opentext™

OpenText™ AppWorks Gateway

Installation and Administration Guide

Install and configure the OpenText AppWorks Gateway, and install and deploy AppWorks apps.

OTAG250400-IGD-EN-1

OpenText™ AppWorks Gateway
Installation and Administration Guide
OTAG250400-IGD-EN-1
Rev.: 2025-Sept-04

This documentation has been created for OpenText™ AppWorks Gateway 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

OpenText AppWorks Gateway Overview

The AppWorks Gateway is an Enterprise Application Development and Management platform. The AppWorks Gateway allows you to quickly and easily build purpose-specific apps for the Enterprise using familiar Web technologies. The AppWorks Gateway uses OpenText™ Directory Services to manage your users.

The AppWorks Gateway) engine component is the single point of contact between all clients and your production server, for example, Content Server. It forwards all incoming operation requests, queues data requests to reduce server load, and handles notifications to all registered client applications.

Communication between the AppWorks Gateway engine and your production server is done through HTTP or HTTPS.

During the installation process, you connect the AppWorks Gateway to a database such as Microsoft SQL Server or Oracle, and to OpenText Directory Services (OTDS) to manage and authenticate your users. As a final step, you connect to your production server and install the applications, components, and services in the AppWorks Gateway.

The AppWorks Gateway can also be run in a containerized environment. OpenText provides Docker images that can be downloaded with Docker YAML files and Kubernetes Helm charts and scripts.

Note: This guide does not replace the documentation for OpenText products that include AppWorks Gateway as part of their deployment, such as eDOCS InfoCenter, Content Server Mobile and Tempo Box. The installation guides for these products may include additional or modified steps for installing the AppWorks Gateway. See OpenText My Support (https://knowledge.opentext.com) for further help on each product.

Chapter 2

Installing the AppWorks Gateway

In a production environment, we recommend the following setup, where OTDS, the AppWorks Gateway and the OpenText product are all installed on separate servers.

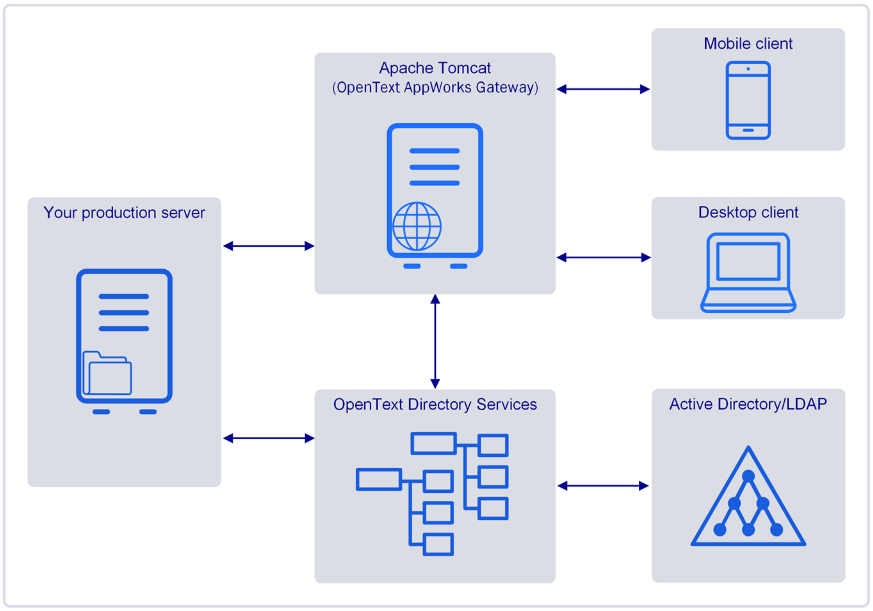


Figure 2-1: OpenText AppWorks Gateway Architecture

Active Directory/LDAP
The Active Directory or LDAP server that you may be using to manage your users. This is optional.

OpenText Directory Services
The OpenText Directory Services server that manages your users.

Your production server
The server hosting an OpenText product, for example, Content Server or eDOCS.

Apache Tomcat
The server hosting Apache Tomcat, with OpenText AppWorks Gateway deployed and configured, and your apps installed and enabled.

Mobile Client
An Apple® or Android™ device.

Desktop Client
A computer running a Windows or macOS® operating system.

2.1 Prerequisites

Before you start, review the *OpenText AppWorks Gateway Release Notes*, available from OpenText My Support (https://knowledge.opentext.com). Search My Support, using the “#AppWorksGateway25.4Documentation” quick link.

The Release Notes contain important information about supported environments and the compatibility of OpenText AppWorks Gateway with OpenText products, such as Content Server.

The OpenText AppWorks Gateway requires the following software.

- Java Development Kit.
- Apache Tomcat

  ! **Important**
  Install the AppWorks Gateway and Apache Tomcat on a dedicated server. Do not install OTDS, or an OpenText product, such as Content Server on the same server as the OpenText AppWorks Gateway.

- A database to connect to. Create a schema and an associated schema admin user in your Microsoft SQL Server, MySQL, Oracle, Oracle RAC, PostgreSQL, or SAP HANA database.
- An OTDS installation. OTDS centralizes user and group identity information and manages single sign on (SSO) between OpenText components.

2.2 Configuring Apache Tomcat for the AppWorks Gateway

After installing Apache Tomcat, in the **Apache Tomcat Properties** dialog, configure the settings in the **Java** tab for the AppWorks Gateway.

To configure Apache Tomcat for the AppWorks Gateway:

1. Ensure you have installed the required software. For details, see “Prerequisites” on page 10.
2. Stop the Apache Tomcat service.
3. Navigate to the <*Tomcat_Home*>`\bin` folder.
4. Double-click the Tomcat executable file, for example, `Tomcat10w.exe` and do the following:

   a. Click the **Java** tab.
   b. In the **Initial memory pool box**, type `0` or leave blank.

   c. In the **Maximum memory pool** box, type `4096`.

   d. Click **OK**.

5. If you are running OpenJDK Java 17, add the following to the **Java 10 Options:** box:

   ```
   --add-opens=java.base/java.net=ALL-UNNAMED
   ```

   Note: To support AppWorks Gateway in an HTTPS environment, add the following:

   ```
   --add-opens=java.base/sun.net.www.protocol.https=ALL-UNNAMED
   ```

6. Start the Apache Tomcat service to make sure your Tomcat installation starts correctly with these new settings.

2.3 Installing the AppWorks Gateway

Before you start, read the "Prerequisites" on page 10 and complete the "Configuring Apache Tomcat for the AppWorks Gateway" on page 10 section.

To install the AppWorks Gateway:

1. Stop the Apache Tomcat service.

2. Navigate to the `<Tomcat_Home>\conf` folder and back up the `server.xml` and `context.xml` files.

   Important
   The OpenText AppWorks Gateway installation package includes modified versions of the `server.xml` and `context.xml` files. If you have customized the `server.xml` or `context.xml` files, and you want to retain those changes, you must add them to the versions of the files supplied by the AppWorks Gateway.

3. Download the AppWorks Gateway installation package from OpenText My Support (https://knowledge.opentext.com). Search My Support, using the "#AppWorks Gateway25.4SoftwareDownloads" quick link.

4. Extract the contents of the `otag-gateway–25.4.zip` file to the root of the Apache Tomcat folder, for example, `C:\Program Files\Apache Software Foundation \Tomcat 10.0`.

5. Optional If you plan on running SSL, comment out or remove the following tag in the `server.xml` file.

   ```
   <Listener className="org.apache.catalina.core.AprLifeCycleListener" SSLEngine="on"/>
   ```

6. Start the Apache Tomcat service.

Tip: Navigate to the <*Tomcat_Home*>`\logs` folder and open the `catalina.`<*yyyy-mm-dd*> file. When you see a line containing INFO: Server startup in <*55278*> ms, you can start to configure the AppWorks Gateway for your preferred database and for OTDS.

2.4 Connecting the AppWorks Gateway to a Database

In a browser, type `http://`<*Tomcat_Host*>`:`<*port*>`/gateway`. Use the IP address or fully qualified domain name of the server on which you are running Apache Tomcat for the AppWorks Gateway. Do not use `http://localhost:8080`.

A configuration page is displayed to enable you to connect AppWorks Gateway to a database.

Important
After you have specified a database and completed the installation, you cannot choose a different database for AppWorks Gateway Therefore, ensure that you are connecting to the database you intend to use for your environment.

To connect to a database:

1. In **Database Configuration**, you connect AppWorks Gateway to a database. The database that you use must be created prior to completing the steps below. Create a blank database and AppWorks Gateway will create the required tables.

2. From the **Database type** drop down, choose from the following database options:

   - **Microsoft SQL Server**

     The minimum roles required for AppWorks Gateway to connect to a Microsoft SQL Server database are:

     – ddladmin role

     – data writer role

     – data reader role

   - **MySQL** – In the MySQL Workbench, select the **Options File** from the **Navigator** panel and choose the **Networking** tab. Locate the `max_allowed_packet_length` variable and change to 1024 MB. You may need to install the JDBC driver (Connector/J) and place it in your <*Tomcat_Home*>`\lib` folder. After adding the driver, restart the Apache Tomcat service.

   - **Oracle** – You may need to download a JDBC driver JAR file and place it in your <*Tomcat_Home*>`\lib` folder. After adding the driver, restart the Apache Tomcat service.

     The minimum privileges required for AppWorks Gateway to connect to an Oracle database are:

  - CONNECT
  - UNLIMITED TABLESPACE
  - CREATE SESSION
  - ALTER SESSION
  - CREATE TABLE

- **Oracle RAC** – You may need to download a JDBC driver JAR file and place it in your <*Tomcat_Home*>`\lib` folder. After adding the driver, restart the Apache Tomcat service.
- **PostgreSQL** – To connect the AppWorks Gateway to a PostgreSQL database, you need to add the following line to the `C:\Program Files\PostgreSQL\11\data\pg_hba.conf` file.

  ```
  # TYPE  DATABASE  USER  ADDRESS   METHOD
    host  all       all   0.0.0.0/0 md5
  ```

  The minimum role required for AppWorks Gateway to connect to a PostgreSQL database is:

  - DBOwner

- **SAP HANA**
- **Custom Connection String**

Caution

The **Custom Connection String** option is intended for experienced database administrators only and must be used with caution.

3. In the **Username** and **Password** boxes, type the credentials for an admin user of your database.
4. When connecting to an Oracle Real Application Clusters database, complete the following:

   - In the **Load Balance** area, choose if you would like to enable client side load balancing.
   - In the **Cluster Host** box, type the virtual cluster name that you chose when you created the RAC database.
   - In the **Port** box, type the port. The default port for Oracle is 1521.
   - In the **Service Identifier** box, type the virtual service name that you chose when you created the RAC database.

   This information is added to the `opentext.properties` file, in the <*Tomcat_Home*>`\conf` folder. The details are encrypted in this file.
5. Click **Next** and wait for a confirmation that the connection is established.

2.5 Connecting the AppWorks Gateway to OTDS

After you have connected AppWorks Gateway to a database, the next step is to connect to OTDS.

Important

- Once you have defined the connection to OTDS, you cannot make changes to it. This means that you cannot switch from one OTDS instance to another.
- We recommend that you do not install OTDS on the same server as AppWorks Gateway. Also, OTDS should be on a separate server to an OpenText product such as Content Server or eDOCS DM.

2.5.1 Connecting to OTDS

To connect to OTDS:

1. In **OTDS Configuration**, you connect AppWorks Gateway to an existing instance of OTDS.
2. Complete the required information, as follows:
   - **Gateway Admin Username** and **Password** – Defines the AppWorks Gateway administrative user account that will be created in Directory Services. The default username is `otag`. The password requirements depend on your OTDS global password policy.
   - **OTDS server URL** – The URL of the Tomcat server that is running OTDS. Do not use `http://localhost:8080`.
   - **OTDS Admin Username** and **Password** – The credentials for an OTDS Admin user.
   - **Resource Name** and **Gateway User Partition Name** – Creates an AppWorks Gateway Resource and User Partition in OTDS with the names provided. The default values are `OpenText AppWorks Gateway` and `otag` respectively.

     Note: If you have other AppWorks Gateway installations that connect to the OTDS service, the values for **Gateway Admin Username**, **Gateway User Partition Name**, and **Resource Name** must be unique for each installation.
3. Click **Save**. The configuration is complete when you are prompted to sign in to the AppWorks Gateway.

   Before you sign in, open a second browser window and go to the Directory Services web client at `http://`*`<Tomcat_Host>`*`:`*`<port>`*`/otds-admin/`.
4. Sign in as the OTDS admin user using “otadmin@otds.admin” as the username and the **OTDS Administrative Account Password** you defined in step 2.
5. From the **SETUP** menu, click **Partitions**.

6. For the **otag** partition, select **View Members** from the **Actions** menu. Next, select the **Groups** tab.
7. Select the `otagadmins` group, then select **Edit Membership**, and click **Add Member**.
8. Search for the `otadmin@otds.admin` member and from the search results box, click **Add Selected** to add the member to the `otagadmins` group.
9. Sign in to the AppWorks Gateway as the AppWorks Gateway administrative user.

   The AppWorks Gateway administrative user is the user you defined in the **Gateway Admin Username** and **Password** fields in step 2.

2.5.2 Deploying the AppWorks Gateway and OTDS in a Non-SSL Environment

Where both OTDS and the AppWorks Gateway are running in a non-SSL environment, you need to make a change in OTDS so that Directory Services no longer checks for SSL.

If you deployed the AppWorks Gateway in a non-SSL environment, you will see the following error in the `gateway.log` file in the <*Tomcat_Home*>`\logs` folder:

```
** HTTP 403 encountered during resource authentication. This is
likely to be because  OTDS is not configured to permit
authentication traffic across unsecured (non-SSL/TLS) transports.
If this is a development environment, you may work around this
security  feature by ensuring OTDS has the System Attribute
"directory.auth.EnforceSSL=false". **
```

To configure OTDS and the AppWorks Gateway for non-SSL environments

1. Sign in to the Directory Services web client at `http://`*`<Tomcat_Host>`*`:`*`<port>`*`/otds-admin/`.
2. From the **SETUP** menu, click **System Attributes**.
3. Click **Add**.
4. Type directory.auth.EnforceSSL in the **Name** field, and false in the **Value** field.
5. Click **Save**.

Chapter 3

Upgrading the AppWorks Gateway

You can upgrade to the latest release of the AppWorks Gateway from all previous versions, starting from 16.*x*. .

Note: AppWorks Gateway requires Apache Tomcat. Check the *OpenText AppWorks Gateway Release Notes* for the version of Apache Tomcat to use. When upgrading to AppWorks Gateway, you must do the following:

1. Install and configure an instance of Apache Tomcat.*x*. See "Configuring Apache Tomcat for the AppWorks Gateway" on page 10.
2. Extract the AppWorks Gateway files into the Apache Tomcat *XX.x* folder. See "Upgrading fromAppWorks Gateway 16.x and later" on page 17.
3. Copy several files from your existing AppWorks Gateway installation to the AppWorks Gateway installation. See "Upgrading fromAppWorks Gateway 16.x and later" on page 17.

3.1 Upgrading fromAppWorks Gateway 16.x and later

Use this procedure to upgrade from an AppWorks Gateway 16.*x* and later to the latest version of AppWorks Gateway.

Note: When upgrading to OpenText AppWorks Gateway, you must decrypt then re-encrypt the `opentext.properties` file in the <*Tomcat_Home*>`\conf` folder. The steps to do this are described in the following procedures.

To upgrade the OpenText AppWorks Gateway:

1. Stop the Apache Tomcat service that will host the new AppWorks Gateway installation,.
2. Back up the database that is associated with the AppWorks Gateway installation. This is the database that you connected to when you installed AppWorks Gateway.
3. Configure Redirect URLs for OAuth clients in OTDS, as follows:

   a. Log in to OTDS as an administrator.
   b. Search for the OAuth client created for the partition during installation.
   c. Click **Properties** from the **Actions** menu option.
   d. Click the **Redirect URLs** tab.
   e. On the button bar click **Add**.

  f. In the **Redirect URLs** text box, type `http.*` and click **Save**.
  g. On the button bar click **Add**.
  h. In the **Redirect URLs** text box, type `https.*` and click **Save**.
  i. On the button bar click **Save**. For further information, see the chapter on OAuth Clients in the *OpenText Directory Services - Installation and Administration Guide (OTDS-IWC)*.

4. If you have made custom changes to the `server.xml` file in the <*Tomcat_Home*>`\conf` folder of your Apache Tomcat 10.*x* instance, make a backup copy of the `server.xml` file.

5. Extract the contents of the `otag-gateway–25.4.zip` file to the root of the Apache Tomcat 10.*x* folder, for example, `C:\Program Files\Apache Software Foundation\Tomcat 10.0`.

6. Go to the <*Tomcat_Home*>`\conf` folder for your previous installation of Apache Tomcat for example, `C:\Program Files\Apache Software Foundation\Tomcat 9.0` and copy the `eula`, `opentext.properties` and `server.xml` files.

7. Paste the `eula`, `opentext.properties` and `server.xml` files into the <*Tomcat_Home*>`\conf` folder of your Apache Tomcat 10.*x* installation, overwriting the existing files.

8. If you have made custom changes to the `server.xml` file in the Tomcat 10.`x` instance, use the `server.xml` back up file to add those changes.

9. Encrypt the `opentext.properties` file in the <*Tomcat_Home*>`\conf` folder, using the Database Encryption Tools available from My Support (https://knowledge.opentext.com/knowledge/llisapi.dll?func=ll&objId=78441633&objAction=browse&viewType=1). The `opentext.properties` file in the <*Tomcat_Home*>`\conf` folder stores the details of the database for the AppWorks Gateway Installation. Proceed as follows:

   - If you know the database details, encrypt the `opentext.properties` file using the 22.3 Database Encryption Tool. See the *Readme* file supplied with the 22.3 Database Encryption Tool for details.
   - If you do not know the database details, decrypt the `opentext.properties` file, using the Database Decryption Tool and re-encrypt the file using the Encryption Tool, as described below.

10. To decrypt and re-encrypt the `opentext.properties` file take the following steps:

    a. Take a backup of the `opentext.properties` file.
    b. Download the Database Decryption and Database Encryption Tools from My Support:

       - If you are upgrading from AppWorks Gateway 16.*x*, use the Database Decryption tool in the 16.7 software folder in My Support. If you are upgrading from 21.*x* or 22.*x*, use the Database Decryption Tool in the 22.3 folder in My Support.

- Use the Database Encryption Tool in the 22.3 software folder in My Support.

c. Decrypt the `opentext.properties` file using the decrypter.jar file in the Database Decryption Tool, as follows:

   i. Expand the `appworks—gateway—`*`<Version No>`*`—decryptor-tool.zip` file.

   ii. Copy the `opentext.properties` file from <*Tomcat_Home*>`\conf` folder to the root of the `AppWorks-Gateway—`*`<Version No>`*`-decryptor-tool` folder.

   iii. Decrypt the `opentext.properties` file using the decrypter jar file:

   ```
   $ java -jar awg-support-tool-decrypter-<Version No>.jar -d
   opentext.properties
   ```

   This decrypts the `opentext.properties` file and outputs a `decrypted.properties` file.

d. Encrypt the `decrypted.properties` file using the 22.3 Database Encryption Tool, as follows:

   i. Expand the `appworks—gateway—22.3—encryptor-tool.zip`file.

   ii. Copy the `decrypted.properties` file to the root of the `AppWorks-Gateway—22—3-encryptor-tool` folder.

   iii. Rename the `decrypted.properties` to `opentext.properties`.

   iv. Encrypt the `opentext.properties` file using the encrypter jar file:

   ```
   $ java -jar awg-support-tool-encrypter-22.3.jar -e opentext.
   properties
   ```

   This encrypts the `opentext.properties` file and outputs an `encrypted.properties` file.

   v. Rename the `encrypted.properties` file to `opentext.properties`.

   vi. Place the `opentext.properties` file in the <*Tomcat_Home*>`\conf` folder, overwriting the existing file.

11. Start the Apache Tomcat service.

12. In a browser, type `http://`*`<Tomcat_Host>`*`:`*`<port>`*`/gateway` and sign in to the OpenText AppWorks Gateway. All of your applications and settings should be present.

3.2 Uninstalling the AppWorks Gateway

This section describes how to uninstall AppWorks Gateway 16.x. The assumptions are that you are removing AppWorks Gateway from a server, and there is no requirement for you to retain any of the components of the installation. You do not need to retain the OTDS partition for the installation or the database that you configured for use with the installation. If you are upgrading the AppWorks Gateway, or the version of Apache Tomcat, see "Upgrading the AppWorks Gateway" on page 17 for details.

Important
The following description does not include any steps to back up the installation you are removing. When you take these steps you are removing the AppWorks Gateway, the **otag** partition in OTDS, and any AppWorks data in the database that you created for use with the installation.

To uninstall AppWorks Gateway:

1. Navigate to the <*Tomcat_Home*> folder and delete the `gateway` folder.
2. Navigate to the <*Tomcat_Home*>`\conf` folder and delete the `opentext.properties` file.
3. Navigate to the <*Tomcat_Home*>`\webapps` folder and contents of the folder, except for the `ROOT` folder.
4. Navigate to the <*Tomcat_Home*>`\temp` folder and delete all of the files in the folder.
5. Navigate to the <*Tomcat_Home*>`\tmp` folder and delete all of the files in the folder.
6. Navigate to the <*Tomcat_Home*>`\lib` folder and delete any OpenText AppWorks Gateway libraries.
7. Navigate to the <*Tomcat_Home*>`\work\Catalina\localhost` folder and delete all of the files in the folder.
8. Sign in to the Directory Services web client at `<http://Tomcat_Host>:<port>/otds-admin/`.
9. Delete the **otag** partition. This is the default partition name. You may have supplied a different name when you installed the OpenText AppWorks Gateway.
10. Delete the database schema that you created for use with the AppWorks Gateway.

Chapter 4

Securing the AppWorks Gateway

Using the AppWorks Gateway in debug mode

The AppWorks Gateway can be run in debug mode. Debug mode allows for more detailed error information to be returned to clients. The AppWorks Gateway should not be left running in debug mode during production use, to avoid the communication of detailed error messages.

Troubleshooting the AppWorks Gateway at debug or trace logging level

When troubleshooting the AppWorks Gateway, you can adjust the logging level to debug or trace. See "Frequently Asked Questions" on page 89.

When debug-level or trace-level logging is enabled, sensitive configuration data may be written to the log files. Ensure that debug-level and trace-level logs are securely disposed of when they are no longer required.

Terminating AppWorks Gateway at the connector layer

Always terminate the AppWorks Gateway with SSL/TLS at the Apache Tomcat connector layer.

Without SSL/TLS at the Tomcat layer, the **Want Secure Cookies** setting will not work correctly. If the **Want Secure Cookies** setting is enabled, cookies issued by the OpenText AppWorks Gateway will contain the "secure" flag if possible, providing SSL/TLS termination is performed on the Tomcat connection. The **Want Secure Cookies** is on the **General Settings** page (**Settings > General Settings**).

Security considerations for Apache Tomcat

Follow industry best practices for securing the Apache Tomcat server. Consider the following resources:

- https://tomcat.apache.org/tomcat-10.0-doc/security-howto.html
- https://www.owasp.org/index.php/Securing_tomcat
- https://geekflare.com/apache-tomcat-hardening-and-security-guide

Protection against Malicious File Uploads

Protection against malicious file upload must use a layered approach – the environment and the application must be protected. Protection should be enforced on the server side and – very importantly - on the client side of the application. At infrastructure level, ensure the following:

- Virus protection is active and up-to-date on all client systems

- Client systems are using a supported browser with the latest updates. Older browsers have fewer protections. To protect against issues with file locking OpenText recommends that you use the following settings:
  - Antivirus scanning is performed “scan-on-write” for files and directories not excluded from antivirus scans.
  - If on-demand or scheduled scanning occurs when the system is operating, files may be locked by the antivirus application. As a result, OpenText recommends that the following folders are not scanned:
    - The External File Store (EFS)
    - Log and tmp folders
    - Folder(s) where the DBMS database and log files are located
    - Search functionality directories
    - The volume in the search administration section.

4.1 Configuring the AppWorks Gateway for HTTPS

OpenText recommends using SSL to increase the security of the system. This section explains how to configure SSL on Apache Tomcat to allow the OpenText AppWorks Gateway to work with the HTTPS protocol. The *OpenText AppWorks Gateway: Security Considerations* Best Practices guide summarizes in a single document the recommendations on security that appear in various places in this guide.

The steps are as follows:

1. Configure Apache Tomcat for HTTPS in Java Options.
2. Create a Java keystore file.
3. Create a Certificate Signing Request (CSR).
4. Sign the CSR File using Active Directory Certificate Services
5. Add the Root CA Certificate to the AppWorks Gateway Server
6. Add the Keystore to the OpenText AppWorks Gateway.

! **Important**

In this procedure we are using Active Directory Certificate Services (ADCS). ADCS allows you to build, in-house, a public key infrastructure, with public key cryptography, digital certificates, and digital signature capabilities, using Microsoft technology. There are many alternative services that you can use, and which may be more appropriate for your organization. The use of ADCS here is for the purpose of providing an “end-to-end” example to explain the procedures for enabling the AppWorks Gateway for HTTPS. There is no requirement to use ADCS. The implementation and maintenance of security protocols at your organization can be complex and should only be attempted by experienced personnel.

4.1.1 Configure Apache Tomcat for HTTPS in Java Options

If you are running OpenJDK Java 17, add the following to the **Java 11 Options** box in the **Apache Tomcat Properties** dialog box.

```
--add-opens=java.base/sun.net.www.protocol.https=ALL-UNNAMED
```

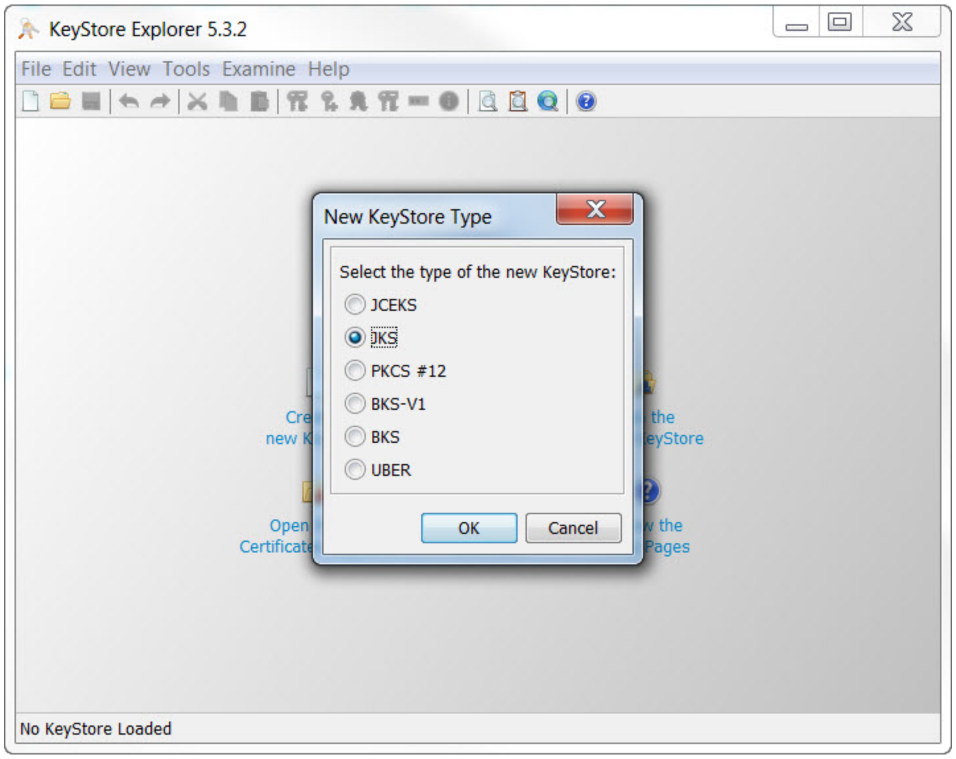
See "Configuring Apache Tomcat for the AppWorks Gateway" on page 10.

4.1.2 Create a Java Keystore File

You can use the Java keytool.exe command line utility to complete these steps. However, to present the required changes as clearly as possible, we are using KeyStore Explorer. KeyStore Explore is an open source utility that provides a graphical user interface for the keytool key and certificate management functionality. KeyStore Explorer is available from http://keystore-explorer.org.

To create a Java keystore file:

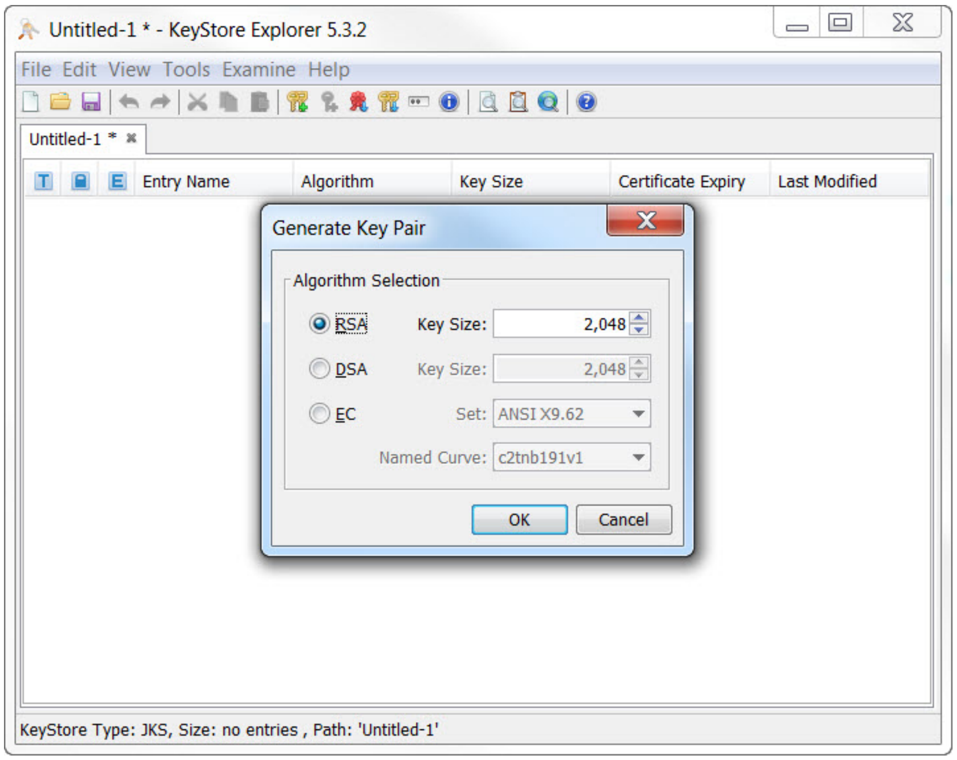
1. Start KeyStore Explorer, click **File > New** and select **JKS** for the keystore type.



Select the type of the new KeyStore:
JCEKS
[✓ JKS]
PKCS #12
BKS-V1
BKS
UBER

2. **Click OK**.

3. Click **Tools > Generate Key Pair**, and select **RSA** for the encryption algorithm.



Algorithm Selection: [✓ RSA] Key Size: 2,048; DSA Key Size: 2,048; EC Set: ANSI X9.62; Named Curve: c2tnb191v1

KeyStore Type: JKS, Size: no entries , Path: 'Untitled-1'

4. Click **OK**.

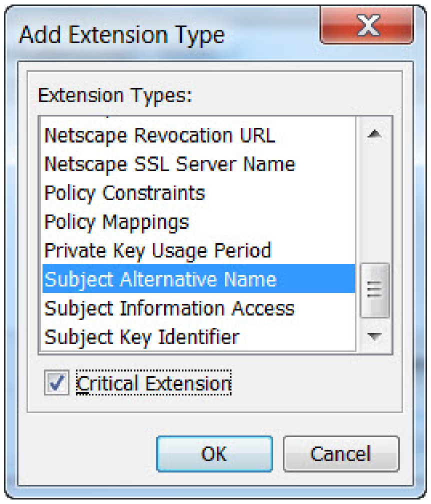
5. In the **Generate Key Pair Certificate** dialog box, select the required **Validity Period**.

6. Click **Add Extensions**.

7. Click ➕ to display the **Add Extension Type** dialog box.

   a. Select **Subject Alternative Name**.

   b. Select the **Critical Extension** check box.



8. **Click OK**.

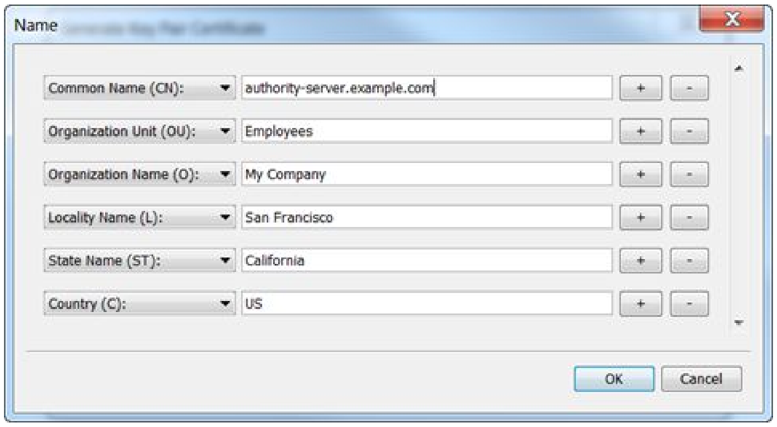
9. In the **Subject Alternative Name Extension** dialog box, click and do the following:

   a. For **General Name Type**, select **DNS Name**.

   b. In the **General Name Value** field, enter the fully-qualified domain name of the server that you want to deploy.

   c. Click **OK** until the **Generate Key Pair Certificate** dialog box is re-displayed.

10. Click the **Edit Name** button and complete the **Name** dialog box with details relevant to your Active Directory Certificate Server.



Common Name (CN):	authority-server.example.com
Organization Unit (OU):	Employees
Organization Name (O):	My Company
Locality Name (L):	San Francisco
State Name (ST):	California
Country (C):	US

11. Click **OK** until you are prompted for an alias name. You can leave the alias name unchanged and click **OK**.

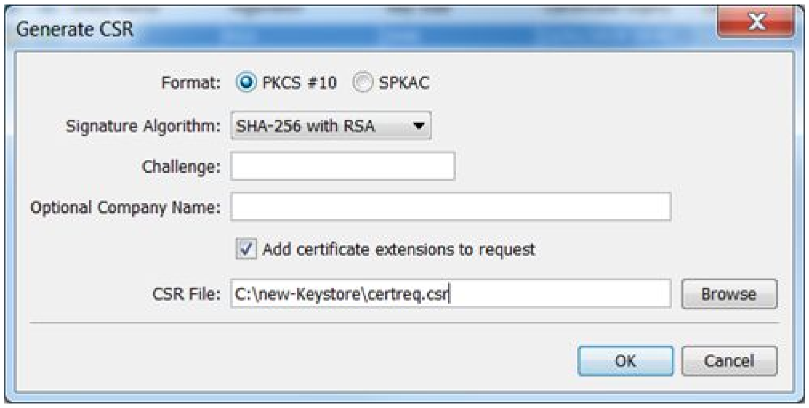
12. In the **New Key Pair Entry Password** dialog box, type a password.

You now have a functioning keystore for your required host. Next, you need to generate a Certificate Signing Request (CSR) that you can submit to your Certificate Authority (CA) for signing.

4.1.3 Generate a Certificate Signing Request (CSR)

To generate a Certificate Signing Request (CSR)

1. Right-click on the generated key pair, and select **Generate CSR** to create a certificate signing request file.
2. In the **Generate CSR** dialog box, select the **Add certificate extensions to request** check box.



Generate CSR

Format: [● PKCS #10] ○ SPKAC

Signature Algorithm: SHA-256 with RSA

Challenge:

Optional Company Name:

[✓] Add certificate extensions to request

CSR File: C:\new-Keystore\certreq.csr

3. Click **Browse** and choose a name and location for the CSR file and click **Save**.
4. In the **Generate CSR** dialog box, click **OK**.

The next step is to sign the CSR that you have created, and this guide describes how to do this using Active Directory Certificate Services. This process can be used if you are building an internal system and do not have access to an external CA, such as VeriSign. If you submit the CSR to an external CA, you should receive both the signed certificate for your server and also the root CA certificate from the Certificate Authority that was used to sign the new certificate. If the CA uses intermediate chain certificates, you will also need these.

Caution

It is crucial that you have the full chain of certificates used to sign your new certificate. Without the complete chain, SSL communication will fail.

4.1.4 Sign the CSR File using Active Directory Certificate Services

Note: These steps offer a worked example using Active Directory Certificate Services. Your organization may use a third-party Certificate Authority to return a signed certificate file.

To sign the CSR file using ADCS:

1. In a text editor, open the `certreq.csr` file that you created in the previous section.
2. On your ADCS server, open a web browser and navigate to `https://`*`<domain_name>`*`/certsrv/certrqxt.asp`. The **Submit a Certificate Request or Renewal Request** page is displayed.
3. Copy and paste the contents of the CSR file from your text editor into the **Saved Request** box for **Base-64–encoded certificate request (CMC or PKCS #10 file or PKCS #7)**.
4. Click **Submit**.
5. In the **Certificate Issued** page, click **Download certificate chain** and save the file to your hard disk.
6. Return to KeyStore Explorer, right-click on the generated key pair and select **Import CA Reply > From File**.
7. Browse to the downloaded certificate.
8. Save the keystore as a JKS file.

4.1.5 Add the Root CA Certificate to the AppWorks Gateway Server

In this step, you update the Java certificate store for the Apache Tomcat instance that is hosting the AppWorks Gateway.

To add the root CA certificate to the Java certificate store:

1. On the ADCS server, open a web browser and go to `https://`*`<domain_name>`*`/certsrv/certrqxt.asp`. The **Download a CA Certificate, Certificate Chain, or CRL** page is displayed.
2. Click the **Download CA certificate chain** link.
3. On the AppWorks Gateway server, navigate to the `cacerts` file in the *`<Java_Home>`*`\lib\security` folder.
4. Open the `cacerts` file in KeyStore Explorer. The default password for the `cacerts` file is “changeit”.

5. Select **Tools > Import Trusted Certificate**.

6. Navigate to your downloaded CA certificate chain.

7. Save the `cacerts` file.

8. Copy the `cacerts` file back into the `<Java_Home>\lib\security` folder on the AppWorks Gateway server.

4.1.6 Add the Keystore to the AppWorks Gateway

The Apache Tomcat instance that will host the AppWorks Gateway needs to be configured to use the chain of certificates from the new keystore so that the runtimes can communicate using SSL to the AppWorks Gateway.

By following these steps, your Apache Tomcat server can still run in normal mode at the same time on port 8080 with HTTP.

To add the new keystore to AppWorks Gateway:

1. Stop the Apache Tomcat server.

2. Place your generated JKS file in the `<Tomcat_Home>\conf` folder on the AppWorks Gateway server.

3. In the `<Tomcat_Home>\conf` folder, open the `server.xml` file in a text editor.

4. Locate the <*Connector*> element with `port=”<8443>”` as seen below.

```
<!-- Define a SSL/TLS HTTP/1.1 Connector on port 8443
This connector uses the NIO implementation that requires the JSSE style
configuration. When using the APR/native implementation, the OpenSSL style
configuration is required as described in the APR/native documentation -->
<!--
<Connector port=”<8443>” protocol="org.apache.coyote.http11.Http11NioProtocol"
maxThreads=”150” SSLEnabled="true" scheme=”https” secure=”true”
clientAuth=”false” sslProtocol=”TLS” />
-->
```

5. Uncomment the property and add the name of the JKS keystore file, and the password you provided when you created it. The following is the new section with the additional line highlighted:

```
<!-- Define a SSL/TLS HTTP/1.1 Connector on port 8443
This connector uses the NIO implementation that requires the JSSE style
configuration. When using the APR/native implementation, the OpenSSL style
configuration is required as described in the APR/native documentation -->

<Connector port=”<8443>” protocol="org.apache.coyote.http11.Http11NioProtocol"
maxThreads=”150” SSLEnabled="true" scheme=”https” secure=”true”
keystoreFile="/conf/keystore.jks" keystorePass="opentext123!"
clientAuth=”false” sslProtocol=”TLS” />
```

6. Save and close the file, and then restart the Apache Tomcat server.

7. To check the setting, in a browser, type `https://<Tomcat_Host>:<8443>`.

Chapter 5

Getting Started with the AppWorks Gateway

The AppWorks Gateway is a deployment and management tool for applications, components, and their associated services and connectors.

5.1 Installing Apps, Services, Components and Connectors

The AppWorks Gateway offers flexibility over the apps and components that your users can access. Apps are separate, independent features in the app runtime client. Because apps are independent from one another, you can allow access to some apps and disable the others. For example, with OpenText ECM Everywhere you can allow access to the Personal Workspace, Enterprise Workspace, and Favorites apps, and hide the Feeds app from your users.

Apps may rely on an associated service. Therefore, you need to make sure all of the services are installed and enabled in the AppWorks Gateway . If you want to prevent your users from accessing a particular app, you must disable the app in the AppWorks Gateway. If you disable a service and not the associated app, users will still be able to see the app on their mobile device or desktop computer. However, users will not be able to use the app.

Components offer additional functionality for apps. For example, with ECM Everywhere, the Object Details Viewer component allows users the ability see the properties, categories, and audit information for objects in the Enterprise Workspace app.

Each app, component, and service in the AppWorks Gateway shows summary, settings, statistics, and history details. Altering the settings might negatively impact performance. Only experienced administrators should modify these settings.

To install an app, service, component or connector:

1. Download the app, service, component or product bundle. A product bundle is a ZIP file of several apps, services, components, and/or connectors. Product bundles typically have a file extension of `.otagbundle`. For example, `tempo_16.2.otagbundle`.
2. Sign in to the AppWorks Gateway as an admin user.
3. In the **Admin Menu**, click **Add** + to install an app.
4. Click **Browse** and navigate to the app, component, service, or product bundle.

   Tip: You can also drag and drop the file onto the indicated region.

5. Click **Install Package** and follow the on-screen instructions. By default, all apps, services, and components are disabled. To give your users access to an app, you need to enable the app as well as the associated service.

5.2 Enabling Apps, Services, Components and Connectors

Once you have installed an app or service and their associated components and connectors, you enable it, to make it available to the clients.

To enable an app, service, component or connector:

- In the AppWorks Gateway, select an option from the **Show** list:
    - Select **Apps**, and then click **Enable** for the apps you want your users to have access to. When an app is disabled, users will not see the app in the runtime client on their mobile devices.
    - Select **Desktop Apps**, and then in click **Enable** for the apps you want your users to have access to. When an app is disabled, users will not see the app in the runtime client on their desktop computers.
    - Select **Components**, and then click **Enable** for all of the components you installed.
    - Select **Services**, and then click **Enable** for all of the services that you installed. When a service is disabled, your users will not be able to access their content. If the associated app is still enabled, users will still see the app in the runtime client on their mobile devices. However, it will be unusable.
    - Select **Connectors**, and then click **Enable** for all of the connectors that you installed.

To enable all installed apps, services, components and connectors:

- To enable all the apps, services, components and connectors that you have installed, select the **Select all** check box and then click **Enable All**.

5.3 Viewing Details and Downloading Apps, Services, and Components

To view details and download apps, services, and components:

1. In the **Admin Menu**, select an option from the **Show** list, for example, **Apps** or **Components**.
2. Click the **Show settings for this app** icon ⚙ for the item.
3. Click the **Overview**, **Settings** or **Audience** tabs to display relevant information, as follows:
   - **Overview** – View the version number, installed date, and the date the feature was last updated.

     You can also download the feature by clicking the **Download** link. Some apps have associated components and services that you also need to download for the app to work in another AppWorks Gateway. Click **Components** or **Services** in the **Admin Menu** and verify if a service or component is present. Based on the name and description of the component or service, it may not be obvious if they affect the downloaded app.
   - **Settings** – These are settings that are specific to the app or service. Use caution when changing and always check the documentation for the app or service.
   - **Audience** – Filter the app based on runtime, user, or group. For details, see "Filtering the Runtime and User Audience" on page 31.

5.4 Filtering the Runtime and User Audience

By default, all enabled apps will appear in every runtime that is connected to the AppWorks Gateway. One example of a runtime is the OpenText AppWorks app that you can download from the Apple App Store (http://www.apple.com/ca) or Google Play (https://play.google.com). In addition to this, all enabled apps will be available to every user or group that is present in your database. To prevent this, you can filter each app to only appear in the runtime that you specify. You can also specify that an app is only available to select users or groups.

To filter the runtime audience:

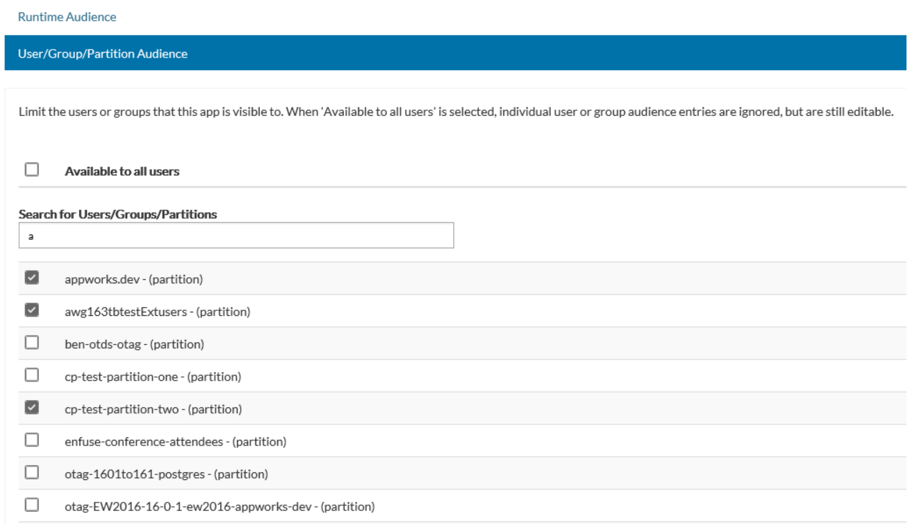
1. In the **Admin Menu**, select **Apps** from the **Show** list.
2. Click the **Settings** icon ⚙ for the app that you want to filter.
3. Click the **Audience** tab and do one of the following:
   - To display the app in all runtimes installed on the user's client, select the **Select all runtimes** check box.

- To display the app in a certain runtime client only, clear the **Select all runtimes** check box and select the check box next to the appropriate runtime.
- To display the app in a runtime that is not displayed, click **Runtimes** from the **Client Management** section of the **Admin Menu**. Click **Add runtime** + to add a new runtime and complete the displayed fields, then click **Save**. Repeat steps 2 – 3, only this time select the check box next to the new runtime.

4. Click **Save**.

To filter the users and groups who can see an app:

1. Click **Installed Applications**, then select **Apps** from the **Show** list.
2. Click the **Settings** icon ⚙ for the app that you want to filter.
3. Click the **Audience** tab.
4. Click the **User**, **Group** or **Partition Audience** link and do one of the following:
   - To display the app in the runtimes of all users, select the **Available to all users** check box.
   - To display the app in the runtimes of a specific user, group or partition, clear the **Available to all users** check box. In the **Search for Users/Groups/Partitions** box, type the first few letters of the name of the user, group or partition. Select one or more check boxes for the users, groups or partitions that you want to be able to access the app.



Limit the users or groups that this app is visible to. When 'Available to all users' is selected, individual user or group audience entries are ignored, but are still editable.

☐ Available to all users

Search for Users/Groups/Partitions

a

- ☑ appworks.dev - (partition)
- ☑ awg163tbtestExtusers - (partition)
- ☐ ben-otds-otag - (partition)
- ☐ cp-test-partition-one - (partition)
- ☑ cp-test-partition-two - (partition)
- ☐ enfuse-conference-attendees - (partition)
- ☐ otag-1601to161-postgres - (partition)
- ☐ otag-EW2016-16-0-1-ew2016-appworks-dev - (partition)

Figure 5-1: Selecting from the results of a search

5. Click **Save**.

5.5 Distributing the Mobile Clients

You can inform your users of the mobile client they need to install for your product by emailing them a link to the app. Before attempting this procedure, make sure you have an SMTP mail server properly configured in your AppWorks Gateway. For details, see “Mail Settings” on page 47.

Note: Although the public versions of the iOS and Android mobile clients for AppWorks Gateway 25.4 are compatible with earlier AppWorks Gateway 16.x versions, push notifications and remote wipe are only available when using AppWorks Gateway 16.5 or a later release. This is due to the migration from Google Cloud Messaging (GCM) to Firebase Cloud Messaging (FCM), which AppWorks Gateway 16.5 and later exclusively support.

To distribute mobile clients:

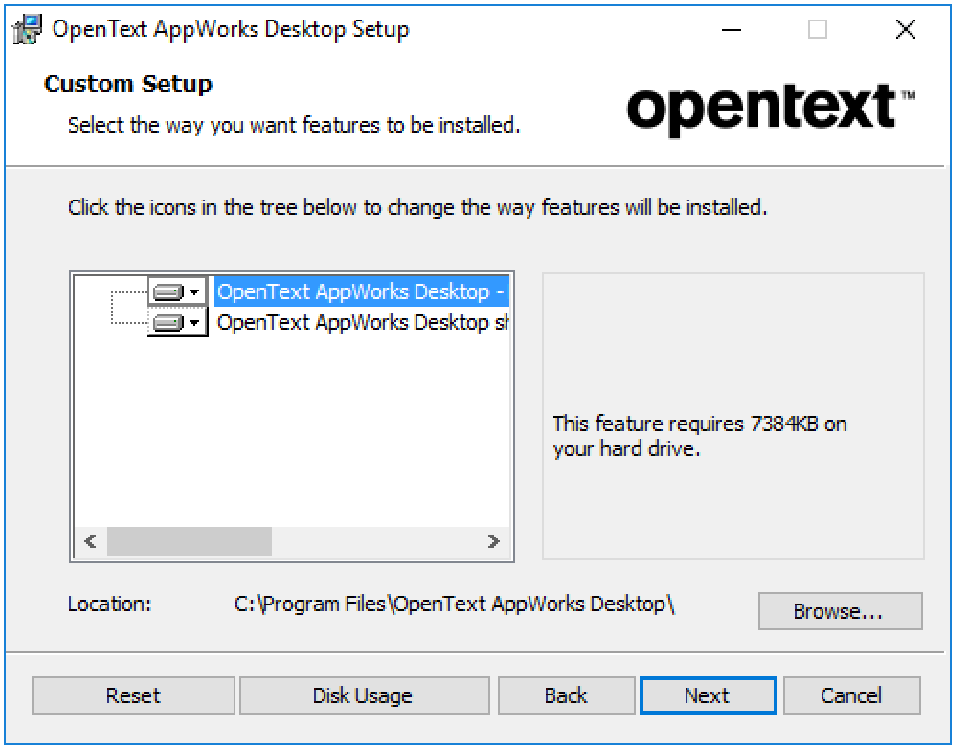
1. Sign in to the AppWorks Gateway as an admin user.
2. In the **Admin Menu**, click **Mobile Client Distribution** from the **Management** section.
3. Complete the details for the distribution email.
4. Select a runtime to be included in the email.
5. Choose to send an email to one ir more groups in OTDS, or to specific users by entering their email address, or to a list of users that are in a CSV file.
6. Click **Send email**.

5.6 Distributing the Desktop Clients

The AppWorks Gateway desktop clients for Windows and macOS® are available from the OpenText My Support.

To install the Desktop Client using the MSI Windows installer

1. Download and run the Windows MSI installation file.
2. Confirm that you have read the OpenText End User License Agreement, and click **Next** to display the **Custom Setup** dialog.


OpenText AppWorks Desktop Setup

Custom Setup

Select the way you want features to be installed.

Click the icons in the tree below to change the way features will be installed.

OpenText AppWorks Desktop -
OpenText AppWorks Desktop sh

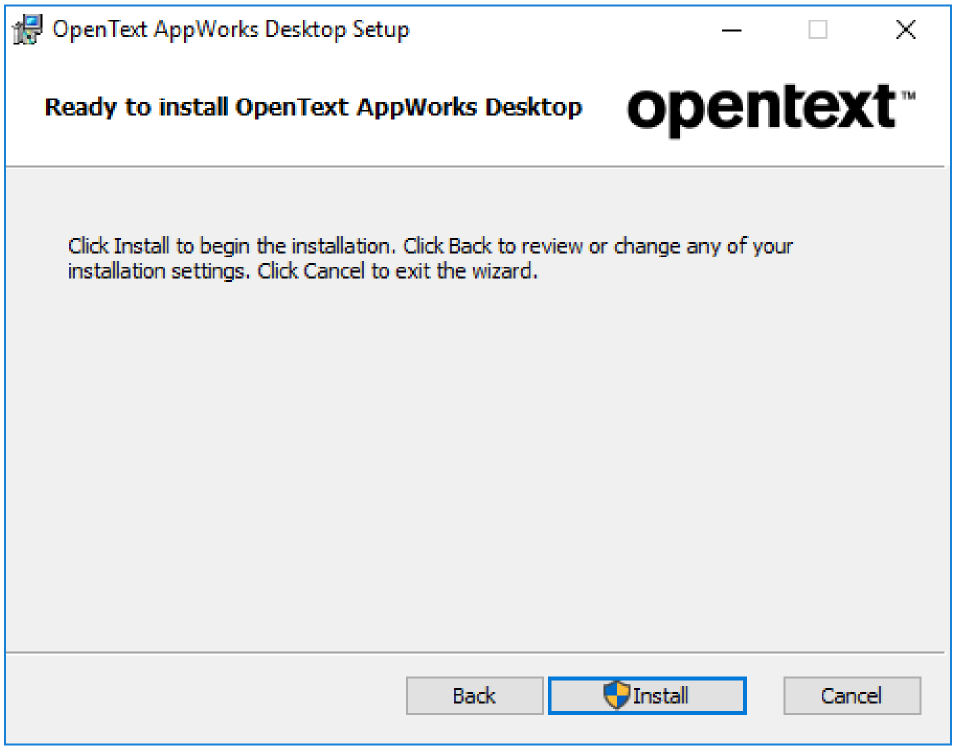
This feature requires 7384KB on your hard drive.

Location: C:\Program Files\OpenText AppWorks Desktop\

Figure 5-2: Installing the Desktop Client – Custom Setup

3. Click **Next** to confirm that you want to install the Desktop Client and the Desktop Client shortcuts in the default location, or click **Browse** to choose an alternative location.

4. Click **Install** to complete the installation.


OpenText AppWorks Desktop Setup
Ready to install OpenText AppWorks Desktop
opentext™

Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.

Figure 5-3: Installing the Desktop Client – Ready to Install

5. Click the OpenText AppWorks Gateway shortcut from the Windows Start menu to display the **Setup** page, and define the following:

   - **Server Address** – The address of the **AppWorks Gateway Server URL**. When you installed the AppWorks Gateway, you entered this URL in the **General Settings** page.
   - **Enable SSO** – Select the check box if you are using SSO.

6. Click **Save**.

To perform an unattended install of the Windows Desktop Client:

You can run the Desktop Client Windows installer silently from a Command Prompt program, passing the filename in a Command Prompt line option. The Command Prompt must be run with administrator level privileges, also known as an *elevated* prompt.

1. From the Command Prompt, navigate to the location of the MSI installer that you have downloaded.
2. The following command will silently install the AppWorks Gateway Desktop Client for Windows, and set the Server Address to `http://localhost:8080`.

```
MSIEXEC /i "OpenText AppWorks Desktop-16.2.0.71.msi" /qn /log log.txt
GATEWAY_URL="http://localhost:8080"
```

Note: Ensure that the above command is on one line.

To install the Desktop Client on macOS®:

1. Mount the macOS® Disk Archive.
2. Read and accept the OpenText End User License Agreement.
3. Drag the AppWorks Gateway Desktop App file to your Applications folder.
4. Launch the OpenText AppWorks Gateway Desktop Client to display the **Setup** page. Define the **Server Address**, **Enable SSO** and **Debug Mode** fields, as described above.
5. Click **Save**.

5.7 Connecting to the AppWorks Gateway Server URL for Mobile and Desktop

When using the OAuth 2.0 or OTDS non OAuth 2 authentication flow in the AppWorks Gateway, ensure that the AppWorks Gateway Server URL is resolvable from clients running the mobile or desktop clients. The AppWorks Gateway Server URL is specified in **Settings** > **General Settings**.

If you receive an Unable to connect error message, when the desktop connects to the AppWorks Gateway Server URL, you can troubleshoot the issue by taking the following steps:

1. On the machine where the desktop client is installed, or on a mobile device, open a browser and navigate to `http://`*`<appworks-server-url:port>`*`/v3/admin/auth/loginurl`.
2. The response displays a value for loginUrl. This is the URL that the desktop client must be able to resolve to, for the InfoCenter Desktop client to connect to the AppWorks Gateway.
3. Enter the loginUrl value in a browser on the desktop or mobile device and check the response. You should get the OpenText login page proxied via the AppWorks Gateway. If the OpenText login page is not displayed, check the format of the AppWorks Gateway Server URL, and change to be a fully qualified domain name.

5.8 Configuring a Desktop Client for HTTPS

To configure a desktop client for HTTPS, you add a PEM file to the machine on which the desktop client is installed. You can then change the server address in the client to HTTPS and connect to an AppWorks Gateway server that is enabled for HTTPS. See "Securing the AppWorks Gateway" on page 21.

The steps to enable the desktop client for HTTPS are as follows:

1. Export the root CA certificate of the AppWorks Gateway server.
2. Convert the CRT file to a PEM File.
3. Add the PEM file to the Application Data folder.

Note: The process for installing the desktop client for specific OpenText products may differ from the steps shown here. Refer to the installation guide for each product to check for additional or modified steps.

5.8.1 Export the Root CA Certificate of the AppWorks Gateway Server

In this step, you export the root CA certificate from the AppWorks Gateway server as a CRT file, using a browser. In this description, we are suing Mozilla Firefox.

To export the root CA certificate on the AppWorks Gateway server:

1. Open Firefox and navigate to the HTTPS address of the AppWorks Gateway login page at `https://`*`<Tomcat_Host>`*`:`*`<port>`*`/gateway`. In "Securing the AppWorks Gateway" on page 21 you enabled the AppWorks Gateway server for HTTPS.
2. Click the padlock icon next to the address bar.
3. Click the chevron icon to the right of the web site address.
4. Click **More Information** and in the **Security** tab, click **View Certificates**.
5. In the **Details** tab, examine the certificate chain in the **Certificate Hierarchy** box.
6. Select the root CA certificate and click **Export**.
7. Save the root CA certificate as a CRT file to your local drive.

5.8.2 Convert the CRT file to a PEM File

In this step, you convert the CRT file to a PEM file using OpenSSL commands.

To convert the CRT file to a PEM file:

1. Open a command prompt window and navigate to the location of the root CA CRT file that you exported in "Export the Root CA Certificate of the AppWorks Gateway Server" on page 37.

2. Type the following OpenSSL command to convert the CRT file to a DER file. Replace the CRT input and DER output values in the example below with your file names.

```
openssl x509 -in appworks-AW-EXAMPLE-CA-2.crt -out appworks-AW-EXAMPLE-CA-2.der -
outform DER
```

3. Type the following OpenSSL command to convert the DER input file to a PEM output file. Replace the DER input value with your file name.

```
openssl x509 -in appworks-AW-DEV-OTDS2-CA-2.der -inform DER -out ca.cert.pem -
outform pem
```

Important
The output file must be called `ca.cert.pem`.

5.8.3 Add the PEM file to the Application Data folder

In this step, you add the PEM file that you created in "Convert the CRT file to a PEM File" on page 38 to the Application Data folder on the machine where you installed the desktop client.

To add the PEM file to the Application Data folder:

1. Navigate to the `\AppData\Roaming\appworks-desktop\`*`<app-type>`*`\` folder.

2. Create a folder called `certs`.

3. Add the `ca.cert.pem` file to the `certs` folder.

4. Open the desktop client and ensure that the server address in the **Settings** page is the HTTPS address of your AppWorks Gateway server.

5. Restart the client. The OpenText login page is now displayed over HTTPS.

Notes

- On a Mac, create the `certs` folder in `~/Library/Application/Support/appworks-desktop/`*`<app-type>`*`/`.

- These steps assume that there are no intermediate certificates in the certificate chain. If there are intermediate certificates, then you need to combine the PEM files for the root CA certificate and the intermediate certificates into a single file, called `ca.cert.pem`. For example, if you have a root CA certificate and two intermediate certificates, assemble the `ca.cert.pem` file as follows:

```
-----BEGIN CERTIFICATE-----
<Certificate for INTERMEDIATE CERTIFICATE AUTHORITY 2>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate for INTERMEDIATE CERTIFICATE AUTHORITY 1>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate for ROOT CERTIFICATE AUTHORITY>
-----END CERTIFICATE-----
```

5.9 Uninstalling the Desktop Clients

To uninstall the Windows Desktop Client:

1. Using the Windows uninstall functionality, remove the Desktop Client, as follows:

   - For Windows 8.1, navigate to **Programs and Features**, select the OpenText AppWorks Desktop application and click **Uninstall**.
   - For Windows 10, navigate to **Apps & features**, select the OpenText AppWorks Desktop application and click **Uninstall**.

2. Open Windows Explorer, navigate to the following folder and delete the contents:

```
C:\Users\<your user name>\AppData\Roaming\appworks-desktop\standard
```

Note: For the eDOCS InfoCenter desktop client, navigate to the following folder and delete the contents:

```
C:\Users\<your user name>\AppData\Roaming\appworks-desktop\edocs-infocenter
```

To uninstall the macOS Desktop Client

1. Drag the Application from the Applications folder to the Trash folder.

2. Use the Finder or Terminal to navigate to the following folder and delete the contents:

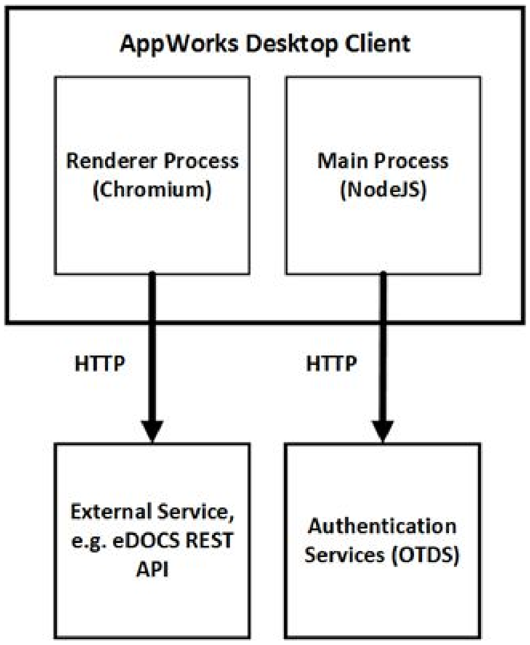
```
~/Library/Application\ Support/appworks-desktop/standard
```

Note: For the eDOCS InfoCenter desktop client, navigate to the following folder and delete the contents:

```
~/Library/Application\ Support/appworks-desktop/edocs-infocenter
```

5.10 Configuring the AppWorks Gateway Client to use an HTTP Proxy

To enable the AppWorks Gateway Client to use an HTTP proxy, you must be aware of the two ways in which the AppWorks Desktop Client picks up HTTP forward proxy configurations. In the diagram below, the Renderer process is a repackaged Chromium browser, while the Main process is repackaged NodeJS runtime.


AppWorks Desktop Client
Renderer Process (Chromium)
Main Process (NodeJS)
HTTP
HTTP
External Service, e.g. eDOCS REST API
Authentication Services (OTDS)

Both the Renderer process and the Main process must make outgoing HTTP connections. The Renderer process uses HTTP connections provided by the Chromium framework, while the Main process uses HTTP connections provided under the NodeJS framework.

To pick up HTTP forward proxy configuration, the Renderer process uses the built-in proxy configuration of the host operating system. The Main process picks up its HTTP configuration from HTTP_PROXY and HTTPS_PROXY environment variables. You must ensure that both configuration methods are employed at the same time to configure the AppWorks Desktop Client to use an HTTP proxy.

5.11 Configuring the AppWorks Gateway for Cross-Origin Resource Sharing

To enhance security, you can use Cross-Origin Resource Sharing (CORS) to specify the web origins that can access your server's resources.

To configure the AppWorks Gateway for CORS update the `AWG_CORS_ALLOWED_ORIGINS` environment variable in Apache Tomcat, using the `setenv.sh` file for Linux, or the `setenv.bat` file for Windows. If the files do not exist, you must create them.

Note: The CORS configuration setting for the container is `allowedOrigins`. See "Updating the AppWorks Gateway (awg) Settings" on page 73.

To enable CORS in a Linux environment:

1. Open the `setenv.sh` file in the `TOMCAT_HOME/bin/` directory.
2. Add a line to the `setenv.sh`, specifying the origins to use in a comma-separated list, for example:
   ```
   export AWG_CORS_ALLOWED_ORIGINS=https://example.com,https://app.example.com
   ```
3. Save the `setenv.sh` file.
4. Restart Apache Tomcat.

To enable CORS in a Windows environment:

1. Open the `setenv.bat` file in the `TOMCAT_HOME/bin/` directory.
2. Add a line to the `setenv.bat` file, specifying the origins to use in a comma-separated list, for example:
   ```
   set AWG_CORS_ALLOWED_ORIGINS=https://example.com,https://app.example.com
   ```
3. Save the `setenv.bat` file.
4. Restart Apache Tomcat.

Chapter 6

AppWorks Gateway Settings

This chapter describes the settings that are available for the following categories:

- **General**: Specify the URL of the AppWorks Gateway server and the OTDS server, and choose from various authentication options.
- **MDM**: Specify the settings for mobile device management, for example, whether to enable or disable client tracking.
- **Mail** : Specify the SMTP settings to use when sending emails from the AppWorks Gateway.
- **Notifications**: Determine the settings to use when sending messages to client applications.
- **Audit**: Determine the audit information that will be captured for updates and changes made to the system, as well as sign in and sign out activities.

6.1 General Settings

To define the general settings:

1. Sign in to the AppWorks Gateway as an admin user.
2. In the **Admin Menu**, click **Settings > General Settings**.
3. Define the following:
   - **OpenText AppWorks Gateway Server URL** – The URL that clients and providers use to reach this server. Use the fully qualified host name of the server. Do not use `http://localhost:8080`. This URL is also needed for links in external invitations and for auto-configuration of the OTSync Connector. The OTSync Connector provides a connection, authentication handling, proxy mappings and trusted provider key registration for OpenText Tempo Box.

     Note: If you change the **OpenText AppWorks Gateway Server URL**, stop and restart the Apache Tomcat service.
   - **Want Secure Cookies** – When enabled, cookies issued by the AppWorks Gateway will have their “secure” flag set. This setting has no effect for responses on a non-secure transport, for example HTTP. The **Want Secure Cookies** setting is selected by default.
   - **Session Token Cleanup Interval (s)** – Timed outAppWorks Gateway session tokens will be cleared from memory periodically given this duration.
   - **Session Token Timeout (s)** – Session tokens generated by the AppWorks Gateway layer will become invalid after this period of inactivity.

- **Enable automatic sign-out** – Select the check box, then enter a period of inactivity in the **Auto Sign-Out Inactivity Period (s)** field. A **Session timeout** dialog box appears to inform the user that they will be logged out after the period specified in **Auto Sign-Out Inactivity Period (s)**. OpenText recommends a timeout of 15 minutes.
- **Auto Sign-Out Inactivity Period (s)** – Users will be automatically signed out after this period of inactivity. Enter a value in seconds of at least 120. The value must not have a decimal point, for example 900.5. The value must be less than the period entered in the **Session Token Timeout (s)** field. A period of inactivity is where the user makes no input into the Administration User Interface by keyboard or mouse. One minute before the user's session is due to end, a dialog box is displayed enabling the user to extend the session.
- **OpenText Directory Services (OTDS) URL** – The URL of the OTDS service that is used by AppWorks Gateway for authentication and identity information. When you installed AppWorks Gateway, you connected to an OTDS service. You cannot change to a different OTDS service after installation, and this field is therefore read-only.
- **Do Not Proxy OTDS** – The proxy server will not be used by the client to obtain an OTDS login URL. Instead, the client will use the URL specified in the OpenText AppWorks Gateway.
- **Cookie Domain for Generated OTDS Cookies** – Enter a cookie domain for generated OTDS cookies generated by OpenText AppWorks Gateway to be used across subdomains. For example, enter ".example.com" to allow subdomain `app2.example.com` to use cookies generated for subdomain `app1.example.com`.
- **Anonymous Access Enabled** – Enable or disable anonymous (unauthenticated) access for mobile clients.

  Note: Contact your OpenText Support representative before enabling anonymous access. There are a number of steps to take to implement this functionality and your representative will discuss the process with you.
- **Force password based (non-OAuth) authentication** – Asks the mobile client to use form-based authentication. There will be no OTDS single sign on regardless of whether OTDS supports OAuth. Ensure that this check box is selected when you install the AppWorks Gateway with the following OpenText products:
  - AppWorks Suite
  - Tempo Box
  - Content Server Mobile
- **Disable OTDS OAuth** – Disables the use of OAuth within OpenText AppWorks Gateway. By default, the mobile client shows the OTDS login page and uses OAuth for login. If you choose to select **Disable OTDS**

OAuth, the mobile client will show the AppWorks Gateway login page but will not use OAuth for login.

- **OTDS SSO Disable by IP (Comma Separated List, CIDR Notation) –** Allows the administrator to disable the single sign on dialog for users outside the network. For example, with Tempo Box external users.
- **List of Tenants, Comma Separated, or * to Permit All –** Type the names of one or more tenants. Tenant names may be comprised of the characters 'a' to 'z', '0' to '9'. Invalid entries will be ignored. When you click **Update Settings**, you can install each tenant by entering the details for a database and an OTDS instance. See “Deploying the AppWorks Gateway for Multi-Tenancy“ on page 61.
- **OpenText AppWorks Gateway Console Developer Mode –** When selected, this adds **User Management** to the **Management** section of the **Admin Menu**. It also increases the level of detail in the `gateway.log` error responses.
- **Limit Concurrent Sessions –** Enables you to apply a limit to the number of concurrent sessions for administrator users and regular users. Enter the required limits in the **Concurrent Session limit for administrators** and **Concurrent Session limit for normal users** fields. When the maximum number of sessions is reached, the user can choose to end an existing session to enable them to start a session on a new device, for example.
- **Concurrent Session limit for administrators –** When **Limit Concurrent Sessions** is enabled, enter the maximum number of concurrent sessions for an administrator user.
- **Concurrent Session limit for normal users –** When **Limit Concurrent Sessions** is enabled, enter the maximum number of concurrent sessions for a regular user.

  Note: OpenText recommends the following concurrent sessions:

  – Three administrator sessions

  – Two regular user sessions.

4. Click **Update Settings**.

6.2 MDM Settings

To define the mobile device management settings:

1. In the **Admin Menu**, click **Settings > MDM Settings**.
2. Define the following:

   - **Client Tracking Enabled –** If enabled, you can track the clients that a user has used and perform a remote wipe of any applications on the mobile client.

- **Remote Wipe Timeout (s)** – An email is sent to the recipients in the **Remote Wipe Failure Email Recipients** field if a remote wipe was not successful before the timeout period expires.
- **Remote Wipe Failure Email Recipients** – The email addresses listed in this field will be emailed if a remote wipe action did not occur before the **Remote Wipe Timeout (s)** period expires. If you receive a remote wipe email failure notification, the wipe did not take place and you must reset the user password to prevent the user from signing into their account.
- **Permit Storage of Password (On Device)** – After a user signs in for the first time, this setting allows your users to close and reopen the app without typing their password. This functionality is only available when the mobile clients are using the default oAuth 2.0 flow.

  Notes

  – If the user signs out of the app, they must re-enter their password to sign back in.

  – If you set a value in seconds in he **Password on Device Expiry (s)** field, the user must sign back in to the app if the duration is exceeded.
- **Password on Device Expiry (s)** – The duration in which a password is stored on a device.
- **Permit Offline Access on Device** – When selected, users can access information on a device when it is offline.
- **Notify Devices of Offline Policy Change** – When selected, users will be notified if the value of the **Permit Offline Access on Device** field changes.
- **Disable Sharing and Downloading Files** – When selected, you can restrict the sharing and downloading of files in mobile clients.
- **Override Consumer Provided Config** – This setting enables the mobile app to override the AppWorks Gateway session timeout settings and adhere to the session timeout configuration of the target system to which the micro mobile app is connected.
- **Enable Biometric Authentication** – The user can enable the mandatory biometric authentication in mobile clients.
- **Enable Offline Pin Retry** – The user can set the offline pin retry in mobile clients.
- **Offline Pin Retry Attempts** – Determines the number of attempts for the offline pin retry in mobile clients. The default value is 5.

3. Click **Update Settings**.

6.3 Mail Settings

In the **Mail** tab, you choose the type of authentication type for email: **Password** or **OAuth**. Depending on your selection, you then provide the details of the SMTP mail server or the OAuth mail server.

To define the mail settings for Password or OAuth authentication:

1. In the **Admin Menu**, click **Settings > Mail**.
2. From the **Authorization type** list, select **Password** or **OAuth**.
   - For the **Password** authorization type, provide the following details:
     - **SMTP Return Address –** The email that is generated by the OpenText AppWorks Gateway will be sent from this email address.
     - **SMTP Host –** The host name or IP for the SMTP mail server.
     - **SMTP Port –** The port for connecting to the SMTP mail server. The default port is 25.
     - **SMTP Username –** The username for connecting to the SMTP mail server. This setting is optional.
     - **SMTP User Password –** The password for connecting to the SMTP mail server. This setting is optional.
     - **SMTP Use SSL –** If enabled, communication with the SMTP mail server will be secured using SSL.
   - For the **OAuth** authorization type, do the following:
     1. Register an application in Azure Active Directory for Microsoft Outlook, or in Google Cloud for Gmail,
     2. Provide the AppWorks Gateway redirect-uri in the **Redirect URI** section of the Azure or Google portal, in the form: `<appworksGatewayUrl>/v3/admin/oauth2/authorize`.

        Note: For the Azure portal, you must also add the OpenID API permissions from the delegated permissions in the Azure Portal.
     3. Complete the following settings:
        - **OAuth mail server –** Select **Google** or **Microsoft**.
        - **Client ID –** Enter the ID of the application that you created.
        - **Client secret –** Enter the client secret that you registered for the application.
        - **(Optional))Refresh token expiry time in days –** Refresh tokens are used to get a new access token when your current access token expires. Enter the lifetime of the refresh token in days. You will be prompted to reauthorize near to the expiry date.

3. Click **Save**.
4. Click **Authorize** to provide consent for the AppWorks Gateway to use this Mail configuration.
5. In the sign-in accounts dialog box for Google or Microsoft, choose an account related to the above application to authorize.
6. Enter the credentials for the account that you are authorizing and click **Next**. A message is displayed to confirm that OAuth authorization has been successful.

   Note: You can update the value in the **(Optional) Refresh token expiry time in days** at any time, and click **Reauthorize**.

7. To test the settings, do the following:

   a. In the **Send Test Email** section, enter an email address in the **To** field.
   b. Complete the **Subject** field.
   c. Optional Type a message in the **Message** field.
   d. Click **Send**.

6.4 Notification Settings

To define the notification settings:

1. In the **Admin Menu**, click **Settings > Notifications**.
2. Define the following:

   - **Notification Purge Interval (s) –** The interval at which the notifications table will be purged of notification records.
   - **Maximum Notifications Table Size –** The maximum number of notification records retained before a purge is triggered by the table monitor.
   - **Back-channel Stale Connection Cleanup Period (s) –** The interval at which back-channel connections will be checked for freshness and stale connections discarded.
   - **Notification Refresh Interval (s) –** The interval at which fresh notification sequence data is published to clients. This affects how often a client will ask for new notifications based on the last known sequence value.
   - **Back-channel Timeout (ms) –** The time in which a client long poll connection for notifications will remain open.
   - **Max Long-poll Connections per User –** This is used to prevent DoS attacks. When a user connects more than this number of clients simultaneously, the oldest connection will be dropped.
   - **Back-channel Stale Connection Cleanup Age (s) –** Back-channel connections will be cleared from memory when they have been idle for at least this long.

- **Permitted Get Request Interval (s)**— The interval at which notifications clients are permitted to request notifications before being throttled. Client requests will be dropped in favour of a Denial of Service at the Gateway.
- **Activate Push Notification Logging** – Specifies whether push notification activity is logged to `fcm.log`.

  Note: Logging the push notification activity results in a large log file that must be actively managed.

3. Click **Update Settings**.

To run a manual notifications purge:

1. Select the **Use Cutoff Date** check box and enter a date.
2. Click **Send Purge Request** to remove notifications that were created before the **Use Cutoff Date**.

   If you have not entered a cutoff date, the current, completed notifications will be purged. The number of notification is shown next to **Current notifications row count**. This value is updated after the period specified in the **Notification Purge Interval (s)**.

6.5 Audit Settings

You can determine the audit information that will be captured for updates and changes made to the system, as well as sign in and sign out activities. The details are stored in the `otagaudit` table, which you can query with a database administration tool.

To configure the audit information to store

1. In the **Admin Menu**, click **Settings > Audit**.
2. Select the check boxes next to the settings you require. The settings are grouped under a main heading, for example, **Authentication**, **Runtimes** and so on. You can deselect the main heading to choose more specific settings. For example, you might want to identify when a runtime is created or deleted, but not when a runtime is updated, or when the list of runtimes is displayed in the Administration User Interface.
3. Click **Update Settings**. Only the actions that you have specified will be reported in the `otagaudit` table.

Chapter 7

Creating a Proxy Rule

In the Proxy Settings page, complete the details for any proxy rules that you want to create for AppWorks Gateway. OpenText supplies various template proxy rules for use with OpenText products such as eDOCS InfoCenter and Content Server. Creating a proxy rule for your OpenText product is part of the installation process for the product, and further details are available in the specific installation documentation.

Note: The proxy is intended as an API proxy for AppWorks services and apps, and is not to be seen as a general purpose replacement for another proxy service, such as mod_proxy for the Apache HTTP Server.

To define a proxy rule:

1. Sign in to the AppWorks Gateway as an admin user.
2. In the **Admin Menu**, click **Proxy**.
3. Click **Add** + and choose from the following options:
   - **Create new**: Add a name for the rule, and the required settings for the rule.
   - **Create from template**: This option provides some default settings for installing the AppWorks Gateway with other OpenText products. For example, the OpenText eDOCS RestAPI template includes URL mappings and Whitelist details that help you to install OpenText eDOCS with InfoCenter.
   - **Import**: Browse to a file in .JSON format, with the required settings for one or more proxy rules.
4. Click **Create** to add the new proxy rule settings.

Chapter 8

Enabling Push Notifications for Firebase Cloud Messaging

The AppWorks Gateway uses the public Firebase Cloud Message (FCM) service to enable native push notifications. Push notifications are configured for each mobile runtime using the Google developer console. A Firebase Project model is used for working with Firebase projects in the AppWorks Gateway.

To set up push notifications

1. Sign in to the AppWorks Gateway as an admin user.
2. In the **Admin Menu**, click **FCM Configuration**.
3. Click **Add Entry** and complete the following fields:
    - **Project ID** – A globally unique identifier for the project
    - **Project Private Key File** – A service account JSON file also used to call Firebase APIs.
    - **Server Key** – A key to authorize AppWorks Gateways calls against the Firebase APIs
    - **Sender ID** – The ID used by the AppWorks Gateway for itself as a sender of push notifications, push clients ask to be sent notifications from a sender
    - **Web Push Certificate Public Key** – A public key used by, the now supported, in browser push clients.
    - **Web Push SDK Configuration** – A JSON configuration String to allow in-browser web clients to easily make use of the Firebase JavaScript client library, see here (https://github.com/firebase/firebase-js-sdk).

The following section assumes you have a valid Google developer account and can access the console at https://console.firebase.google.com. Select the cog wheel in the left-hand menu at the **Project Overview** level, and select **Project settings** to view the Firebase Project Settings.

This tabbed view provides access to the general project's details and the cloud messaging configuration, which includes any number of apps and their access files (google-services.json, GoogleService-Info.plist), and service accounts.

Project ID
The general project identifier found on the **General** tab under **Settings**.

Server Key
A server key that authorizes our app server, the AppWorks Gateway, to access Google services. It is available on the **Cloud Messaging** under **Settings**.

Project Private Key File

This is a service account file generated for the project. It is used by the AppWorks Gateway when communicating with the Google APIs involved in sending push notifications to mobile and web clients.

In the **Service Account** tab of the **Settings** we can generate a new file using the **Generate new private key**. This file should be uploaded in the Firebase Project creation form in the AppWorks Gateway.

Web Push Notifications

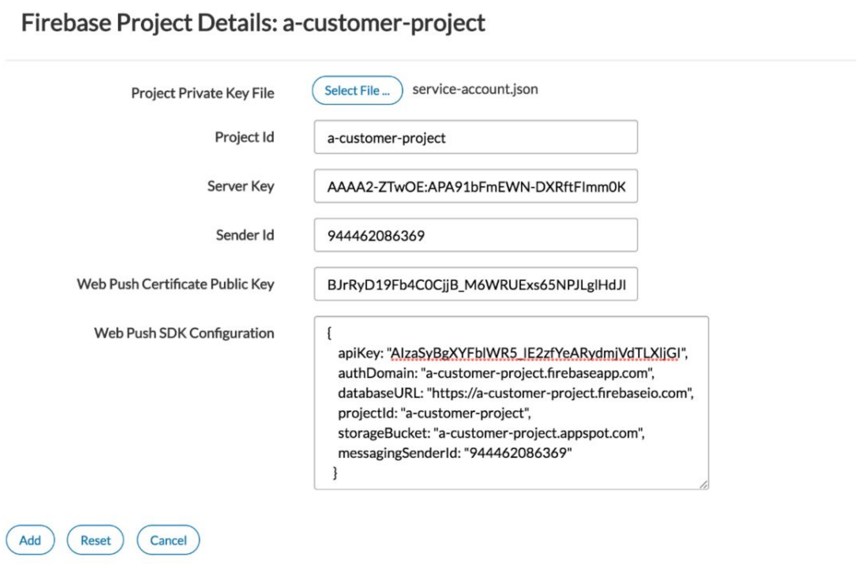
Firebase Cloud Messaging supports web clients as well as Android and iOS clients. The following two fields on the Firebase Project support web-based clients that contact the AppWorks Gateway for such messages.

- **Web Push Certificate Public Key** – The public key of any Web Push certificate generated in the Web Configuration found at the bottom of the **Cloud Messaging** tab under **Settings** for the project.
- **Web Push SDK Configuration** – The public configuration code block for a JavaScript app that will use the FCM client exposed by the AppWorks Gateway. The value can be copied directly from the Firebase console as described below. JavaScript applications may use the Firebase JavaScript SDK to receive push notifications in the browser. From the Firebase projects overview page, do the following:

  1. Click **+ Add App** below the project name.
  2. Select the web application, depicted by angled brackets.
  3. Copy the content of the config variable is JSON block out of the provided <script> tags, for example:

```
{
   apiKey: "AIzaSyBgXYFblWR5_IE2zfYeARydmjVdTLXljGI",
   authDomain: "a-customer-project.firebaseapp.com",
   databaseURL: "https://a-customer-project.firebaseio.com",
   projectId: "a-customer-project",
   storageBucket: "a-customer-project.appspot.com",
   messagingSenderId: "944462086369"
  }
```

  4. Paste the JSON block into the Web Push Notifications field in the AppWorks Gateway:



Firebase Project Details: a-customer-project

Project Private Key File	service-account.json
Project Id	a-customer-project
Server Key	AAAA2-ZTwOE:APA91bFmEWN-DXRftFlmm0K
Sender Id	944462086369
Web Push Certificate Public Key	BJrRyD19Fb4C0CjjB_M6WRUExs65NPJLglHdJI

Web Push SDK Configuration

```
{
  apiKey: "AIzaSyBgXYFblWR5_IE2zfYeARydmjVdTLXljGI",
  authDomain: "a-customer-project.firebaseapp.com",
  databaseURL: "https://a-customer-project.firebaseio.com",
  projectId: "a-customer-project",
  storageBucket: "a-customer-project.appspot.com",
  messagingSenderId: "944462086369"
 }
```

Figure 8-1: Firebase Project Details

8.1 Sending a Test Notification

You can send a test notification to connected mobile devices. A mobile client must have connected to the AppWorks Gateway successfully at least once to receive a notification. When you complete the details of the notification, you specify the target app and the users that you want to notify.

To send a test notification:

1. Sign in to the AppWorks Gateway as an admin user.
2. In the **Admin Menu**, click **Push Notification > Notification Test**.
3. In the **Native Alert** region, enter a **Title** and **Notification Summary**. The data provided here will be shown in the alert seen on the receiving devices operating system, as opposed to within the AppWorks mobile client.

   Depending on the device version and platform this may mean the data will be visible in a notification center, on a banner or in an interactive alert.
4. In the **AppWorks Data** region, enter the **Data Summary** and **Data Payload**. The data provided is used by the AppWorks mobile client, and will feature on the notifications seen therein. Notification data can be passed directly into a target AppWorks app running within the AppWorks client via a JSON payload. This payload is used in conjunction with the AppWorksJS framework provided for mobile developers to create reactive hybrid applications.
5. In the **Target AppWorks App** list, select the app to which you want to send the notification.

6. In the **Username/First Name/Last Name** field, search for the user to whom you want to send this notification.
7. Select the required user from the search results.
8. Click **Send**.

Chapter 9

Monitoring and Managing User’s Access

Using Client Management, you can see and wipe the runtimes from your user’s devices that your users have used to access their account. Wiping deletes all local copies of server data from the runtime and prevents the user from signing in. If the runtime was accidentally wiped, the user can re-enter the AppWorks Gateway URL, sign in and continue to access their account. The user will need to reinstall all of the apps and download their favorite files again. Assuming the wipe was intentional, you should change the user's password or delete their account altogether to prevent the user from signing in again.

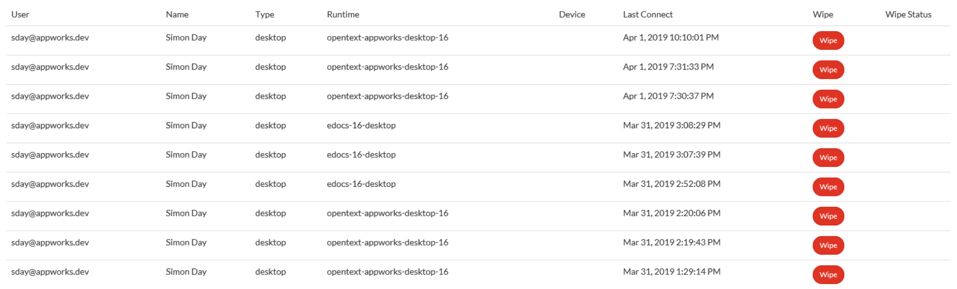
9.1 Monitoring Client Management

To monitor client management:

1. Sign in to the AppWorks Gateway as an admin user.
2. In the **Admin Menu**, click **Client Management** from the **Management** section.
3. Type the username, first name, or last name of a user and then press **ENTER**.

   Tip: If you are unable to search for a user, verify that **Client Tracking Enabled** is selected on the **MDM Settings** page.

   You will see the username, runtime, device information, and connection times for every client the user has used to access their account.



User	Name	Type	Runtime	Device	Last Connect	Wipe	Wipe Status
sday@appworks.dev	Simon Day	desktop	opentext-appworks-desktop-16		Apr 1, 2019 10:10:01 PM	Wipe	
sday@appworks.dev	Simon Day	desktop	opentext-appworks-desktop-16		Apr 1, 2019 7:31:33 PM	Wipe	
sday@appworks.dev	Simon Day	desktop	opentext-appworks-desktop-16		Apr 1, 2019 7:30:37 PM	Wipe	
sday@appworks.dev	Simon Day	desktop	edocs-16-desktop		Mar 31, 2019 3:08:29 PM	Wipe	
sday@appworks.dev	Simon Day	desktop	edocs-16-desktop		Mar 31, 2019 3:07:39 PM	Wipe	
sday@appworks.dev	Simon Day	desktop	edocs-16-desktop		Mar 31, 2019 2:52:08 PM	Wipe	
sday@appworks.dev	Simon Day	desktop	opentext-appworks-desktop-16		Mar 31, 2019 2:20:06 PM	Wipe	
sday@appworks.dev	Simon Day	desktop	opentext-appworks-desktop-16		Mar 31, 2019 2:19:43 PM	Wipe	
sday@appworks.dev	Simon Day	desktop	opentext-appworks-desktop-16		Mar 31, 2019 1:29:14 PM	Wipe	

Figure 9-1: Admin Menu – Client Management

9.2 Performing a remote wipe

To perform a remote wipe of the applications on a client, the **Client Tracking Enabled** setting in the **MDM Settings** page must be selected.

Note: This functionality is only available for the Mobile Client.

To perform a remote wipe of the applications on a client:

1. In the **Admin Menu**, click **Client Management** from the **Management** section.
2. Type the username, first name, or last name of a user and then press **ENTER**. If you are unable to search for a user, verify that **Client Tracking Enabled** is selected on the **MDM Settings** page.
3. Click the **Wipe** link for the client that you want to wipe.
4. Click **Ok** to confirm the action. The next time the user attempts to access their account using the client ID you wiped, the runtime wipe will take place.

Chapter 10

Deploying the AppWorks Gateway in a Cluster

You can deploy and configure additional AppWorks Gateway servers to use an existing AppWorks Gateway database. Any applications, services, and components that are installed on one node in the cluster are automatically installed on all of the other nodes. All settings changes are also propagated to all of the nodes. You do not need to perform any manual steps on any other node or restart the Apache Tomcat service. However, you must restart the Apache Tomcat service if you change any of the following settings in the AppWorks Gateway:

- On the **Notifications** page, changes to the **Back-channel Stale Connection Cleanup Period (s)** only take effect when the current period elapses or when the Apache Tomcat servers are restarted.
- On the **General Settings** page, changes to the **Session Token Cleanup Interval (s)** and the **Session Token Timeout (s)** will only take effect once the current interval elapses or when the Apache Tomcat servers are restarted.

Note: In most cases, you should not change these settings.

Clients that are connected to one node can also switch to another node and continue functioning.

To deploy OpenText AppWorks Gateway on additional nodes:

1. Install Apache Tomcat on the new node. For details, see "Installing the AppWorks Gateway" on page 9.
2. Stop the Apache Tomcat service.
3. Extract the contents of the `otag-gateway–25.4.zip` file to the root of the Apache Tomcat folder, for example, `C:\Program Files\Apache Software Foundation \Tomcat 10.0`.
4. Start the Apache Tomcat service.

   Tip: Navigate to the <*Tomcat_Home*>`\logs` folder and open the `catalina.`<*yyyy-mm-dd*> file. When you see a line containing INFO: Server startup in <*55278*> ms, you can start to configure the AppWorks Gateway.
5. Complete the database configuration steps to connect to the database you are using for all nodes in the cluster.
6. Configure a load-balancer in front of the AppWorks Gateway nodes. When complete, sign in to the AppWorks Gateway, click **Settings**, and edit the **OpenText AppWorks Gateway Server URL** to point to the load-balancer's address.

Note: Repeat this procedure at any time to add nodes to the cluster. They must also be added to the load-balancer.

Chapter 11

Deploying the AppWorks Gateway for Multi-Tenancy

A multi-tenant AppWorks Gateway installation enables you to create multiple AppWorks Gateway instances on the same server, each with their own database and OTDS configuration details. Each tenant installation can have a separate administrator, with the administrator of the primary AppWorks Gateway having overall control.

To create a new tenant for a primary AppWorks Gateway installation:

1. Install AppWorks Gateway on a server, as described in "Installing the AppWorks Gateway" on page 9.
2. In the **Admin Menu**, click **Settings > General Settings**.
3. In the **List of Tenants** field, type one or more tenants. Tenant names can include the characters 'a' to 'z', and '0' to '9'. Upper case characters are not valid.
4. Click **Update Settings**. A URL is created for each tenant name. For example, if you type "tenant001" in the **List of Tenants** field, the URL `http://host:port/awtenant/tenant001/` is created.
5. To complete the installation, enter the URL for the tenant into a browser address bar. A configuration page appears with options to specify a database and an OTDS instance. This is the same page that you use to complete the configuration of any AppWorks Gateway installation.
6. For **Step 1: Database Configuration**, specify a database, and click **Configure Database Connection**. The database for each tenant must be unique. It must not be the same as the database for the primary AppWorks Gateway installation.
7. For **Step 2: OTDS Configuration**, enter the OTDS server details that you used for your primary AppWorks Gateway installation.
8. For each tenant, provide details for the **Gateway Admin Username/Password**, and **OTDS Administrative Account/Username**. You can use the same administrator as your primary installation, or you can create different administrators for each tenant.
9. For each tenant, provide a **Resource Name** and **Gateway User Partition Name**.
10. Click **Configure OTDS**.

Chapter 12

Deploying the AppWorks Gateway as a Container

This chapter contains information about setting up, supporting, installing, configuring, and deploying the AppWorks Gateway on certified cloud environments. Migration to the cloud can convey the following benefits:

- Reduce the high operating costs to develop, manage and maintain on-premises applications
- Avoid end user adoption issues caused by slow performance and lengthy deployment timelines
- Gain access to extensive resources to support enterprise information management applications
- Deploy enterprise information management applications and grow as needed to scale to your business needs

AppWorks Gateway provides Docker images that can be downloaded from OpenText with Docker YAML files and Kubernetes Helm charts and scripts. You can create an init container to add the desktop and mobile apps that you want to deploy with the AppWorks Gateway . If you deploy the product using the default parameters, AppWorks Gateway will use Apache Tomcat as its web server, PostgreSQL as its database, and OpenText Directory Services (OTDS) to manage user and group identity information.

Note: Apache Tomcat and OpenJDK are installed as part of the AppWorks Gateway container deployment. You do not have to install this software separately. The OTDS and database installations must be installed separately.

12.1 Before You Start

To perform the steps outlined in this document, you will need the items listed below. Refer to the *OpenText AppWorks Gateway Release Notes* for information on supported platforms and versions.

A database installation
The AppWorks Gateway container deployment process connects to the database and creates a database instance. The supported databases for a container deployment of AppWorks Gateway are PostgreSql, Oracle and Microsoft SQL Server.

OpenText Directory Services (OTDS)
The AppWorks Gateway container deployment process checks for an existing instance of OTDS.

Helm
Helm is used to deploy the Docker images.

Docker
You tag the Docker image files and push them to your registry.

Kubernetes
You use Kubernetes to run various commands during the deployment of the AppWorks Gateway containers. Normally, you should have a local installation of Kubernetes, but it is also possible to run Kubernetes commands on certain cloud platforms. For example, on the Google Cloud Platform (GCP), you can use the Google Cloud Shell to run Kubernetes commands.

12.2 Deploying the AppWorks Gateway with Docker

Use this procedure to deploy the AppWorks Gateway in Docker.

To install AppWorks Gateway in a Docker container:

1. Go to OpenText My Support (https://knowledge.opentext.com) and search for AppWorks Gateway software, using the "#AppWorksGateway25.4SoftwareDownloads" quick link.
2. Download the `docker-compose.zip` file.
3. Extract the `docker-compose.zip` file to a location, for example, `C:\OpenTextCE\docker`. The contents of the `docker-compose` folder are as follows:
   - `.env`: Contains details of the AppWorks Gateway ,OTDS and PostgreSQL installations. In this file, you also configure the General, MDM and Mail settings for the AppWorks Gateway deployment.
   - `docker-compose.yml`
   - `install`: Add the mobile and desktop apps that you want to deploy with the container to this folder.
   - `pg`: Contains `config`, `env` and `secrets` folders. The `pg` folder enables you to add the details of your PostgreSQL database.
4. Navigate to the `pg\secrets` folder, and update the `PGPASSWORD` and `PGNEWPASSWORD` files.
   a. Update the `PGPASSWORD` file, replacing `postgresadminpassword` with the password for your database administrator.

      Note: The file starts with a sequence of asterisks and colons `*:*:*:*:`. Do not remove or alter this sequence.

   b. Update the `PGNEWPASSWORD` file, replacing `postgresuserpassword` with the password for a new user who will deploy the AppWorks Gateway database.

5. Navigate to the `pg\config` folder, and update the `PGNEWDATABASE` and `PGNEWUSER` files.

   a. Update the `PGNEWDATABASE` file, replacing `postgresdatabase` with the username of your database administrator.

   b. Update the `PGNEWUSER` file, replacing `postgresuser` with the username for a user who will deploy the AppWorks Gateway database.

6. In the `env` folder, update the `pg.env` file with the details of the PostgreSQL installation, for example, hostname, port and so on.

7. In the `.env` file, enter the details for the following sections:

   - **Port details**

     Update the `OTAWG_PORT` used by the Apache Tomcat installation, if required.

   - **Gateway Setup related environment variables**

     Update the AppWorks Gateway and OTDS settings. See "Updating the OTDS Settings" on page 73 and "Updating the AppWorks Gateway (awg) Settings" on page 73 for details.

   - **Environment variables to set various settings values**

     These are the **General**, **MDM** and **Mail** settings that appear in the AppWorks Gateway Administration UI. If you change the defaults here, you cannot change them in the Administration User Interface of the deployed Gateway. See "Updating the AppWorks Gateway (awg) Settings" on page 73for details.

8. Open a PowerShell session as an administrator and navigate to the `docker.compose` installation folder, for example, `C:\OpenTextCE\docker`.

9. Run the command `docker-compose up -d` to build the container.

10. Run `docker container ls` to list the docker containers, and check that the container is available and running. The name of the container is `docker-compose_otawg_1`.

11. In a browser, type `http://<hostname of the server>:<port number>` and sign in to the AppWorks Gateway Administration UI, using the credentials that you supplied in the `.env` file.

12. To view the logs run the command `docker logs container <container name> –f`.

13. To remove the container deployment, run the command `docker-compose down`.

12.3 Deploying the AppWorks Gateway with Helm

To install the AppWorks Gateway as a container through Helm, you require the following:

- The **appworks-gateway.zip** file containing Helm charts. Go to OpenText My Support (https://knowledge.opentext.com) and search for AppWorks Gateway software, using the "#AppWorksGateway25.4SoftwareDownloads" quick link.
- Access to the **AppWorks Gateway** container image: This installs the AppWorks Gateway software. This image is available from the OpenText Container Repository.
- Access to the **PostgreSQL** container image: This creates a database with a name you have supplied in your PostgreSQL installation.
- An AppWorks **init container** image: This initializes the AppWorks Gateway container. By default no mobile or desktop apps are installed with the container. You customize the AppWorks **init container** or build your own app container to include the mobile and desktop apps that you want to add to the AppWorks Gateway deployment. This image is available from the OpenText Container Repository. See "Building and Deploying an AppWorks Gateway Container with Custom apps" on page 67.

To download the appworks-gateway.zip file

1. Go to OpenText My Support (https://knowledge.opentext.com) and search for AppWorks Gateway software, using the "#AppWorksGateway25.4SoftwareDownloads" quick link.
2. Download the `appworks-gateway.zip` file.
3. Extract the `appworks-gateway.zip` file to a location, for example, `C:\OpenTextCE\appworks-gateway`. The contents of the `appworks-gateway` folder are as follows:
    - `Sample-Proxy-Rules`: Contains proxy rule templates to support the integration of various OpenText products. You can add the templates to the `Ruletemplate.JSON` file. When you deploy the AppWorks Gateway container, these rules appear in the **Proxy** section of the AppWorks Gateway Administration User Interface.
    - `templates`
    - `trustCertificates`: Add a certificate to this folder if you want to connect to a PostgreSQL database in Amazon RDS using SSL. See "Updating the Database Settings" on page 71 for further details.
    - `.helmignore`
    - `Chart.yaml`
    - `Ruletemplate.json`: Contains the proxy rules to add to the AppWorks Gateway container deployment. The `Ruletemplate.JSON` file contains several default rules. You can add further rules to the `Ruletemplate.JSON` file to include them in the AppWorks Gateway container deployment.

- `values.yaml`: Contains the General, MDM and Mail settings to use when deploying an AppWorks Gateway container. The `values.yaml` file also includes the details of the OTDS and database installations, and various other settings.

To confirm access to the container images

1. The `image` section of the `values.yaml` file contains the `registry` and `repository` details of the AppWorks Gateway images in the OpenText Container repository. See “Updating the Image Settings” on page 71.
2. Confirm that you can access the repository by entering the registry address in a browser. You must supply credentials to view the contents of the repository. Check with your Customer Support Representative for details.

12.3.1 Building and Deploying an AppWorks Gateway Container with Custom apps

AppWorks Gateway 25.4 enables you to build init container that extends the Gateway main container. This must be built each time you install or upgrade to a new release and this process increases the size of your custom init container. Alternatively, you can build your own apps container independently and deploy the apps to the Gateway.

Whether you want to build your own container or extend the init container, you can use the `awg-init-container.zip` file. Enter the “#AppWorksGateway25.4SoftwareDownloads” quick link in the My Support search box to locate the software. The contents of the `awg-init-container.zip` file are as follows:

- `apps`: This is the folder in which you place the desktop and mobile apps that you want to deploy with the AppWorks Gateway
- `Dockerfile`: You modify this file to include the names of the apps that you want to deploy
- `README`: This text file provides an explanation of the environment variables and exit codes used by the init container.

Note: You can create your own `Dockerfile` for building the `appsInitContainer` and follow the steps to deploy the apps. The sample `Dockerfile` is for reference, which you can download from the **AppWorks Gateway 25.4** folder and modify, as explained below.

To update the AppWorks Gateway image with your apps

1. Go to OpenText My Support (https://knowledge.opentext.com) and search for AppWorks Gateway software, using the “#AppWorksGateway25.4SoftwareDownloads” quick link.
2. Download and extract the `awg-init-container.zip` file to a location, for example, `C:\OpenTextCE\awg-init-container`.

3. Use one of the following approaches to deploy your apps to the Gateway:

   - Extend the init container to deploy your apps to the Gateway. Add the names of the apps that you added to the apps folder. For example, if your apps are called `starterApp1_0.0.1.zip`, and `starterApp2_0.0.2.zip`, then the `Dockerfile` will be as follows, with a separate **Copy** command for each file:

```
FROM artifactory.otxlab.net/otag/awg:23.3.0-22122020010729
MAINTAINER OpenText Corp.
ENV AWG_UNATTENDED_SETUP="true"
ENV WG_EXIT_ON_SETUP_COMPLETE="true"

# Unzip to auto-install location
RUN mkdir -p ./conf/install
COPY apps/starterApp1_0.0.1.zip ./conf/install
COPY apps/starter1App2_0.0.2.zip ./conf/install
```

     Note: This method to extend the main container for apps deployment is deprecated in AppWorks Gateway 22.3 and will be removed in future releases.

   - Build your own container to deploy your apps to the Gateway. Use the sample code below to create your own container to include the apps that you want to deploy to the Gateway. Replace the following values as required:

     – “Alpine Linux” is the base OS

     – “`starterApp1_0.0.1.zip`” is the file to be deployed to the Gateway

     – “8877” is the user code

     – “tomcat” is the username

```
FROM alpine:latest
MAINTAINER OpenText Corp.

RUN adduser -s /bin/bash -S tomcat


RUN mkdir /opt/awg

COPY ./configure.sh /opt/awg/configure.sh

RUN chown -R tomcat /opt/awg

RUN chmod +x /opt/awg/*.sh;
#Copying zip to temporary location. Later to mounted volume location using
script.
RUN mkdir -p /tmp/awg/otag

COPY apps/otag/offlineApp_0.0.1.zip /tmp/awg/otag

#create a directory with the tenant name for each tenant and copy the apps eg:
RUN mkdir -p /tmp/awg/<tenant-name>
#RUN mkdir -p /tmp/awg/tenant1
#COPY apps/tenant1/offlineApp_0.0.1.zip /tmp/awg/tenant1

# Directory mapped with volume
RUN mkdir -p /conf/install

USER tomcat
```

```
ENTRYPOINT ["/opt/awg/configure.sh"]
```

In the above code, uncomment the tenancy-related `RUN` and `COPY` commands to deploy apps for tenants.

4. Save the `Dockerfile`.
5. Open a PowerShell session as an administrator and navigate to the root folder of the `awg-init-container`.
6. Run the following command to build the image.

   `docker build -t <imagename>`
7. Run the `docker image ls` command to check that the container has been built:
8. Provide a tag number for the image, using the following command:

   `docker tag <imagename> <registryname>/<repositoryname>/<container_name>:<imagename>.<tagnumber>`

   where `<registryname>/<repositoryname>` are the details of your own registry.
9. Run the `docker image ls` command to check that the tagging has been successful.
10. Run the following command to push the new image to your registry:

    `docker push <registryname>/<repositoryname>/awg-init-container:<imagename>.<tagnumber>`
11. Confirm that the image now appears in your registry.

12.3.2 Deploying the AppWorks Gateway Container

To deploy the AppWorks Gateway container image with the desktop and mobile apps, you modify the parameters in a Helm chart file, then deploy the product using a Helm command.

Initially, deploy the AppWorks Gateway with a single replica. Afterwards, you can scale up the deployment to the required number of replicas. Update the `replicaCount` in the `values.yaml` file and run the helm upgrade command: `helm upgrade <deployment-name> appworks-gateway`. See "Updating the replicaCount setting" on page 75.

To deploy the AppWorks Gateway container:

1. Locate the `values.yaml` file that you extracted from the `appworks-gateway.zip` file. See "Deploying the AppWorks Gateway with Helm" on page 66 for details.
2. Update the `values.yaml` file as required. See"About the values.yaml file" on page 70 for details.
3. Open a PowerShell administrator session and navigate to the root directory, for example, `C:\OpenTextCE`.

4. Run the following command to install the AppWorks Gateway container.

   ```
   helm install <deployment name> appworks-gateway
   ```

5. Run the following command to monitor the status of the deployment:

   ```
   kubectl get pods -w
   ```

   A status of `1/1 Ready` indicates that the deployment is complete.

6. Open a browser and access the AppWorks Gateway using the URL specified in the `values.yaml` file. If you have enabled ingress, use the value you entered in the `host` setting.

7. Sign in to the AppWorks Gateway using the Username and Password details that you supplied in the `awg` section of the `values.yaml` file.

8. Check the **Installed** section of the **Admin Menu** to confirm that the desktop and mobile apps that you included in your deployment are present.

12.3.3 About the values.yaml file

The `values.yaml` file contains the following settings for deploying the AppWorks Gateway as a container:

12.3.3.1 Updating the Global Settings

The `values.yaml` file contains a number of settings that can be defined globally or locally. The global settings apply across OpenText products. The local settings apply to the AppWorks Gateway only. When the local setting is supplied in the `values.yaml`, it will be used in preference to the global setting.

imageSource
The local setting is the `registry` setting under `image`.

imagePullPolicy
The local setting is the `pullPolicy` setting under `image`.

imageTag
The local setting is the `tag` setting under `image`.

ingressAnnotations
The local settings are the `annotations` settings under `ingress`.

IngressEnabled
The local setting is the `enabled` setting under `ingress`.

IngressClassName
The local setting is the `IngressClassName` setting under `ingress`.

serviceType
The local setting is the `type` setting under `service`.

timezone
The local setting is the `timezone` setting.

otdsPublicUrl
The local setting is the `url` setting under `server` in the `otds` section. When you add a value for `otdsPublicUrl`, you must still add the details of the admin user and password to the `otds` section.

12.3.3.2 Updating the Image Settings

Initial registry and repository settings
These values refer to the AppWorks Gateway image in the OpenText registry.

awgInitContainer
To deploy the AppWorks Gateway container image without any desktop or mobile apps, or deploying your apps with your own container, keep the default settings for `registry`, `repository` and `tag`.

To deploy your desktop and mobile apps, on your AppWorks Gateway when extending the init container, do the following:

appsInitContainer
This is required when you are deploying apps to the Gateway using your own container. The`appsInitContainer` copies your desktop and mobile apps into a shared folder.

To deploy your desktop and mobile apps, on your AppWorks Gateway using your own container, do the following:

1. Build your own container with your apps and push it to your registry. See "Building and Deploying an AppWorks Gateway Container with Custom apps" on page 67.
2. Uncomment the `appsInitContainer` section and update the `registry`, `repository` and `tag` values with the details of the container that you pushed to your registry.

pullpolicy, imagepullsecrets
OpenText recommends that you do not modify these settings. If the images are not available locally they will be deployed from the OpenText registry according to the settings for `registry`, `repository` and `tag`.

12.3.3.3 Updating the Database Settings

AppWorks Gateway supports on premises and cloud deployments of PostgreSQL, Oracle and Microsoft SQL Server databases.

The cloud deployment is available for the Amazon Relational Database Service (Amazon RDS). For the cloud deployment you must first create the database in Amazon RDS, as follows:

1. In Amazon RDS, establish a PostgreSQL, Oracle and MsSQLserver database with a version of that is supported by AppWorks Gateway. Refer to the *AppWorks Gateway Release Notes* for details.

2. Using a database administration tool, such as pgAdmin for PostgreSQL, create a database instance to use with the AppWorks Gateway. See the settings required below.

vendor
Enter the database vendor. Valid values are “PostgreSql”, “Oracle” and “MsSqlServer”.

server
Enter the server `host` and `port` details of an on premises PostgreSQL database or a PostgreSQL database in Amazon RDS.

admin
Enter the `user`, `password` and `database` details for the database that will be created by `InitContainer`. You can keep the defaults or update with new values.

- If you provide the admin credentials during installation, the AppWorks Gateway will automatically create the user and the database (appworksdb).
- If you choose to create the database yourself, execute a `GRANT ALL ON SCHEMA public TO <appworksdb.user>;` command. For example, if the <*appworksdb.user*> is `customerUser` and the <*appworksdb.database*> is `customerAWGDB`, execute the following command in PostgreSQL interactive terminal (PSQL):

```
\c customerAWGDB
GRANT ALL ON SCHEMA public TO customerUser;
```

appworksdb
Enter a username, password, and database name. For an on premises PostgreSQL database, deploying the container will create an instance with these credentials in the PostgeSQL installation. Make sure that the `username` and `database` details are in lower case characters.

Notes

- For a cloud deployment of PostgreSQL in Amazon RDS, you must first create the database using a PostgreSQL database administration tool.
- For a cloud deployment of Azure PostgreSQL, you must first add the users to the database using a PostgreSQL database administration tool.

useSSL
The default value is false. Enter true if you want to connect to Amazon RDS over SSL.

trustDbCerts
The default value is false. Set **trustDbCerts** to true if you want the certificate to be downloaded automatically when connecting with Amazon RDS over SSL.

Notes

- If **useSSL** is true and **trustDbCerts** is false, place the certificate inside the `appworks-gateway\trustCertificates` folder to connect over SSL.

sslMode
: The **sslMode** setting is used for establishing a secure connection when connecting to an SSL enabled database. The supported values are "require", "verify-ca", and "verify-full". The default is "verify-ca". Use "require" or "verify-ca" for a Google Cloud SQL database instance.

12.3.3.4 Updating the OTDS Settings

server, admin, password
: Enter the details for the OTDS installation that you want to use with the AppWorks Gateway container deployment.

partition resource
: Enter the name of the OTDS `partition` to be used for the AppWorks Gateway container deployment.

customPartition
: Enter the names of one or more OTDS custom partitions to be used for AppWorks Gateway container deployment. For example:

```
customPartition: ['custom1,'custom2','custom3']
```

The names you enter for **customPartition** will be added to the list of access roles for the AppWorks Gateway partition in OTDS.

resource
: Enter the name of the OTDS `resource` to be used for the AppWorks Gateway container deployment.

12.3.3.5 Updating the AppWorks Gateway (awg) Settings

admin
: Enter values for the `newadminuser` and `newadminpassword`. You use these values to sign in to the AppWorks Gateway Administration User Interface. The password requirements depend on your OTDS global password policy.

externalurl
: Enter the URL that will be used to access the AppWorks Gateway Administration User Interface.

logging
: The default setting for logging is `info`. You can change to `trace`, `debug`, `warn` or `error`.

allowedOrigins
: To configure the AppWorks Gateway for CORS, enter a comma-separated list of web origins, for example:

```
cors:
    allowedOrigins: "https://example.com,https://app.example.com"
```

settings

This section enables you to determine the values for the General, MDM and Mail settings that are available in the AppWorks Gateway Administration User Interface. If the setting is not specified here. you can update it in the Administration User Interface. If you choose to set a value for a setting here, you cannot update it in the Administration User Interface of the deployed AppWorks Gateway. To update a setting, after you have deployed AppWorks Gateway as container, see "Upgrading a Deployed AppWorks Gateway Container" on page 79.

The following table lists the settings. For an explanation of each setting, see "AppWorks Gateway Settings" on page 43.

Table 12-1: General, MDM and Mail Settings

Setting	Data Type	Value if Required
wantSecureCookies	Binary	true or false
doNotProxyOTDS	Binary	true or false
anonymousAccessEnabled	Binary	true or false
forcePasswordBasedNonOauthAuthentication	Binary	true or false
disableOTDSAuth	Binary	true or false
otdsSSODisableByIP	String	Enter an IP address
listOfTenants	String	Enter a comma-separated list of tenants or * to permit all.
awgDeveloperMode	Binary	true or false
clientTrackingEnabled	true	true or false
remoteWipeTimeoutSeconds	String	300 seconds is the default value
remoteWipeFailureEmailRecipients	String	Enter an email address.
permitStorageOfPasswordsOnDevice	Binary	true or false
passwordOnDeviceExpirySeconds	Integer	604800 seconds is the default value
notifyDevicesOfOfflinePolicyChange	false	true or false
disableSharingAndDownloadingFiles	false	true or false
mailAuthenticationType	String	Choose Password or OAuth types. The default is Password authentication.
smtpReturnAddress	String	The email that is generated by the AppWorks Gateway will be sent from this email address

Setting	Data Type	Value if Required
smtpHost	String	Enter the host name or IP for the SMTP mail server
smtpPort	Integer	25 is the default value
smtpUsername	String	The username for connecting to the SMTP mail server
smtpPassword	String	The password for connecting to the SMTP mail server
smtpUseSSL	Binary	true or false
oathScope	String	Choose Google or Microsoft
oauthClientId	String	The client ID from the Google Cloud console or the Microsoft Azure Portal
oauthClientSecret	String	The client secret from the Azure portal for Google or Microsoft
enableAutoLogout	Binary	true or false
autoLogoutTimeIntervalSeconds	Integer	The minimum recommended value is 120 seconds
enableConcurrentSessionLimit	Binary	true or false
concurrentSessionLimitAdmin	Integer	The maximum number of concurrent administrator users
concurrentSessionLimitUser	Integer	The maximum number of concurrent regular users

12.3.3.6 Updating the replicaCount setting

This setting determines the number of AppWorks Gateway pods that you want to deploy. You may want to deploy multiple pods for the purposes of load balancing. Review the `ingress` settings if you set `replicaCount` to more than 1.

Initially, deploy the AppWorks Gateway with a single replica. Afterwards, you can scale up the deployment to the required number of replicas. Update the `replicaCount` in the `values.yaml` file and run the helm upgrade command: `helm upgrade <deployment-name> appworks-gateway`.

12.3.3.7 Updating the Proxy Settings

The proxy setting refers to the proxy rules that you can create in the AppWorks Gateway Administration User Interface. To add the proxy rules to the container deployment, update the `Ruletemplate.JSON` file, as follows:

1. Open the `Ruletemplate.JSON` file in a text editor. The `Ruletemplate.JSON` contains OTDS and eDOCS templates by default.
   a. To include the `Sample OTDS` rule, add the hostname of your OTDS server to the `replace` setting, under `urlRules`.
   b. To include the `Sample eDOCS RestAPI` rule, add the hostname of your eDOCS REST API Server to the `replace` setting, under `urlRules`.
2. Additional rules are available in the `Sample-Proxy-Rules` folder. To install a rule from this folder, open the file and copy the content into the `Ruletemplate.JSON` file.
3. Change the value for `automate` to true.
4. Save the `Ruletemplate.JSON` file.

12.3.3.8 About the System and Resource Settings

deploySecretsFromHelm
The default for this setting is true. When true, the passwords in the `values.yaml` file are stored in plain text format. To connect to a password managing service, for example, AWS Secrets Manager/Azure Key Vault, set `deploySecretsFromHelm` to false. See “Deploying AppWorks Gateway with External Secrets Operator” on page 80.

fipsModeEnabled
The default for this setting is false. To integrate the Gateway with the requirements of the Federal Information Processing Standards (FIPS), set `fipsModeEnabled` to true.

timezone
By default, the AppWorks Gateway server is deployed in UTC timezone. To deploy in a different timezone, set to a UTC time offset, for example, UTC-05.

readiness probe
OpenText recommends that you leave the `readiness probe` settings unchanged. They relate to the Kubernetes architecture and perform checks to determine when a container is ready to start accepting traffic.

liveness probe
OpenText recommends that you leave the `liveness probe` settings unchanged. They relate to the Kubernetes architecture and perform checks to determine when to restart a container.

service
OpenText recommends that you leave the `service` settings unchanged.

annotations
OpenText recommends that you only use this setting to provide service level custom annotations, for example:

```
annotations: ["example.com/baz :aux" , "example.com/foo: bar"]
```

ingress
Ingress provides load balancing functionality where you have deployed multiple pods by setting `replicaCount` to a number greater than 1. Update the `enabled` setting for load balancing. Also, if you want to access multiple AppWorks Gateway pods with a single URL, set `enabled` to true and enter a URL in the `host` setting. Update the `IngressClassName` with the `IngressClass` to be used to implement Ingress.

tls
To use HTTPS, set `enabled` to true, then provide values for the `secretName` settings.

resources
OpenText recommends that you leave the default settings unchanged. However, if you want to provide customer settings, uncomment the lines for `limits` and `requests`, and enter your preferred values.

customPodLabels
Add any required custom pod labels in key value pairs, for example, `#LabelName : AWGPodName`. A label enables you to identity a pod when automating the Gateway container deployment.

secretName
To create a secret with a custom secret name add the name under `secretName`, for example, `secretName: otag-awg`. To use an existing custom secret that applies to an existing namespace, provide the custom secret name. Alternatively, the Gateway will create a secret with the value you specify for **secretName**,

12.3.3.9 Integration with New Relic

New Relic is an application performance management (APM) tool. The following settings are mandatory to enable application monitoring of the AppWorks Gateway pod in a New Relic APM dashboard.

enabled
Set **enabled** to true to enable APM monitoring of the pod.

appName
Add the name you wish to see in an APM dashboard. The default is “APPWORKS GATEWAY”.

licenseKey
Enter a valid New Relic license key.

logFileName
: The default is “STDOUT”, With the default value, when the pod is running and the Kubernetes command is executed, the AppWorks Gateway logs can be viewed in the command console.

  To output the logs to a log file, enter a name for the log file, for example `filename.log`, and the file will be placed in the <*Tomcat_Home*>`\logs` folder on the server that is hosting the AppWorks Gateway.

logLevel
: The default is “info”. See the New Relic documentation for a list of the possible log levels.

If the AppWorks Gateway is deployed in a network-restricted environment where a proxy is required, complete the following settings:

proxyHost
: Enter the hostname of the proxy.

proxyPort
: Enter the port of the proxy.

proxyScheme
: Enter “HTTP” or “HTTPS” for the protocol of the proxy.

12.3.3.10 Integration with OpenText Managed Vault

You can use OpenText Managed Vault to store the OpenText AppWorks Gateway passwords in encrypted form.

enabled
: Set **enabled** to true to enable integration with OpenText Managed Vault . The default value is false.

type
: Set to “Opentext”. This is the only value for type that is currently supported.

host
: The host name of the vault. This is provided by OpenText cloud services.

port
: The port for the vault host. This is provided by OpenText cloud services.

namespace
: The namespace in which the vault is configured. This is provided by OpenText cloud services.

secrets
: The secrets path that is stored on the vault. This is provided by OpenText cloud services.

authConfigPath
The Kubernetes service account token path for which the vault has access. This is provided by OpenText cloud services.

authPath
This role authorizes the service account for the given namespace. This is provided by OpenText cloud services.

role
Type “jwt” for JSON web token. This is the format that is used to authenticate the pods.

serviceaccount
The service account that has been created in Managed Vault. This is provided by OpenText cloud services.

12.3.4 Upgrading a Deployed AppWorks Gateway Container

You can upgrade a AppWorks Gateway deployment to update the General, MDM and Mail settings. You can also increase the value for `replicaCount` to add further pods to the deployment and update the `ingress` settings to add load balancing.

To update a deployed AppWorks Gateway container

1. Update the settings in the `values.yaml` file, as required. A setting that you specify in the `values.yaml` file cannot be altered in the AppWorks Gateway Administration User Interface.
2. Run the following upgrade command:

   ```
   helm upgrade <deployment name> appworks-gateway
   ```

3. Run the following command to delete the pod. You must run the command for all pods in the cluster. The pod is automatically recreated.

   ```
   kubectl delete pod <pod name>
   ```

   Tip: The command `kubectl get pods -w` displays the pod name and the status of the pod.

12.3.5 Checking the AppWorks Gateway Deployment Status

For the OpenText AppWorks Gateway container deployment, the `init container` process deploys the AppWorks Gateway with any desktop and mobile apps

If the deployment fails to initialize `init container`, the status shows `Error` or `CrashloopBack`. Run the following command to see the log files:

```
kubectl logs <podname> -c awg-init-container
```

Run the following command to show the standard container log files:

```
kubectl logs <podname> -f
```

12.4 Deploying AppWorks Gateway with External Secrets Operator

The External Secrets Operator is a Kubernetes operator that integrates external secret management systems. You can manage and sync secrets from external secret management systems into your cluster.

Note: If choose to use your own `customSecretName` and want to have all init-container and main container secrets in a single external secret file, then in the following sections, instead of creating two `ExternalSecret` manifests, create a single manifest file and add both container secrets. Ensure that the spec.target.name in the `ExternalSecret` manifest file is the same as the `customSecretName` in the AppWorks Gateway `values.yaml` file.

12.4.1 Configuring AppWorks Gateway to use AWS Secrets Manager

To create and configure the secrets, refer to the AWS Secrets Manager documentation

To configure AppWorks Gateway to use AWS Secrets Manager:

1. Create the `SecretStore` using the following YAML file.

```
apiVersion: external-secrets.io/v1
kind: SecretStore
metadata:
  name: <secret-store-name>
spec:
    provider:
        aws:
          service: SecretsManager
          region: <region-name>
          auth:
            jwt:
              serviceAccountRef:
                name: <service-account-name> #external secret service account name
                namespace: <name-space> #namespace
```

   The will be used to create a connection between the external secret manager and the AWS secret manager, that is, to create a Secret Store.

2. Create the `ExternalSecrets` using following YAML files.

   - `appworks-init-container-secrets.yml`:

```
apiVersion: external-secrets.io/v1
kind: ExternalSecret
metadata:
  name: <external-secret-name>
spec:
  secretStoreRef:
    name: <secret-store-name>
    kind: SecretStore
  target:
    name: awg-init-container-secret
  data:
    - secretKey: AWG_GATEWAY_NEW_ADMIN_PASSWORD
      remoteRef:
        key: <AWS-secret-name>
        property: <key-name> #Key name provided in the AWS Secret for the
specific secretKey
    - secretKey: AWG_DB_PASSWORD
      remoteRef:
        key: <AWS-secret-name>
        property: <key-name>
    - secretKey: AWG_OTDS_ADMIN_PASSWORD
      remoteRef:
        key: <AWS-secret-name>
        property: <key-name>
    #In a multi-tenancy environment, you must supply values for each tenant
    #Uncomment below if deploying AWG with multi-tenancy
    #<TENANT-NAME> is the value of tenants.name from values.yaml
#     - secretKey: AWG_<TENANT-NAME>_GATEWAY_NEW_ADMIN_PASSWORD
#       remoteRef:
#         key: <AWS-secret-name>
#         property: <tenant-key-name>
#     - secretKey: AWG_<TENANT-NAME>_DB_PASSWORD
#       remoteRef:
#         key: <AWS-secret-name>
#         property: <tenant-key-name>
#     - secretKey: AWG_<TENANT-NAME>_OTDS_ADMIN_PASSWORD
#       remoteRef:
#         key: <AWS-secret-name>
#         property: <tenant-key-name>
```

- `appworks-gateway-secrets.yml`

```
apiVersion: external-secrets.io/v1
kind: ExternalSecret
metadata:
  name: <external-secret-name>
spec:
  secretStoreRef:
    name: <secret-store-name>
    kind: SecretStore
  target:
    name: otag-awg-secret
  data:
    - secretKey: AWG_DB_PASSWORD
      remoteRef:
        key: <AWS-secret-name>
        property: <key-name> #Key name provided in the AWS Secret for the
specific secretKey
    - secretKey: AWG_OTAG_SMTP_PASSWORD
      remoteRef:
        key: <AWS-secret-name>
        property: <key-name>
    - secretKey: AWG_OTAG_OAUTH_CLIENT_SECRET
      remoteRef:
        key: <AWS-secret-name>
        property: <key-name>
    #In a multi-tenancy environment, you must supply values for each tenant
    #Uncomment below if deploying AWG with multi-tenancy
    #<TENANT-NAME> is the value of tenants.name from values.yaml
#     - secretKey: AWG_<TENANT-NAME>_DB_PASSWORD
#       remoteRef:
#         key: <AWS-secret-name>
#         property: <tenant-key-name>
#     - secretKey: AWG_<TENANT-NAME>_OTAG_SMTP_PASSWORD
#       remoteRef:
#         key: <AWS-secret-name>
#         property: <tenant-key-name>
#     - secretKey: AWG_<TENANT-NAME>_OTAG_OAUTH_CLIENT_SECRET
#       remoteRef:
```

```
#        key: <AWS-secret-name>
#        property: <tenant-key-name>
```

Note: The `secretKey` is the name that AppWorks Gateway uses to identify secrets. Do not change this name in the Secret YAML files.

3. Apply the `SecretStore` manifest to configure access to the external secret management system.

```
kubectl apply -f secret-store.yaml
```

4. Check that the connection is successful. You receive a "VALID" status.

```
kubectl get secretstore
```

5. Apply the `ExternalSecret` manifest file to automatically create and sync a Kubernetes Secret in your cluster.

```
kubectl apply -f appworks-init-container-secrets.yaml
kubectl apply -f appworks-gateway-secrets.yaml
```

6. Execute the following command to check that the secrets are synced. If they are synced, you receive a "SecretSynced" status

```
kubectl get externalsecret
```

12.4.2 Configuring AppWorks Gateway to use Azure Key Vault

To create and configure the secret key, refer to the Azure Key Vault documentation:

To configure AppWorks Gateway to use Azure Key Vault:

1. Create the `SecretStore` using the `secret-store.yaml` file.

```
apiVersion: external-secrets.io/v1
kind: SecretStore
metadata:
  name: <az-secret-store> #Name for SecretStore
spec:
  provider:
    azurekv:
      # azure tenant ID, see: https://docs.microsoft.com/en-us/azure/active-
directory/fundamentals/active-directory-how-to-find-tenant
      tenantId: ""
      # URL of your vault instance, see: https://docs.microsoft.com/en-us/azure/key-
vault/general/about-keys-secrets-certificates
      vaultUrl: ""
      authSecretRef:
        # points to the secret that contains
        # the azure service principal credentials
        clientId:
          name: <azure-secret-name> #name of the secret created with clientID and
clientSecret.
          key: ClientID
        clientSecret:
          name: <azure-secret-name> #name of the secret created with clientID and
clientSecret.
          key: ClientSecret
```

This will be used to create a connection between the external secret manager and the AWS secret manager, and will create a Secret Store

2. Create the `ExternalSecrets` using the following YAML files.

- `appworks-init-container-secrets.yaml`

```
apiVersion: external-secrets.io/v1
kind: ExternalSecret
metadata:
  name: <az-init-external-secret> #Name for the external secret for init-
container
# Both init and main container external secrets should be different
spec:
  secretStoreRef:
    name: <secret-store-name>
    kind: SecretStore
  target:
    name: awg-init-container-secret
   #Below "key" are the secret name which we created in key vault for
respective secretKey's
  data:
    - secretKey: AWG_GATEWAY_NEW_ADMIN_PASSWORD
      remoteRef:
        key: <keyvault-secret-name>
    - secretKey: AWG_DB_PASSWORD
      remoteRef:
        key: <keyvault-secret-name>
    - secretKey: AWG_OTDS_ADMIN_PASSWORD
      remoteRef:
        key: <keyvault-secret-name>
    - secretKey: AWG_DB_ADMIN_PASSWORD
      remoteRef:
        key: <keyvault-secret-name>
    #In a multi-tenancy environment, you must supply values for each tenant
    #Uncomment below if deploying AWG with multi-tenancy
    #<TENANT-NAME> is the value of tenants.name from values.yaml
#    - secretKey: AWG_<TENANT-NAME>_GATEWAY_NEW_ADMIN_PASSWORD
#      remoteRef:
#        key: <keyvault-tenant-secret-name>
#    - secretKey: AWG_<TENANT-NAME>_DB_PASSWORD
#      remoteRef:
#        key: <keyvault-tenant-secret-name>
#    - secretKey: AWG_<TENANT-NAME>_OTDS_ADMIN_PASSWORD
#      remoteRef:
#        key: <keyvault-tenant-secret-name>
#    - secretKey: AWG_<TENANT-NAME>_DB_ADMIN_PASSWORD
#      remoteRef:
#        key: <keyvault-tenant-secret-name>
```

- `appworks-gateway-secrets.yaml`

```
apiVersion: external-secrets.io/v1
kind: ExternalSecret
metadata:
  name: <az-main-external-secret> #Name for the external secret for main-
container
# Both init and main container external secrets should be different
spec:
  secretStoreRef:
    name: <secret-store-name>
    kind: SecretStore
  target:
    name: otag-awg-secret
   #Below "key" are the secret name which we created in key vault for
respective secretKey's
  data:
    - secretKey: AWG_DB_PASSWORD
      remoteRef:
        key: <keyvault-secret-name>
    - secretKey: AWG_OTAG_SMTP_PASSWORD
      remoteRef:
        key: <keyvault-secret-name>
    - secretKey: AWG_OTAG_OAUTH_CLIENT_SECRET
      remoteRef:
        key: <keyvault-secret-name>
    #In a multi-tenancy environment, you must supply values for each tenant
    #Uncomment below if deploying AWG with multi-tenancy
    #<TENANT-NAME> is the value of tenants.name from values.yaml
#     - secretKey: AWG_<TENANT-NAME>_DB_PASSWORD
#       remoteRef:
#         key: <keyvault-tenant-secret-name>
#     - secretKey: AWG_<TENANT-NAME>_OTAG_SMTP_PASSWORD
#       remoteRef:
#         key: <keyvault-tenant-secret-name>
#     - secretKey: AWG_<TENANT-NAME>_OTAG_OAUTH_CLIENT_SECRET
#       remoteRef:
#         key: <keyvault-tenant-secret-name>
```

Note: The `secretKey` is the name that AppWorks Gateway uses to identify secrets. Do not change this name in the Secret YAML files.

3. Apply the `SecretStore` manifest to configure access to the external secret management system:

   ```
   kubectl apply -f secret-store.yaml
   ```

4. Check that the connection is successful. You receive a "VALID" status:

   ```
   kubectl get secretstore
   ```

5. Apply the `ExternalSecret` manifest file to automatically create and sync a Kubernetes Secret in your cluster:

   ```
   kubectl apply -f appworks-init-container-secrets.yaml
   kubectl apply -f appworks-gateway-secrets.yaml
   ```

6. Execute the following command to check that the secrets are synced. You receive a "SecretSynced" status:

   ```
   kubectl get externalsecret
   ```

12.5 Deploying the AppWorks Gateway Container for Multi-Tenancy

A multi-tenant AppWorks Gateway installation enables you to create multiple AppWorks Gateway instances in a container each with their own database and OTDS configuration details. Each tenant installation can have a separate administrator, with the administrator of the primary AppWorks Gateway having overall control.

In the `values.yaml` file, you specifiy the details for each tenant, for example the tenant name, the database details and so on.

Before you update the values.yaml file, follow the steps in "Deploying AppWorks Gateway with External Secrets Operator" on page 80.

- In "Configuring AppWorks Gateway to use AWS Secrets Manager" on page 80, you add new sections to the `appworks-init-container-secrets.yml` and `appworks-gateway-secrets.yml` files for each tenant.
- In "Configuring AppWorks Gateway to use AWS Secrets Manager" on page 80 you create the key value pairs for each tenant.

To update the values.yaml for multiple tenants

1. In the `values.yaml` file, locate the `listOfTenants` setting. The setting is in the `awg` section.
2. Enter the names of the required tenants: for example:

   ```
   listOfTenants: "Tenant1", "Tenant2"
   ```

3. Locate the `tenants` section, and uncomment.
4. For each tenant, provide a name and complete the `database`, `OTDS`, and `awg` sections:

   - See "Updating the Database Settings" on page 71.
   - See "Updating the OTDS Settings" on page 73
   - See "Updating the AppWorks Gateway (awg) Settings" on page 73

5. For the `proxy configuration` section, create a `proxy.json` file with the tenant name, and place it inside the proxy folder in the helm charts. Then, in the `values.yaml` file, update the file path in the proxy setting for the tenant, for example:

   ```
   #    proxy:
   #      automate: false
   #      filePath: proxy/tenant1.json
   ```

   Note: To create a new proxy file, refer to the `Rulestemplate.json` sample proxy file in the `proxy` folder in helm charts.

6. Repeat the steps for each tenant. For example, if you want to support two tenants, "tenant1" and "tenant2", do the following:

   a. Enter "tenant1" for the `name`, and add the details for the `database`, `OTDS`, `awg` and `proxy configuration` sections.
   b. Copy the content from the tenant name down to and including the settings for `proxy`.
   c. Paste the content after the proxy `filePath` setting.
   d. Enter "tenant2" for the name, and update the details for the `database`, `OTDS`, `awg` and `proxy configuration` sections.

12.6 Migrating from AppWorks Gateway Classic to Container

You can migrate OpenText AppWorks Gateway from the non-container (classic) version to the container version. You cannot upgrade AppWorks Gateway and migrate at the same time. This means that to migrate from a an earlier classic version of the AppWorks Gateway to a later container version, you must first upgrade the AppWorks Gateway to the target version. For example, to migrate to 25.4 container from an earlier classic release, first upgrade to 25.4, then migrate from 25.4 classic to 25.4 container.

To migrate AppWorks Gateway from classic to container

1. Update the `values.yaml` file with the details of your current system, as follows:

   - Your database. See "Updating the Database Settings" on page 71. You must use the same database in the container as you used in classic.
   - Your OTDS settings. See "Updating the OTDS Settings" on page 73. You must use the same OTDS details in the container as you used in classic.
   - Your AppWorks Gateway settings. See "Updating the AppWorks Gateway (awg) Settings" on page 73.

2. In OTDS, in the **OTDS Admin** page, add the redirect URL for the container to the **Redirect URLs** tab for the OTAG OAuth client.

12.7 Updating the Ingress ConfigMap for X-Forwarded-For HTTP headers

You may not want to see the OTDS login details in the browser URL, when your users log in to the AppWorks Gateway. To prevent this, update the Ingress ConfigMap as follows:

1. Edit the Ingress `configmap.yaml` file using the command: `kubectl edit configmap <ingress-controller-name> -n <ingress-namespace>`
2. Add the following to the `data` section:

```
data:
  compute-full-forwarded-for: "true"
  forwarded-for-header: X-Forwarded-For
  real-ip-header: X-Forwarded-For
  use-forwarded-headers: "true"
```

3. Save the `configmap.yaml` and restart Ingress, with the command: `kubectl rollout restart deployment <ingress-controller-name> -n <ingress-namespace>`

   The `configmap.yaml` should look similar to the following:

```
apiVersion: v1
data:
  compute-full-forwarded-for: "true"
  forwarded-for-header: X-Forwarded-For
  real-ip-header: X-Forwarded-For
  use-forwarded-headers: "true"
kind: ConfigMap
metadata:
  annotations:
    meta.helm.sh/release-name: ingress
    meta.helm.sh/release-namespace: default
  creationTimestamp: "2025-03-14T05:56:28Z"
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/instance: ingress
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx
    app.kubernetes.io/version: 1.12.0
    helm.sh/chart: ingress-nginx-4.12.0
  name: ingress-ingress-nginx-controller
  namespace: default
  resourceVersion: "24069"
  uid: cab766bd-4741-41f0-8c52-f99f0d8529a1
```

Chapter 13

Frequently Asked Questions

The following FAQs provide advice on additional setup options, and basic troubleshooting information.

How do I set the debug log level in the AppWorks Gateway?

The OpenText AppWorks Gateway provides information in the `gateway.log` file, which is located in the <*Tomcat_Home*>`\logs` directory. The log level for the AppWorks Gateway is set via a `log4j2.xml` file in the `gateway\web-inf\classes` folder.

To set the log level to a higher level, for example, debug, take the following steps:

1. Stop the Apache Tomcat service.
2. Open the `log4j2.xml` file in the `gateway\web-inf\classes` folder.
3. Search for the following entry in the file:

```
<Root level="info">
     <AppenderRef ref ="OTAG"/>
</Root>
```

4. Change the Root level setting from `info` to `debug`.
5. Save the `log4j2.xml` file.
6. Start the Apache Tomcat service.

   In an AppWorks Gateway multi-tenant environment, the log file includes information for the primary installation and all tenants.

! **Important**
When debug-level or trace-level logging is enabled, sensitive configuration data may be written to the log files. Debug-level and trace-level logs should be securely disposed of when they are no longer required. Return the log level to its lower level when you have completed any troubleshooting tasks.

How can I troubleshoot the OpenText AppWorks Gateway?

The OpenText AppWorks Gateway provides information in the `gateway.log` file, which is located in the <*Tomcat_Home*>`\logs` directory. Set the log level to a higher level, as described above.

The OpenText AppWorks Gateway proxy also logs information in the `gateway.log` file when the log level is set to debug.

Information is also available in the default Catalina logs such as `catalina_`*`<date>`*`_.log` and `localhost-access_logs.`*`<date>`*`.txt`.

For debugging, first check the Catalina log for errors, and if necessary check the `localhost-access_logs.<date>.txt` because they record each request. This allows you to confirm timing and response codes for each request when nothing appears in the logs.

Commonly reported issues revolve around OpenText Directory Services (OTDS) configuration. On the initial OpenText AppWorks Gateway configuration, there is a chance that the OTDS configuration will fail either due to port conflicts, memory issues, resource conflicts with existing OTDS data, or failure to communicate with an external OTDS due to firewall blocking requests or mistyped configuration information. The Catalina logs will show exceptions in the OTDS configuration. For more information, review the OTDS logs which can provide additional information. These logs can be found in the *<Tomcat_Home>*`\logs` directory on the Tomcat server used to host your OTDS installation. The OTDS log files are also available in the Directory Services web client. Errors relating to port conflicts and resource name conflicts will be clearly indicated in the logs. Other error cases may require further investigation.

Beyond initial configuration of the OpenText AppWorks Gateway, there are very few error scenarios that would be indicated in the logs. The primary issue that can occur relates to authentication errors. If the user has entered a correct password, but cannot authenticate, these issues will not be captured in the log files. The following steps are recommended to troubleshoot authentication errors.

1. Review the OpenText AppWorks Gateway node information to verify the database and OTDS connections. Sign in to the OpenText AppWorks Gateway and click **Overview** from the **About** section of the **Admin Menu**.
2. Double check that the credentials and server URL were typed correctly. This is often the problem.
3. Try connecting using a different client.
4. Try using the fully qualified name. For example. <*username*>@otag for internally created users, <*username*>@<*domain*> for OTDS synced users.
5. Sign in to the Directory Services web client and check that the user's account has not expired or been locked.
6. Add a new user in the OpenText AppWorks Gateway. Then, sign in to the Directory Services web client and verify that the new user was added to the otag partition.

How can I check the status of the AppWorks Gateway connections?

Open a browser window and navigate to `http://gateway:port/v3/admin/status`. The following information is returned, showing the status of the connection to OTDS and to the database:

```
{"dbSetup":true,"dbConnected":true,"otdsSetup":true,"otdsConnected":true,"developerMode":
true}
```

How can I troubleshoot the status of my apps, services, components, or connectors?

The OpenText AppWorks Gateway provides information in the `gateway.log` file which is located in the <*Tomcat_Home*>`\logs` directory. Your feature might also have a service log that is also located in this directory.

How can I configure AppWorks Gateway to use an HTTP forward proxy?

In AppWorks Gateway 16.1 and above, the Gateway is capable of making external HTTP calls via an HTTP forward proxy, using JAVA_OPTS.

A good example of this is for the routing of Firebase Cloud Messaging (FCM) communications when the environment within which the Gateway is installed only permits outbound HTTP communication to the public Internet via an HTTP proxy.

The example settings shown below are from a Linux installation of the AppWorks Gateway.

1. Stop the Apache Tomcat service.
2. Open the `setenv.sh` script in a text editor.
3. Add the following values and edit the hostnames and ports to match your environment.

   - `export CATALINA_OPTS="$CATALINA_OPTS -Dhttp.proxySet=true"`
   - `export CATALINA_OPTS="$CATALINA_OPTS -Dhttp.proxyHost=proxy-syd. cloud.opentext.com"`
   - `export CATALINA_OPTS="$CATALINA_OPTS -Dhttp.proxyPort=80"`
   - `export CATALINA_OPTS="$CATALINA_OPTS -Dhttps.proxyHost=proxy-syd. cloud.yourserver.com"`
   - `export CATALINA_OPTS="$CATALINA_OPTS -Dhttps.proxyPort=80"`
   - `export CATALINA_OPTS="$CATALINA_OPTS -Dhttp.nonProxyHosts= \"localhost|127.*|syd-awgw-p01|syd-awgw-p01.appworks.cloud. yourserver.com|syd-awdba-p01.appworks.cloud.yourserver.com\""`

   Where:

   - `Dhttp.proxyHost` is the address of the HTTP Proxy Server
   - `Dhttp.proxyPort` is the HTTP listening port for the HTTP Proxy
   - `Dhttps.proxyPort` is the HTTPS listening port for the HTTP Proxy
   - `Dhttp.nonProxyHosts` contains the short name and FQDN of the AppWorks Server `syd-awgw-p01` and `syd-awdba-p01.appworks.cloud.yourserver.com` is the OTDS Server used by AppWorks

We must NOT proxy requests via the HTTP Forward Proxy to:

- The Gateway itself.
- OpenText Directory Services (OTDS).

4. Save the `setenv.sh` file and start the Apache Tomcat service.

Tips

- Select the **Activate Push Notification Logging** check box in the **Notifications** section of the AppWorks Gateway, and then check the `fcm.log` to confirm that Firebase Cloud Messaging communications are being sent correctly.
- Use cURL, via a terminal session. to check the server host itself can connect to FCM via the HTTP Forward proxy. This should return an HTTP 200 response (adjust the proxy address and port to match your environment) `curl -v -I -x proxy-syd.cloud.opentext.com:80 https://fcm-http.googleapis.com`.

How can the AppWorks Desktop Client authenticate the user when the environment includes a Web proxy server?

When you install an AppWorks Desktop Client, such as eDOCS InfoCenter, an initial authentication request is made from the client machine to OTDS. If your environment includes a Web proxy server to handle web requests, you need to set the `HTTP_PROXY` and `HTTPS_PROXY` environment variables for the operating system to the address of the proxying server, for example, `HTTP_PROXY=`*`<http://proxy.mycompany.com:8080/>`*.

If you do not set the environment variables, then the authentication request from the AppWorks Desktop Client to OTDS will fail.

Note: For further details of the AppWorks platform and technologies, see the Technical White Paper, available from the AppWorks Quick Start page of the AppWorks Developer web site (https://developer.opentext.com).

Issue with logging in to desktop and mobile clients after upgrading Directory Services

To use OAuth 2.0 with Directory Services (OTDS), an OAuth client must be registered in OTDS.

By default, the AppWorks Gateway adds an OAuth client to OTDS, with the **Grant refresh token (when protocol permits)** check box selected. When you upgrade OTDS to 16.6, we recommend that you confirm that the **Grant refresh token (when protocol permits)** check box is still selected. If the check box is not selected a desktop or mobile client does not receive the refresh token on login and the user will therefore be asked to log in again, once they have logged out.

To check the **Grant refresh token (when protocol permits)** setting, do the following:

1. In OTDS, from the web administration menu, click **OAuth Clients** .
2. Select the OAuth Client for OTAG and click **Advanced**.
3. Select the **Grant refresh token (when protocol permits)** check box and click **Save**.

For further information, see the chapter on OAuth Clients in the *OpenText Directory Services - Installation and Administration Guide (OTDS-IWC)*.

How do I avoid timeout exceptions when uploading files through Gateway clients?

By default the socket timeout is set to 1 minute. To extend this period, and avoid timeout exceptions when uploading large files through Gateway clients, do the following:

For Windows:

Navigate to the <*Tomcat_Home*>\bin folder and double-click the Tomcat executable file to open the **Apache Tomcat Properties** dialog box. In the **Java Options** panel add the following to extend the timeout value to, for example, 2 minutes:

```
-Dotag.proxy.socket.timeout=120000

-Dotag.proxy.http.timeout=120000
```

For Linux:

Navigate to the <*Tomcat_Home*>\bin folder and add the following to the `setenv.sh` file to extend the timeout value to, for example, 2 minutes:

```
export JAVA_OPTS="$JAVA_OPTS -Dotag.proxy.socket.timeout=120000"

export JAVA_OPTS="$JAVA_OPTS -Dotag.proxy.http.timeout=120000"
```

Password-based authentication is deprecated by all email providers

If you still want to use password-based authentication, follow the steps below for a Gmail account.

1. Enable 2FA on your Gmail account.
2. Generate an app-specific password from here: https://myaccount.google.com/apppasswords.
    a. In your Google Account settings, in the **Security** section, navigate to **App Passwords**.
    b. Generate a new app-specific password, and save this password for use in the AppWorks Gateway mail settings.
3. In the AppWorks Gateway, go to **Mail Settings** and enter the following:

- **SMTP Host –** `smtp.gmail.com`
- **SMTP Port –** 587
- **SMTP Username –** Your complete Gmail address.
- **SMTP User Password –** Your 16–digit app-specific password.
- **SMTP Use SSL –** true