentumWorkflowDesigner
```

If you get the login page for Workflow Designer, the installation is successful. To login, use the OTDS user credentials with valid license.

Note: To access the Workflow Designer application, make sure that the `documentum_workflow_designer` role is assigned to the repository user in Documentum Administrator (DA). Only the users who are part of the `documentum_workflow_designer` role can access the application.

2.3 Workflow Designer – Command Line Interface

Download and extract the Workflow Designer ZIP file from OpenText My Support.

This section describes the Workflow Designer CLI commands.

Notes

- If you have enabled vault services in cloud environment, to access Workflow Designer CLI, provide password in the following format: `vault secret/key name`.

2.3.1 Upgrading the system

Before working with Workflow Designer CLI, complete the following steps:

- Perform all the relevant tasks as described in "Licensing OpenText Documentum CM" on page 16.
- Run the `system upgrade` command. For more information, see "System upgrade command" on page 18
- Validate and reinstall the existing processes that are created using previous versions of Workflow Designer in Workflow Designer 25.4.
- Export the validated processes as packages to make those packages compatible with Workflow Designer 25.4.

2.3.2 Configuring Workflow Designer CLI

Configure the `wfd.properties` file to use Workflow Designer CLI commands:

Configuration name	Description
wfd.rest-base-url	The base URL (HTTPS) of the Workflow Designer. Use HTTPS to prevent OpenText Documentum CM user credentials from being visible to the network.
wfd.repo-name	The default target repository name. If specified, commands can read the value `if -Drepo` argument is not provided.
wfd.ssl.truststore.file	Enable this property if HTTPS is enabled on Workflow Designer. Set the file path of the trust store file containing the certificate published by Workflow Designer.
wfd.ssl.truststore.password	Enable this property if HTTPS is enabled on Workflow Designer. Set the password of the trust store file specified in the property `wfd.ssl.truststore.file`

Example

```
wfd.rest-base-url=http://ip:8000/DocumentumWorkflowDesigner
wfd.repo-name=repositoryname
#wfd.ssl.truststore.file=keystore location
#wfd.ssl.truststore.password=kestore password
```

2.3.3 System upgrade command

Use the `upgrade` command to upgrade internal system artifacts after upgrading from the previous version.

Syntax

```
java -jar xcp-designer-cmd-xx.x.jar -Dcmd=upgrade [-Drepo=<repo>] -Duser=<repo user> -
Dpwd=<repo user password> [-Dconfig=<path to wfd.properties file>]
```

Arguments

Argument	Description
Dcmd	Command name.
(Optional) Drepo	Name of the repository to connect. By default, connection is established with the repository configured in `wfd.properties` file.
Duser	Valid OpenText Documentum CM licensed user and member of `documentum_workflow_designer` role.
Dpwd	OpenText Documentum CM password.
(Optional) Dconfig	Path to the configuration file. By default, the `wfd.properties` file in the `execution` directory is used.

Example

```
java -jar %wfd_cli%\xcp-designer-cmd-25.4.jar -Dcmd=upgrade -Drepo=xcprepo -
Duser=dmadmin -Dpwd=password -Dconfig=%wfd_cli%\wfd.properties
```

2.3.4 List project processes command

Use the `list` command to write output to an XML file.

Syntax

```
java -jar xcp-designer-cmd-<version>.jar -Dcmd=list [-Drepo=<repo>] -Duser=<repo user> -
Dpwd=<repo user password> [-Dall=1] [-Dprocess-list-out=<output process list file
location>] [-Dconfig=<path to wfd.properties file>]
```

Arguments

Argument	Description
Dcmd	Command name.

Argument	Description
(optional) Drepo	Name of the repository to connect. By default, connection is established with the repository configured in `wfd.properties` file.
Duser	Valid OpenText Documentum CM licensed user and member of `documentum_workflow_designer` role.
Dpwd	OpenText Documentum CM password.
(Optional) Dall	Specifies whether non-validated processes should be listed. By default, only validated processes are listed.
Dprocess-list-out	Output file where the line-separated list of processes in a project is saved. Otherwise, a file named *<processes_yy:MM:dd:mm:ss>* is created in the `execution` directory.
(Optional) Dconfig	Path to the configuration file. By default, the `wfd.properties` file in the execution directory is used.

Example

```
java -jar %wfd_cli%\xcp-designer-cmd-25.4.jar -Dcmd=list -Drepo=xcprepo -Duser=dmadmin -
Dpwd=password -Dall=1 -Dprocess-list-out=%wfd_test%\process-list-out -Dconfig=%wfd_cli%
\wfd.properties
```

2.3.5 Export project processes command

Use the `pkg-export` command to export processes from a project into an importable package.

Syntax

```
java -jar xcp-designer-cmd-<version>.jar -Dcmd=pkg-export [-Drepo=<repo name>] -
Duser=<repo user> -Dpwd=<repo user password> [-Dall=1] [-Dpackage=<output package file
location>] [-Dprocess-list-in=<selection input file location>] [-Dconfig=<path to
wfd.properties file>]
```

Arguments

Argument	Description
Dcmd	Command name.
(optional) Drepo	Name of the repository to connect. By default, the connection is established with the repository configured in the `wfd.properties` file.
Duser	A valid OpenText Documentum CM licensed user who is a member of the `documentum_workflow_designer` role.

Argument	Description
Dpwd	OpenText Documentum CM user password.
(Optional) Dall	Specifies whether non-validated processes should be listed. By default, only validated processes are listed. Set the value to 1, to list all the validated and non-validated processes.
Dpackage	Destination output package where the processes from a project are exported. By default, processes are exported to a file named `dctmbpm_yy.MM.dd.mm.ss.pkg` in the `execution` directory.
Dprocess-list-in	Input file that contains the line-separated list of processes to export into a package. By default, a file named *<processes_yy:MM:dd:mm:ss>* is created in the `execution` directory.
(Optional) Dconfig	Path to the configuration file. By default, the `wfd.properties` file in the `execution` directory is used.

Example

```
java -jar %wfd_cli%\xcp-designer-cmd-25.4.jar -Dcmd=pkg-export -Drepo=xcprepo -
Duser=dmadmin -Dpwd=password -Dall=1 =Dpackage=%wfd_test%\output_package.pkg -Dprocess-
list-in=%wfd_test%\process-list-in -Dconfig=%wfd_cli%\wfd.properties
```

2.3.6 List package processes command

Use the `pkg-list` command to write a line-separated list of processes in a package to an output file.

Syntax

```
java -jar xcp-designer-cmd-<version>.jar -Dcmd=pkg-list [-Drepo=<repo>] -Duser=<repo
user> -Dpwd=<repo user password> -Dpackage=<input package file location> [-Dall=1] [-
Dprocess-list-out=<output process list file location>] [-Dconfig=<path to wfd.properties
file>]
```

Arguments

Argument	Description
Dcmd	Command name.
(optional) Drepo	Name of the repository to connect. By default, connection is established with the repository configured in `wfd.properties` file.
Duser	Valid OpenText Documentum CM licensed user and member of `documentum_ workflow_designer` role.

Argument	Description
Dpwd	OpenText Documentum CM password.
Dpackage	Input package file path.
(Optional) Dall	Specifies whether non-validated processes should be listed. By default, only validated processes are listed. Set the value to 1, to list all the validated and non-validated processes.
Dprocess-list-out	Specifies the output file where the line-separated list of processes in a package is saved. By default, a file named *<processes_yy:MM:dd:mm:ss>* is created in the `execution` directory.
(Optional) Dconfig	Path to the configuration file. By default, the `wfd.properties` configuration file in the execution directory is used.

Example

```
java -jar %wfd_cli%\xcp-designer-cmd-25.4.jar -Dcmd=pkg-list -Drepo=xcprepo -
Duser=dmadmin -Dpwd=password -Dpackage=%wfd_test%\pkg_all.pkg -Dall=1 -Dprocess-list-out=
%wfd_test%\process-list-out -Dconfig=%wfd_cli%\wfd.properties
```

2.3.7 Import processes from a package to project command

Use the `pkg-import` command to import processes from a package in to a project.

Syntax

```
java -jar xcp-designer-cmd-<version>.jar -Dcmd=pkg-import [-Drepo=<repo>] -Duser=<repo
user> -Dpwd=<repo user password> -Dpackage=<varname>input package file location</
varname> [-Dall=1] [-Dforce=1] [-Dforce-clean=1] [-Dskip-upgrade=1] [-Dprocess-list-
in=<varname>selection input file location</varname>] [-Dconfig=<varname>path to
wfd.properties file</varname>]
```

Notes

- Starting with Workflow Designer release 24.4, the format of the `process-list-in` file is XML. Previous versions of Workflow Designer accept line-separated file.
- Starting with Workflow Designer release 24.4, you can set the `skip-upgrade` flag to skip the system upgrade process. The default value of this attribute is 0.

Arguments

Argument	Description
Dcmd	Command name.

Argument	Description
(optional) Drepo	Name of the repository to connect. By default, the connection is established with the repository configured in the `wfd.properties` file.
Duser	A valid OpenText Documentum CM licensed user who is a member of `documentum_workflow_designer` role.
Dpwd	OpenText Documentum CM user password.
Dpackage	Path to the package file.
Dforce	Uninstall the deployed process and pause its instances to import the new template. If specified, the deployed processes are set to the `draft` state, the process instances are paused, and the new template is imported from the package. Otherwise, the command exits with an error if the existing process is in the deployed state.
Dall	Import the non-validated processes. If specified, the non-validated processes are imported, otherwise, the command exits with an error for the non-validated processes.
Dforce-clean	Purge the currently deployed process and its instances to import the new template. If specified, the deployed processes and instances are deleted and the new template is imported from the package. Otherwise, the command exits with an error if the existing process is in the deployed state.
Dskip-upgrade	Skip the system upgrade process. If specified, system upgrade is performed before importing the processes, otherwise, by default the system upgrade is performed before importing the processes. **Note:** The default value is 0 and indicates that the system is upgraded. You can set the value to 1, to disable automatic upgrade.
Dprocess-list-in	Input file that contains the line-separated list of processes to delete.
(Optional) Dconfig	Path to the configuration file. By default, the `wfd.properties` file in the `execution` directory is used.

Example

```
java -jar %wfd_cli%\xcp-designer-cmd-25.4.jar -Dcmd=pkg-import -Drepo=xcprepo -
Duser=dmadmin -Dpwd=password -Dpackage=%wfd_test%\pkg_validated.pkg -Dforce=1  -Dprocess-
list-in=%wfd_test%\test\process-list-in  -Dall=1 -Dskip-upgrade=0 -Dconfig=%wfd_cli%
\wfd.properties
```

2.3.8 Import and install processes command

Use the `pkg-install` command to import and install processes from a package in to a project.

Syntax

```
java -jar xcp-designer-cmd-<version>.jar -Dcmd=pkg-install [-Drepo=<repo>] -Duser=<repo
user> -Dpwd=<repo user password> -Dpackage=<input package file location> [-Dforce=1] [-
Dforce-clean=1] [-Dskip-upgrade=1] [-Dprocess-list-in=<selection input file location>] [-
Dconfig=<path to wfd.properties file>]
```

Notes

- Starting with Workflow Designer release 24.4, the format of the `process-list-in` file is XML. Previous versions of Workflow Designer accept line-separated file.
- Starting with Workflow Designer release 24.4, you can set the `skip-upgrade` flag to skip the system upgrade process. The default value of this attribute is 0.

Arguments

Argument	Description
Dcmd	Command name.
(optional) Drepo	Name of the repository to connect. By default, the connection is established with the repository configured in the `wfd.properties` file.
Duser	Valid OpenText Documentum CM licensed user and member of `documentum_workflow_designer` role. User should have assigned
Dpwd	OpenText Documentum CM user password.
Dpackage	Path to package file.
Dforce	Uninstall the currently deployed process and pause its instances to import and install the new template. If specified, the deployed processes are set to the `draft` state, the process instances are paused, and the new template is imported and installed from the package. Otherwise, the command exits with an error if the existing process is in the deployed state.

Argument	Description
Dforce-clean	Purge the currently deployed process and its instances to import and install the new template. If specified, the deployed processes and instances are deleted and the new template is imported and installed from the package. Otherwise, the command exits with an error if the existing process is in the deployed state.
Dskip-upgrade	Skip the system upgrade process. If specified, the system upgrade is performed before importing the processes. Otherwise, by default the system upgrade is performed before importing the processes. **Note:** The default value is 0 and indicates that the system is upgraded. You can set the value to 1, to disable the automatic upgrade.

Example

```
java -jar %wfd_cli%\xcp-designer-cmd-25.4.jar -Dcmd=pkg-install -Drepo=xcprepo -
Duser=dmadmin -Dpwd=password -Dpackage=%wfd_test%\pkg_validated.pkg -Dskip-upgrade=0 -
Dprocess-list-in=%wfd_test%\test\process-list-in -Dforce-clean=1 -Dconfig=%wfd_cli%
\wfd.properties
```

2.3.9 Uninstall and delete processes command

Use the `purge` command to uninstall and delete deployed processes from a project.

Syntax

```
java -jar xcp-designer-cmd-<version>.jar -Dcmd=purge [-Drepo=<repo>] -Duser=<repo user> -
Dpwd=<repo user password> [-Dprocess-list-in=<selection input file location>] [-
Dconfig=<path to wfd.properties file>]
```

Arguments

Argument	Description
Dcmd	Command name.
(optional) Drepo	Name of the repository to connect. By default, connection is established with the repository configured in `wfd.properties` file.
Duser	Valid OpenText Documentum CM licensed user and member of `documentum_workflow_designer` role. User should have assigned
Dpwd	OpenText Documentum CM password.

Argument	Description
Dprocess-list-in	Input file that contains the line-separated list of processes to delete.
(Optional) Dconfig	Path to the configuration file. By default, the `wfd.properties` file in the execution directory is used.

Example

```
java -jar %wfd_cli%\xcp-designer-cmd-25.4.jar -Dcmd=purge -Drepo=xcprepo -Duser=dmadmin -
Dpwd=password  -Dprocess-list-in=%wfd_test%\test\process-list-in -Dconfig=%wfd_cli%
\wfd.properties
```

2.4 Troubleshooting SSO

To enable error message logging, configure these packages in the `log4j2.properties` configuration file:

```
# XRest Security
logger.xRestSecurity.name=com.emc.rest.rest.security
logger.xRestSecurity.level=DEBUG
logger.xRestSecurity.additivity = false
logger.xRestSecurity.appenderRef.xRestSecurity.ref=xRestSecurity //create your own
appender

# Documentum Rest
logger.documentumRest.name=com.emc.documentum.rest
logger.documentumRest.level=DEBUG
logger.documentumRest.additivity = false
logger.documentumRest.appenderRef.documentumRest.ref= documentumRest  //create your own
appender

# Spring
logger.spring.name=org.springframework
logger.spring.level=DEBUG
```

2.4.1 Accessing SSO-enabled Workflow Designer application fails

Problem

While accessing the SSO-enabled Workflow Designer application that is configured in the rest-api-runtime properties, you may see the login page instead of being automatically logged into the application using SSO.

Resolution

Verify the parameters in the `rest-api-runtime.properties` file to ensure the system picks up the right version of the rest-api-runtime properties file from the current file path.

2.5 Configuring OpenText Directory Services Authentication in Documentum CM Server

OpenText Directory Services (OTDS) enables identity management and SSO user authentication across OpenText products.

To configure OTDS authentication for Workflow Designer:

1. Make sure that OTDS authentication is already configured in Documentum CM Server.

   Notes

   - OTDS implementation in Workflow Designer does not provide a fallback mechanism.
   - Workflow Designer supports Multi-factor authentication through OTDS. For more information, see *OpenText Directory Services - Installation and Administration Guide (OTDS240400-IWC)*.

2. Workflow Designer supports full-way OTDS SSO authentication over OAuth2 protocol. To configure an OAuth2 client in OTDS, perform the following steps:

   1. Open the *otds-admin* web application in your browser using `http://<OTDS machine>:8080/otds-admin` and log in.
   2. Go to the *otds-admin* web application, click **OAuth Clients** and click **Add**.

      Note: Make sure that the **Confidential** check box is selected.

   3. Click **Next** until the **Redirect URLs** page appears, add `https:\/\/<regular expression for host name and port of AppHost>\/.*\/otds-signin\.jsp` as the redirect URL.

      ```
      Examples:

      http:\/\/10\.194\.52\.246:8000\/.*\/otds-signin\.jsp

      Or,

      https:\/\/appserver\.wfd\.com:8443\/.*\/otds-signin\.jsp
      ```

   4. Click **Save**. The **Client Secret** is generated. Copy the **Client Secret** value.
   5. Generate the OAuth2 access token using **otdsws**:

      ```
      https://<otdsidp>:8443/otdsws/oauth2/auth?
      response_type=token&client_id=<client_id>&client_secret=<client_secret>&redirec
      t_uri=<url encoded value of app host URL /testapp/otds-signin.jsp>
      ```

   6. Log in to OTDS as a user from the partition configured in the OTDS.
   7. Copy the OAuth2 **access_token** from the redirect URL and use it to authenticate on Documentum CM Server. Run the following commands in IAPI

```
API> connect,<repo_name>,<USER_LOGIN_NAME>,dm_otds_oauth=<OAuth2 access token>
```

3. Update the `rest-api-runtime.properties` file available in the `DocumentumWorkflowDesigner/WEB-INF/classes/` folder:

```
rest.security.auth.mode=otds
rest.security.otds.login.url=<otds-idp>/otdsws/login?
               response_type=token&client_id=<client_id>
               &client_secret=<client_secret>
rest.security.realm.name=com.emc.documentum.rest
//Default is 480 minutes

rest.signon.logout.url=/otds-signin.jsp?logout=yes
rest.security.auth.mode=otds-basic
rest.security.realm.name=com.emc.documentum.rest
rest.security.otds.login.url=<otds-idp>/otdsws/login?
response_type=token&client_id=<client_id>&client_secret=<client_secret>&logon_appnam
e=<app_name>rest.signon.logout.url=/otds-signin.jsp?logout=yes
          # # Supported expiration policies for client token cookie are:
          # (1)
com.emc.documentum.rest.security.ticket.impl.HardTimeoutExpirationPolicy
          #     The client token expires after a specified duration. If the REST
client sends a request before the duration, the
          #     REST server accepts the client token. If the REST client sends a
request after the duration, the REST server
          #     rejects the client token, and the client has to authenticate again.
          #
          # (2)
com.emc.documentum.rest.security.ticket.impl.TolerantTimeoutExpirationPolicy
          #     - the client token expires at a date/time for inactivity;
          #     - if the REST client comes before the expiry time, the REST server
accepts the client token;
          #     - if the REST client comes after the expiry time but still within a
tolerant time slot,
          #       the REST server accepts the client token, and renews another
client token with a new lifetime to set back for
          #       next call use;
          #         - if the REST client comes after the expiry time and out of the
tolerant time slot,
          #       the REST server rejects the client token, then the REST client
needs to authenticate again.
          #     - the tolerant time slot is set as the same duration to the expiry
time
          #     The client token expires after two times of the specified duration.
The REST server issues new client tokens under
          #     certain conditions. For details, see the following:
          #     - If the REST client sends a request before the duration, the REST
server accepts the client token.
          #     - If the REST client sends a request after the duration, and before
two times of the duration, the REST server
          #       accepts the client token, and issue another client token with the
same duration to the client for subsequent
          #       requests.
          #     - If the REST client sends a request after two times of the
duration comes to an end, the REST server rejects the
          #       client token, and the client has to authenticate again.
          #
          # (3)
com.emc.documentum.rest.security.ticket.impl.TouchedTimeoutExpirationPolicy
          #     The client token expires after a specified duration. The REST
server issues new client tokens under certain
          #     conditions. For details, see the following:
          #     - If the REST client sends a request before the duration, the REST
server accepts the client token, and issues
          #       another client token with the same duration to the client for
subsequent requests.
          #     - If the REST client sends a request after the duration, the REST
server rejects the client token, and the client
          #       has to authenticate again.
          #
          # By default, the REST server uses the TolerantTimeoutExpirationPolicy
policy.
      #
rest.security.client.token.expiration.policy=
      # This property is used by all expiration policies. The timeout in seconds
for the client token cookie.
          # This property MUST be specified with a non-empty value. The default is
3600 seconds.
      #
rest.security.client.token.timeout=3600
      # Symmetric crypto algorithm that RESTful Services uses to encrypt and
decrypt client tokens, the block cipher mode,
      # and the padding strategy in the following pattern:
      # rest.security.crypto.algorithm=<crypto_algorithm>/<block_cipher_mode>/
<padding_strategy>
      # Each provider MAY have different implementations and limitations. This
property is optional. We do not recommend that
      # you set <padding_strategy> to NoPadding because the CT length is not always
compatible with the block size.
      # Default: AES/CBC/PKCS5Padding
      #
rest.security.crypto.algorithm=AES/CBC/PKCS5Padding
      # Crypto algorithm parameters class name
      # Valid values: javax.crypto.spec.IvParameterSpec, or blank.
      # If the algorithm requires extra parameters, such as initialization vector
(IV), set this property to
      # javax.crypto.spec.IvParameterSpec. Otherwise, leave it blank.
      # For example, if AES/CBC/PKCS5Padding is used, the value MUST be set to
javax.crypto.spec.IvParameterSpec.
      # If AES/ECB/PKCS5Padding is used, the value MUST be blank. This property is
optional for the ECB, CBC, OFB, and CFB
      # modes. The system selects the correct class for these modes.
      #
rest.security.crypto.algorithm.parameters.class=
      # Algorithm to generate the secret key
      # Examples: AES, DESede, DES, RC5, HmacMD5, and Blowfish
      # The key generating algorithm MUST be compatible with the crypto algorithm.
This property is optional.
      # Default: AES
      #
rest.security.key.algorithm=AES
      # Random algorithm name
      # Default: SHA1PRNG
      # You CAN set this property to other random generation algorithms.
      #
rest.security.random.algorithm=SHA1PRNG
      # Crypto provider name
      # Default: empty (the JVM provider)
      # Alternatively, set this property to 'BC' to use the Bouncy Castle provider.
When REST Services uses the Bouncy Castle
      # provider, its library MUST be specified in the class path.
      # To use RSA provider, the value can be set to JsafeJCE. And also please make
sure the RSA libs are in the class path.
      #
rest.security.crypto.provider=
      # Crypto provider class name
      # Default: empty (the JVM provider)
      # It can be com.rsa.jsafe.provider.JsafeJCE for the RSA JsafeJCE provider.
      # Or org.bouncycastle.jce.provider.BouncyCastleProvider for the Bouncy Castle
as well.
      # This property is optional if the provider is RSA JsafeJCE or Bouncy Castle.
      #
rest.security.crypto.provider.class=
      # Key size in bit for the secret key
      # This property MUST be compatible with the crypto algorithm.
      # If you want to use a key size larger than 128 bits, download the "JCE
Unlimited Strength Jurisdiction Policy Files" from
      # the Oracle web site. By default, the key size is 64 for DES, and 128 for
all other algorithms.
      #
rest.security.crypto.key.size=
      # Block size in bit
      # This property MUST be compatible with the crypto algorithm. By default, the
block size is 64 bits for Blowfish, DES,
      # DESede, RC2, and RC5, and 128 bits for all other algorithms.
      #
rest.security.crypto.block.size=
      # Crypto salt for client token encryption and decryption
      # For a multi-node deployment of REST servers, this property MUST be
consistently set across all REST servers. For a
      # single-node deployment of REST servers, this property is optional. The
value CAN be any ascII characters. We recommend
      # that you specify a text no less than 8 characters.
      #
rest.security.crypto.key.salt=
      # Specifies the mode of the JsafeJCE provider.
      # Valid values are:
      #  -    FIPS140_MODE: only FIPS140-approved algorithms and default algorithms
are allowed
      #  -    NON_FIPS140_MODE: all algorithms are allowed
      # When you deploy Documentum REST Services on IBM WebSphere and use CAS or
Kerberos authentication, the value must be
      # set to NON_FIPS140_MODE.
      #
      # This property only takes effect when CryptoJ toolkit in runtime supports to
switch mode.
      #
      # Editing property "com.rsa.cryptoj.fips140initialmode" in Java security
properties file
      # (by default, it is JAVA_HOME/lib/security/java.security) will change the
initial FIPS mode.
      # For example, "com.rsa.cryptoj.fips140initialmode=NON_FIPS140_MODE" will set
the initial mode as NON FIPS.
      #
      # Currently REST will only try to switch mode at runtime when
NON_FIPS140_MODE is specified here no matter what the initial FIPS mode is.
      # When it fails to switch mode, logs are dumped to indicate the failure and
the working FIPS mode is dumped as well.
      #
      # Default value: FIPS140_MODE
      #
rest.security.crypto.provider.jsafejce.mode=FIPS140_MODE
```

Sample configuration:

```
rest.security.otds.login.url=https://10.194.52.116:8080/otdsws/login?
response_type=token&client_id=wfd-
client&client_secret=Gnj3zjTVON8SxyWTiLgoHpfMeakZMOeC
&logon_appname=DocumentumWorkflowDesigner
```

4. Restart the application server on which the Workflow Designer application is deployed.

5. Access the Workflow Designer login page. The user is redirected to the OTDS sign in page.

2.5.1 Skipping SSO authentication

You can skip SSO authentication if the fallback authentication is configured using the *`rest.security.auth.mode`* parameter in the `rest-api-runtime.properties` file. For example:

```
rest.security.auth.mode=otds-basic
```

To skip SSO authentication, you can append the *`skipsso=true`* query parameter to the sign-in page to indicate to the system to use the fallback basic authentication method. For example:

```
https://FQDN:8443/DocumentumWorkflowDesigner/signin.html?skipsso=true
```

2.6 Uninstalling Workflow Designer

To remove Workflow Designer from an application server, refer to the application server documentation for uninstalling the war file.