Installation Guide

Version: 8.2

Revised: June 2021

Copyright and Trademark Notices

Copyright © 2021 SailPoint Technologies, Inc. All Rights Reserved.

Contents

Install and Deploy SailPoint IdentityIQ

Use the following information to install and deploy SailPoint IdentityIQ on your application server. After IdentityIQ is deployed it must be configured to work within your enterprise. See System Setup to continue with your deployment of SailPoint IdentityIQ.

See the System Configuration documentation to continue with your deployment of SailPoint IdentityIQ.

During the installation and deployment procedure you must deploy a new Web application in your application server and create a new database and modify its schema in a database server instance. Ensure that you have the required authorization credentials before you begin the installation and deployment process. The IdentityIQ application and the IdentityIQ database can reside on the same server.

Each application server instance may only contain one instance of IdentityIQ in a given cluster, using the same database. There are implicit and explicit methods used when defining an application to an application server. Implicit methods include placing a `.war` file or directory in a well-defined location, for example, the `webapps` directory in Tomcat. Explicit methods include adding configuration. For example, In Tomcat, this configuration can be in any number of places such as, `conf/server.xml` or `conf/Catalina/<hostname>`. It is very important that the application only be defined once so that it will only be instantiated once.

The installation and deployment process contains the following parts:

- Download and expand the installation files. See Download and Expand the Installation Files.
- Configure the number of extended and searchable attributes allowed for your environment. See Configure the Number of Extended and Searchable Attributes Allowed.
- Create the database and tables required for IdentityIQ. SeeCreate the IdentityIQ Database and Tables.
- Configure IdentityIQ to connect to its database. See Configure the IdentityIQ Installation
- Access IdentityIQ to continue with the configuration for your enterprise. See See Open IdentityIQ.
- Refer to Advanced Installation Information for additional information on how to deal with specific application server and database server environments, and for integration requirements for some external systems.

Supported Platforms

Operating Systems

> **Linux Support:** The distributions and versions of Linux highlighted below have been verified by IdentityIQ Engineering, but any currently available and supported distributions and versions of Linux will be supported by SailPoint. Implementers and customers should verify that the distribution and version of Linux of choice is compatible with the application server, database server, and JDK also being used.

- SUSE Linux Enterprise Server 15 and 12
- Windows Server 2016 and 2019
- CentOS 8.3 and 7.9

Application Servers

- Apache Tomcat 9.0 and 8.5
- Oracle WebLogic 14c and 12cR2
- IBM WebSphere 9.0
- JBoss EAP 7.3 and 7.2
- IBM WebSphere Liberty 20.0 and 19.0

Databases (On Site)

- IBM DB2 11.5
- MySQL 5.7 and 8.0
- MS SQL Server 2019 and 2017
- Oracle 19c

Cloud Platforms

- AWS EC2
- AWS Aurora
- AWS RDS (MySQL, MS SQL, Oracle)
- Azure (VM, Azure SQL)
- Google Cloud Platform - Google Compute Engine

Java Platform

- Sun, Oracle or IBM JDK 1.8 (8), JDK 11 for all application servers
- OpenJDK11 is now supported on all structures, but we have specifically tested against Adopt OpenJDK 11 for Windows and Red Hat OpenJDK 11 for Linux.

JDKs are supported on 8 as needed by the specific application servers listed above. 6 and 7 are no longer supported.

Browsers

- Google Chrome Latest Version
- Internet Explorer New Chromium version of Edge
- Safari 14
- Firefox Latest Version

Mobile User Interface OS/Browser Support

- Android 11 on Chrome
- iOS 14 (beta) and 13 using Safari

Languages

- Brazilian Portuguese
- Danish
- Dutch
- English
- French
- French Canadian
- German
- Italian
- Japanese
- Korean
- Norwegian
- Polish
- Portuguese
- Simplified Chinese
- Spanish
- Swedish
- Traditional Chinese
- Turkish

Special Considerations

Special Java Considerations

JVM Arguments

To support connectivity to managed systems through a proxy server, use the Java system properties listed below to configure the proxy connectivity. The use of these system properties is described in the java.net Networking Properties documentation that accompanies the Java SDK.

- http.proxyHost
- http.proxyPort
- http.proxyUser
- http.proxyPassword
- http.nonProxyHosts
- https.proxyHosts
- https.proxyPort
- https.proxyUser
- https.proxyPassword

Consult the documentation for the application server in use to determine the method for adding Java system properties to the environment. As an example for Apache Tomcat, define a value for the `JAVA_OPTS` environment variable in `bin\catalina.bat` or `bin/catalina.sh` of your Tomcat installation.

Special Reporting Considerations

Configure Jasper to Export Reports

To modify the delimiter used in CSV report exports, create a file named jasperreports.properties that contains

```
net.sf.jasperreports.export.csv.field.delimiter=;
```

and add a Java system property using the method appropriate for the application server in use named `net.sf.jasperreports.properties` that contains a value of the full path to the jasperreports.properties file. This is often configured in the startup script for the application server by modifying the `JAVA_OPTS` environment variable, but can be configured in the administrative user interface for some application servers.

Use Custom Fonts with JasperReports Font Extensions

IdentityIQ uses JasperReports to render some reports. The live reports do not use JasperReports for rendering. JasperReports uses a specially packaged jar file known as a Font Extension to embed custom fonts in reports, for example, fonts not natively available on the host operating system. Creating a Font Extension involves editing an XML file and creating a jar archive file containing the configuration and font files.

1. Assemble all the font files in a new directory. There may be multiple files depending on all the different styles available to the font. For example, your font may have plain, bold, bold-italic, and italic versions.
2. Create a new XML file called `fonts.xml` in the same directory with the following structure.

Replace sections between square brackets [ ] with the appropriate information.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.s-
pringframework.org/dtd/spring-beans.dtd">
<beans>
    <bean id="[unique name of font family e.g. 'myFontFamily']" class-
s="net.sf.jasperreports.engine.fonts.SimpleFontFamily">
        <property name="name" value="[font name as referenced in jasper reports]"/>
        <property name="normal" value="[file name of normal font]"/>
        <property name="bold" value="[file name of bold font]"/>
        <property name="italic" value="[file name of italic font]"/>
        <property name="boldItalic" value="[file name of bold-italic font]"/>
        <property name="pdfEncoding" value="Identity-H"/>
        <property name="pdfEmbedded" value="true"/>
    </bean>
</beans>
```

3. Create a new file in the same directory called `jasperreports_extension.properties` and populate it with the following.

This does not need to be edited, unless you change the name of fonts.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.s-
pringframework.org/dtd/spring-beans.dtd">
<beans>
    <bean id="[unique name of font family e.g. 'myFontFamily']" class-
s="net.sf.jasperreports.engine.fonts.SimpleFontFamily">
        <property name="name" value="[font name as referenced in jasper reports]"/>
        <property name="normal" value="[file name of normal font]"/>
        <property name="bold" value="[file name of bold font]"/>
        <property name="italic" value="[file name of italic font]"/>
        <property name="boldItalic" value="[file name of bold-italic font]"/>
        <property name="pdfEncoding" value="Identity-H"/>
        <property name="pdfEmbedded" value="true"/>
    </bean>
</beans>
```

4. Use the Java `jar` command to package up the fonts and meta data:

   `jar cvf myfont.jar *`

   This creates a jar file called `myfont.jar` containing all the fonts in the directory as well as the `fonts.xml` and `jasperreports_extension.properties` files.

5. Copy `myfont.jar` into `WEB-INF/lib` directory on the IdentityIQ server. The font should now be available to any JasperReport reports.

6. In order for any reports to use this new font, the report must be edited to reference the font. This is accomplished by modifying the appropriate JRXML object in IdentityIQ. This can be done using the IdentityIQ debug pages or by modifying the `.jrxml` file in the IdentityIQ installation and re-importing the file to update the object in IdentityIQ.

   For example, to change the font used in the title style to `myFont`:

```
<style
    name="title"
    isDefault="false"
    fontName="myFont"
    fontSize="24"
    isBold="true"
    isBlankWhenNull="true"
/>
```

Using Custom Fonts with JasperReports Pie Charts

For IdentityIQ to use the font that supports the language and character set that you need, add the following into the system configuration object specifying the font, font size, and font style that is desired:

```
<entry key="chartTitleFontName" value="MingLiU"/>
<entry key="chartTitleFontSize" value="12"/>
<entry key="chartTitleFontStyle" value="plain"/>
```

> You must add the font size, otherwise the size defaults to 0, which is invisible. The style options include plain, bold, or italic

How to Install and Deploy IdentityIQ

Download and Expand the Installation Files

You must have access to both the *identityiq_installation* and *identityiq_home* directories. where: *identityiq_installation* is the directory in which you download the installation files and *identityiq_home* is the directory in which you expand the `identityiq`.war file.

The installation files are contained in a .zip file available from SailPoint.

> Only download one instance of the IdentityIQ installation files to this application server.

Download the IdentityIQ installation files to a temporary installation directory on your application server.

For example, `C:\identityiq_installation`.

The IdentityIQ installation files and directories are as follows:

- `identityiq.war`
- `database`
- `doc`
- `integration`

1. Expand the `identityiq.war` file to an IdentityIQ staging directory.
   a. Create an `IdentityIQ` staging directory.
      `mkdir identityiq_home`
   b. Access the `IdentityIQ` staging directory.
      `cd identityiq_home`
   c. Expand the `identityiq.war` file to this directory.
      `jar -xvf` *identityiq_installation*`\identityiq.war`
      where *identityiq_installation* is the directory in which you downloaded the installation files.

> On UNIX platforms, run the following command to make the IdentityIQ CLI launch script executable: `chmod +x WEB-INF/bin/iiq`

Configure the Number of Extended and Searchable Attributes Allowed

> You do not need to perform this procedure if the default extended and searchable attributes are sufficient for the needs of your enterprise. If you do not need to configure these attributes, continue to Create the IdentityIQ Database and Tables and use the sample scripts provided.

IdentityIQ is configured by default to enable the following:

- Identity — 10 searchable attributes, 5 indexed
- Account — 5 searchable attributes, 1 indexed
- Certification — 5 searchable attributes, 1 indexed
- Role — 4 extended attributes, 1 indexed
- Application — 4 extended attributes, 1 indexed
- Managed Attribute — 3 extended attributes, 3 indexed
- Target — 1 extended attribute, 1 indexed
- Alert — 1 extended attribute, 1 indexed

If your enterprise requires more than those configured, you must use the following procedure to add as many additional extended and searchable attributes as needed, up to a maximum of twenty (20). You can also use this procedure to set these attributes to be indexed to enhance search speeds. You should take into consideration, however, that while indexing these attributes will increase search speed, it might reduce processing speed for other IdentityIQ functions.

If you make changes to the account attributes you must make the same changes to the certification item attributes. This enables searchable attributes from links to be stored with additional entitlements on certifications to enable searching and the display of account status icons.

> See the comments at the top of the IdentityExtended.hbm.xml file for database-specific considerations on column sizes.

Edit the following files:

- `IdentityExtended.hbm.xml` — identity attributes
- `LinkExtended.hbm.xml` — account attributes
- `CertificationItemExtended.hbm.xml` — certification attributes
- `ApplicationExtended.hbm.xml` — application attributes
- `BundleExtended.hbm.xml` — role attributes
- `ManagedAttributeExtended.hbm.xml` — managed attributes
- `TargetExtended.hbm.xml`— permission targets
- `AlertExtended.hbm.xml` — activity alerts

The files are located in *`identityiq_home`*`\WEB-INF\classes\sailpoint\object\` where *`identityiq_home`* is the directory in which you expanded the `identityiq.war` file.

To edit the files:

1. Open the file with an XML or text editor.
2. Scroll down to the section that appears similar to the following:

```
<property name="extended1" type="string" length="450"
            index="spt_identity_extended1_ci"/>
```

3. Enter as many additional attributes as needed, up to a maximum of twenty.
   Each line `<property name="extended2" type="string" length="450"/>` represents one extended or searchable attribute. As you add additional attribute lines, number them sequentially. For example:

```
<property name="extended2" type="string" length="450"/>
<property name="extended3" type="string" length="450"/>
<property name="extended4" type="string" length="450"/>
```

4. *Optional:* Specify attributes that should be indexed.
   For example in the identity file, add `index="spt_identity_extended`*N*`_ci"` to each attribute line that should be indexed. Where *N* matches the number of the attribute.

   If case insensitivity is required, use `index="spt_identity_extended`*N*`_ci"`.

   For example:

```
<property name="extended1" type="string" length="450"
            index="spt_identity_extended1_ci"/>
<property name="extended2" type="string" length="450"
            index="spt_identity_extended2_ci"/>
<property name="extended3" type="string" length="450"
            index="spt_identity_extended3_ci"/>
```

5. Save the file.

To create new database scripts:

Use the `iiq` script to run the schema command to create the new database creation scripts based on your changes to the `.hbm.xml` files. For example, do the following to run the schema command:

1. Access the proper directory.

   ```
   cd identityiq_home\WEB-INF\bin
   ```

2. Run the command to create the scripts you will use to create the `IdentityIQ` databases.

   ```
   iiqschema
   ```

Continue with Create the IdentityIQ Database and Tables.

Create the IdentityIQ Database and Tables

> Refer to the Advanced Installation Information for specific information about the databases you are using.

The database DDL scripts for the supported database platforms are located in *`identityiq_home`*`\WEB-INF\database` where *`identityiq_home`* is the directory in which you expanded the `identityiq.war` file. Use these scripts to create the IdentityIQ database and tables.

The database directory contains sample database DDL scripts for the supported database platforms. The scripts that correspond to the default extended and searchable attributes contain the product version number in the filename, for example, `create_identityiq_tables-8.2.mysql`. If you changed the extended and searchable attribute configuration by modifying the `.hbm.xml` configuration files and ran the **iiq** `schema` command, then the appropriate sample DDL script is in a file without the product version number in the filename, for example, `create_identityiq_tables.mysql`.

The scripts are designed to create an application database instance, a plugin database instance, the tables and indexes for each database, create a user with a password for each database, and grant the user privileges for accessing the databases.

With the exception of the Oracle script, the scripts provided are sufficient as they are. The Oracle script must be modified to specify the `DATAFILE` location and to un-comment the associated commands that create the database instance. For all database types, it is likely that you will want to change the database user name and password.

The scripts can be modified as required for your environment as long as the table names, column names, and column types are maintained.

Take note of the following information, you will need it when you configure IdentityIQ:

> The user/password in the following list can, but do not have to be, the same.

- Host Name
- Database Type
- Database Name
- User ID/Password for a user that can create tables and modify database schema
- User ID/Password for a user that IdentityIQ will use to access the database. This IdentityIQ user must be able to create, modify, and delete objects in the tables that are created.
- User ID/Password for a user that IdentityIQ will use to access the plugin database. This IdentityIQ user must be able to create, modify, and delete objects in the tables that are created.

A database client is used to execute the DDL scripts. Example commands for MySQL would look as follows:

```
mysql -u user -p
Password: password
mysql> source create_identityiq_tables.mysql;

— Confirmation output from the SQL commands contained in the script is displayed on
the screen —

mysql> show databases; (optional - verification that the database was created)

mysql> quit
```

where *user* and *password* are the credentials for a database user that has permissions to create databases, tables, and users.

Create Site-Specific Encryptions Keys

A default encryption key is compiled into the product for ease of deployment in demonstration environments. Site-specific keys can be generated and stored in a file-based keystore. Use of site-specific keys is strongly recommended to ensure that sensitive data cannot be decrypted outside of this installation of IdentityIQ.

For more details, see Data Encryption.

For more details, see the **System Configuration** documentation.

Configure the IdentityIQ Installation

To configure IdentityIQ to use the database you created:

The `iiq.properties` file is located in *identityiq_home*`\WEB-INF\classes`, where *identityiq_home* is the directory in which you expanded the `identityiq.war` file.

1. Access the `iiq.properties` file and update the following information for the application and plugin databases:
    - Host Name
    - Database Type
    - Database Name
    - User ID
    - Password
2. Run the `iiq encrypt <password>` command from the *identityiq_home*`\WEB-INF\bin` directory to create an encrypted password. The password for the database connection can be clear text or encrypted. It is highly recommended that you use an encrypted password.

3. Import the initial configuration objects into the IdentityIQ database.
4. Launch the IdentityIQ console by running the `iiq console` command from the *identityiq_home*`\WEB-INF\bin` directory. Use the console command **import init.xml** to import the configuration objects as shown:

```
iiq console
> import init.xml
> quit
```

5. Start your application server.

Install or Deploy IdentityIQ

> IdentityIQ may only be defined once on an application server so that it will only be instantiated once.

The IdentityIQ application is packaged in the Java EE standard WAR format in the `identityiq.war` file. Use the tools provided by your application server for the installation or deployment. The staging directory created in the steps above can be copied into the application server installation or used as an installation source depending on the application server deployment process. Alternatively, the contents of the staging directory can be packaged in a war file for deployment.

When deploying IdentityIQ into an application server as a war file instead of a directory containing the expanded war file, the configuration file `WEB-INF\classes\packtag.properties` must be modified to set `resources.check-timestamps=false`. The file contains a sample configuration line that must be un-commented and modified to set the value.

IdentityIQ requires a high level of class loader isolation so that the third party libraries provided with IdentityIQ are used instead of ones provided by the application server. Some of the supported application servers require additional configuration to achieve this isolation. Refer to **Advanced Installation Information** for specific information about your application servers.

> On UNIX platforms, run the following command to make the IdentityIQ CLI launch script executable: chmod +x WEB-INF/bin/`iiq`

Open IdentityIQ

Open IdentityIQ from your Web browser and continue with the configuration and definition needed for your enterprise.

See IdentityIQ Configuration for information on configuring IdentityIQ.

See the **System Configuration** documentation for information on configuring IdentityIQQ.

> If you purchased SailPoint Lifecycle Manager refer to the **Lifecycle Manager** documentation for further instruction on how to activate that product.

1. Launch a Web browser. See Supported Platforms
2. Point the browser to your IdentityIQ installation.

   Example:

   If you performed the default Tomcat installation and are accessing IdentityIQ from the same system on which it was installed, use `http://localhost:8080/identityiq`.

3. Log on to the IdentityIQ using the default user ID and Password.
   The default user ID is `spadmin` and the default password is `admin`.

   You should change the default password as soon as possible. To change the default password for `spadmin`:

4. From the navigation menu bar, click **Identities > Identity Warehouse** to display the Identities page.
5. Click `spadmin` in the list to display the View Identity page.

   > To filter by identity name and quickly find a user, enter the first few letters in the search field and click the search icon.

6. Click **Change Password** to display the password information.
7. Change and confirm the password and save your changes.

Advanced Installation Information

> Other, supplemental, components such as password interceptor plugins, Connector Gateway instances, and connector agents should be updated to match the version of the IdentityIQ server. See the Direct Connectors Administration and Configuration, Integration, and Quick Reference documentation for Gateway Connectors delivered with the product, for additional details.

Use the advanced installation information to customize IdentityIQ for your enterprise. Use the advanced installation instructions to do the following:

Configure Using Oracle

The Oracle JDBC driver implementation is no longer provided in IdentityIQ. You should obtain the JDBC driver that is provided with the database installation.

If you are using the Oracle ojdbc8 driver, you should make a configuration change to your JVM parameters. Disable `nio` by adding `-Doracle.jdbc.javaNetNio=false`. This prevents issues where the db connection is closed in error.

Configure Using SQL Server

> You must choose a case insensitive collation for this database.

The SQL Server JDBC driver implementation is no longer provided in IdentityIQ. You should obtain the JDBC driver that is provided with the database installation.

Configure Using Aurora

> You must choose a case insensitive collation for this database.

When using Aurora, you must modify the create user section of the `create_identityiq_tables.mysql` script by removing the MySQL section and un-commenting the Aurora section. You will then use that script to create your tables.

Configure Using Azure SQL

> You must choose a case insensitive collation for this database.

When using Azure SQL, you must first create a login for the identityiq and identityiqPlugin databases before creating new users. Run the following on the master database to create the new databases and logins:

```
CREATE DATABASE identityiq
GO
CREATE DATABASE identityiqPlugin
GO
CREATE LOGIN identityiq WITH password='<PASSWORD>'
GO
CREATE LOGIN identityiqPlugin WITH password='<PASSWORD>'
GO
```

Then connect to the individual databases to create the database users and tables as follows:

```
CREATE USER [identityiq] FOR LOGIN [identityiq] WITH DEFAULT_SCHEMA=[identityiq]
GO
EXEC sp_addrolemember 'db_owner', 'identityiq';
GO
GRANT EXECUTE TO identityiq
GRANT INSERT TO identityiq
GRANT DELETE TO identityiq
GRANT UPDATE TO identityiq
GRANT SELECT TO identityiq
GO
CREATE SCHEMA identityiq AUTHORIZATION identityiq
GO
ALTER DATABASE identityiq SET ALLOW_SNAPSHOT_ISOLATION ON
GO
ALTER DATABASE identityiq SET READ_COMMITTED_SNAPSHOT ON
GO
```

Once the users and databases are created, proceed with the creation of IdentityIQ tables.

Configure Using MySQL

> You must choose a case insensitive collation for this database.

File-Per-Table Tablespaces

When using MySQL, IdentityIQ can achieve significant performance improvements by using File-Per-Table tablespaces. This enables each table and its indexes to be in its own data file.

> This setting must be configured prior to creating the IdentityIQ tables using the DDL scripts provided as it only affects behavior during table creation

File-Per-Table tablespaces can be enabled by using the innodb_file_per_table MySQL configuration parameter.

ONLY_FULL_GROUP_BY SQL Mode

MySQL 5.7.5 and later enable the ONLY_FULL_GROUP_BY SQL mode by default, and this setting is incompatible with IdentityIQ. To disable this setting, the sql_mode MySQL configuration parameter must be defined to not contain the ONLY_FULL_GROUP_BY SQL mode.

Default configurations do not list the enabled modes, so you can use the following MySQL command to view the current settings

```
show variables where variable_name='sql_mode';
```

MySQL system variable and configuration parameters are configured in the [mysqld] section of my.cnf or other equivalent methods.

For more information, consult the MySQL documentation.

Configure Using DB2

The DB2 JDBC driver implementation is no longer provided in IdentityIQ. To use DB2 as the IdentityIQ repository or for a JDBC application that connects to a DB2 database, the correct implementation jar file should be obtained from the DB2 installation or from the IBM web site.

Deploy Using Tomcat

To deploy a Web application using Tomcat unpack the `.war` file into an application directory in the webapps directory of the Tomcat installation.

1. Access the `webapps` directory.

   ```
   cd Tomcat_home\webapps
   ```

2. Create an identityiq home directory.

   ```
   mkdir identityiq
   ```

3. Access the identityiq home directory.

   ```
   cd identityiq
   ```

4. Expand the `identityiq.war` file to this directory.

   ```
   jar -xvf identityiq_home\identityiq.war
   ```

   where *`identityiq_home`* is the directory in which you downloaded the installation files.

Tomcat System Properties

> SailPoint recommends increasing you cache size to 50MB to avoid insufficient free space warnings

To enable UTF-8 characters in IdentityIQ installations using Tomcat, add the following line to each Connector element in your Tomcat server.xml.

```
URIEncoding="UTF-8″
```

The following is an example entry:

```
<system-properties>
  <property name="org.jboss.as.logging.per-deployment" value="false"/>
</system-properties>
```

Deploy Using WebSphere

Prerequisites:

- The default JVM heap size for a WebSphere installation is not sufficient to deploy the IdentityIQ web application. The heap size should be changed using the steps defined in the WebSphere documentation.
- A complete `.war` file that contains all of the configuration and customization required for the environment.

To deploy IdentityIQ on an IBM WebSphere Application Server:

Use the following procedure to deploy IdentityIQ on an IBM WebSphere Application Server. These steps are used once a complete `.war` file is created that contains all of the configuration and customization required for the environment.

> **The following steps refer to 9.0 interfaces.** WebSphere 9.0 requires an additional jar file to be placed into the shared library created as part of the following deployment steps.

1. Use the contents of the `.war` file provided with the product and any configuration and customizations required to create a `.war` file of the SailPoint IdentityIQ application.
2. Create the following directories in your WebSphere installation directory (`C:\Program Files (x86)\IBM\WebSphere-9.0\AppServer` by default on Windows platforms)

   ```
   identityLib
   identityLib\classes
   ```

3. Copy the files listed below from your *`identityiq_home`* directory (where you expanded the `identityiq`.war file) to the `identityLib` directory using the following structure:

   > Make sure the commons-logging.properties file is located in the classes directory.

   ```
   commons-codec-1.13.jar
   hk2-api-2.6.1.jar
   hk2-locator-2.6.1.jar
   hk2-utils-2.6.1.jar
   httpclient-4.5.13.jar
   httpcore-4.4.13.jar
   javax.faces-2.2.20.jar
   javassist-3.27.0-GA.jar
   jakarta.ws.rs-api-2.1.6.jar
   ```

4. Start the WebSphere server.
5. Run the Integrated Solutions Console and login.

6. Click **Applications**, select **New Application**, and click **New Enterprise Application**.
7. Select the `.war` file you created in the beginning step from the local or remote filesystem and then click **Next**.
8. Select **Fast Path** and click **Next**.
9. Ensure that **Distribute Application** and **Use Binary Configuration** are selected and that **Create MBeans** is not.
10. Click **Next** until the Map context roots for Web Modules screen is displayed.
11. Set the Context Root to your desired value, for example `/identityiq`.
12. Click **Next** and then **Finish**.
13. When the application has completed installation, click **Save** to save the changes to the master configuration.
14. After IdentityIQ is installed, configure it by selecting it from the **Applications > Application Types > WebSphere Enterprise Applications** list.
    The name in the list will match the name of the .war file that you selected to deploy.
15. Click **Class loading and update detection**.
16. Ensure that a value is entered in the **Polling interval for updated files** field and that **Classes loaded with local class loader first (parent last)** and **Class loader for each WAR file in application** are selected.
    The polling interval can be zero (0), but there must be a value in the field.
17. Click **OK** and save your changes.
18. Click **Application Types > WebSphere Enterprise Applications > identityiq.**`war`.
19. Click **JSP and JSF options**.
20. Click **OK**.
21. Click on **Environment > Shared libraries**.
22. Click **New**.
23. Enter `identityLib` as the name.
24. Add the following files to the list as the classpath:

```
${WAS_INSTALL_ROOT}/identityLib/commons-codec-1.13.jar

${WAS_INSTALL_ROOT}/identityLib/hk2-api-2.6.1.jar

${WAS_INSTALL_ROOT}/identityLib/hk2-locator-2.6.1.jar

${WAS_INSTALL_ROOT}/identityLib/hk2-utils-2.6.1.jar

${WAS_INSTALL_ROOT}/identityLib/httpclient-4.5.13.jar

${WAS_INSTALL_ROOT}/identityLib/httpcore-4.4.13.jar

${WAS_INSTALL_ROOT}/identityLib/javax.faces-2.2.20.jar

${WAS_INSTALL_ROOT}/identityLib/javassist-3.27.0-GA.jar

${WAS_INSTALL_ROOT}/identityLib/jakarta.ws.rs-api-2.1.6.jar
```

25. Verify that the **Use an isolated class loader for this shared library** box is checked.
26. Click **OK**.
27. Go back to the `identityiq.war` application page.
28. Click **Shared library references** near the bottom of the configuration panel in the References section.
29. Check the box next to the **identityiq _war** row under the Application column and click **Reference Shared Libraries** above the table.
30. Select **identityLib** from the table of Available libraries and click the right arrow to move it into the Selected table.
31. Click **OK**.

32. Click **OK** again.
33. Click **Save** to apply the changes to the master configuration.
34. Add a new Custom Property:
    a. Go to **Application Servers > [serverName] > Expand Java and Process Management > Process Definition > Java Virtual Machine > Custom Properties**
    b. Click **New**
    c. Name: `com.ibm.ws.cdi.enableCDI`
    Value: `false`
    d. Click **OK**
    e. Save the changes.
35. Re-start the application server.
36. Start the application from **Applications > Application Types > WebSphere Enterprise Applications**.

Deploy Using WebSphere Liberty

When IdentityIQ is deployed in WebSphere Liberty, specific features must be enabled. Using the featureManager and feature tags in the server configuration defined in `server.xml`, enable the jsp-2.3 feature and ensure that no other features are enabled. This will automatically include the required servlet and el features. Features other than these three are known to cause conflicts that will prevent IdentityIQ from working properly.

If you have enabled full text indexing for IdentityIQ, the location of the full text index should be outside of the IdentityIQ installation or application update monitoring should be disabled. See the updateTrigger attribute of the applicationMonitor element in the server configuration documentation for additional details.

Deploy Using JBoss

> Previously supported releases of JBoss Application Server required configuration that is no longer required in the Application Server versions supported in this release. If you use older unsupported versions of JBoss, the additional configuration, defined in previous versions of the *IdentitIQ Installation Guide*, are still required.

JBoss Datasources

When using a container-provided or application server managed datasource for the IdentityIQ database, follow the complete instructions provided in the Datasource Management section of the JBoss Configuration Guide for how to configure JBoss and IdentityIQ to use the datasource. These instructions include:

- Define a JBoss JDBC driver core module for the JDBC driver.
- Configure the datasource using the JBoss GUI or CLI.

Additionally, you must:

- Remove the JDBC driver jar file from the IdentityIQ `WEB-INF/lib` directory.
- Add the JDBC driver core module as a dependency in the IdentityIQ `WEB-INF/jboss-deployment-structure.xml` file.

JBoss System Properties

UTF-8

To enable UTF-8 characters in IdentityIQ installations using JBoss, add the following xml after the `<extensions/>` element to the `standalone.xml` or `domain.xml` configuration file

```
<system-properties>
  <property name="org.apache.catalina.connector.URI_ENCODING"
value="UTF-8"/>
  <property
name="org.apache.catalina.connector.USE_BODY_ENCODING_FOR_QUERY_STRING"
value="true"/>
</system-properties>
```

Maximum Parameter Requirements

Editing an IdentityIQ application with a large number of schema attributes will not function properly unless you increase the maximum number of parameters allowed. Add the following to the `standalone.xml` file to increase the maximum number of parameters.

```
<system-properties>
          <property
name="org.apache.tomcat.util.http.Parameters.MAX_COUNT" value="1000"/>
</system-properties>
```

To avoid seeing the HTTP error code 405 on saving a ACF2 application in the JBOSS environment, add following entry in your server configuration file and restart the JBOSS server.

```
<http-listener name="default" socket-binding="http" max-parameters="5000"/>
```

Logging Using log4j

To enable application logging using the log4j logging facilities, add the following to `standalone.xml` to disable per-deployment logging:

```
<system-properties>
  <property name="org.jboss.as.logging.per-deployment" value="false"/>
</system-properties>
```

JBoss EAP 7.2 and Higher

The JBoss Undertow subsystem default behavior is to proactively authenticate HTTP basic authentication headers against its internal authentication system and not pass credentials through to IdentityIQ. This prevents web service APIs, such as the SCIM interface, from being used with IdentityIQ credentials.

To use SailPoint IdentityIQ web service APIs with JBoss EAP 7.2 and higher, disable proactive authentication by running the following command in the JBoss administrative CLI:

```
/subsystem=undertow/servlet-container=default:write-attribute(name=proactive-authen-
```

```
tication,value=false)
```

> For this setting to take effect, you must restart the application server.

Deploy Using WebLogic

When IdentityIQ is deployed in WebLogic, you must configure the environment before inbound web service requests can be received. Inbound service request are required for IdentityIQ RESTful web services and some of the provisioning providers with whom IdentityIQ integrates.

The Security Configuration MBean flag **enforce-valid-basic-auth-credentials** must be set to **false** to prevent HTTP requests for web services with embedded authentication information from needing to pass WebLogic Server IdentityIQ authentication. You can accomplish this by adding the following to the *security-configuration* element in the `config.xml` file.

```
<enforce-valid-basic-auth-credentials>false</enforce-valid-basic-auth-credentials>
```

Install and Register the IQService for Use with Windows

The IQService is a native Windows service that enables IdentityIQ to participate in a Windows environment and access information only available through Win32 APIs.

After the IQService is installed and running on a Windows machine you can configure tasks in IdentityIQ that use the service such as Windows activity collection, Windows target collection, Active Directory provisioning, and more.

> Version 4.5 of Microsoft .NET is required for installation of the IQService executable with most configurations of IdentityIQ. If your configuration requires a different version it is noted in the connector documentation.

> The IQService version must match the IdentityIQ server version including patch versions. When you upgrade one, you must upgrade the other.

Install the IQService in a location that is running Windows.

To install and register the IQService, do the following:

1. Create a directory in which to download the service. For example, `C:\iqservice`.
2. Copy the `IQService.zip` file from the IdentityIQ installation into the new directory. The `IQService.zip` file is located in *`identityiq_home`*`/WEB-INF/bin/win` where *`identityiq_home`* is the directory in which you expanded the `identityiq.war` file.
3. Expand the `IQService.zip`.
4. Run **IQService.exe -i** to install a Windows service named IQService.
5. Start the service from the Windows Services Applet or from the command line by running **IQService.exe -s**.

   Other command line options with this service are:

   ```
   -? | h — This help output
   -d — run in console mode
   -i — Install a service
   -k — stop the service
   -n — (Optional) name of IQService for installing multiple instances. Default:
   ```

```
IQService-Instancex, where x is an incremental integer value.
-p — (Optional) unique available port number specified at the time of IQService
installation. Default: 5050. Incremental based on the next available port.
-r — remove the service
-s — Start the service
-t — Restart (stop/start) the service
-u: Uninstall the service. Removes the service components and clears the
registry entries
-v — Print version information
```

The IQService will persist configuration settings related to the port, trace file, and trace level in the Windows registry at `HKEY_LOCAL_MACHINE\SOFTWARE\SailPoint\IQService`. The following keys are used:

- `-port` - port on which to listen
- `-tracefile` - path to the trace file
- `-tracelevel` - 0 (off), 3 (verbose)
- `-maxTraceFiles` - maximum number of Trace log files
- `-traceFileSize`: maximum file size of a trace file in bytes

Configure Integration with Third Party Applications

SailPoint provides support for integration with third party vendor applications to automate workflow and enable you to leverage the role management, risk assessment, and policy enforcement features of IdentityIQ.

Integration focuses on the following points:

- Automated remediation — provides automated workflow to remediate inappropriate access and policy violations using immediate, real-time execution of provisioning or de-provisioning requests.
- Role provisioning — enables you to provision user access to data, applications, and systems based on roles defined within IdentityIQ.
- Policy checking for access changes — incorporates compliance and risk management by analyzing new user requests and role changes against defined policy within IdentityIQ.
- Bulk Re-provisioning of modified roles — enables third party applications to receive requests from IdentityIQ to bulk modify user accounts if roles in IdentityIQ have changed.

Integration requires configuration to both IdentityIQ and the applications with which it is integrating. This configuration is highly dependent on the applications and on the environment in which they are running. SailPoint provides some integration examples in the `integration` folder of the installation package. As well as documentation on integration and configuration delivered with the product.

Configure IdentityIQ for Single Sign-on

IdentityIQ provides rule based SSO authentication support. Create the rules and then specify the rule used to drive authentication from the Login Configuration pages. A rule approach enables you to handle any SSO system that uses the HTTP header to store credentials or other identifying information by writing a new rule.

The `SSOAuthenticationRule` rule is provided the `HttpRequest` and a `SailpointContext` as input and returns an identity for use in the application.

Configure IdentityIQ for Single Sign-on with SiteMinder

An example SiteMinder rule that illustrates exactly how SiteMinder attributes can be pulled from the header and how the `sailpoint.api.Correlator` can be used to return a Link/Identity using an identity attribute like a `dn` can be found in `config/examplerules.xml` file delivered with your installation package.

For SSO to work with SiteMinder, the `sm-authdirname` from the SiteMinder header must be mapped to an application that is configured to work with IdentityIQ and aggregation must have been run on that application so that an Identity Cube exists with a `sm-userdn` user value.

After those criteria are met, the rule specified on the Login Configuration page, performs authentication by matching the `sm-userdn` header variable provided by SiteMinder to an Identity Cube in IdentityIQ.

The rule performs the authentication by pulling the `sm-userdn` and `sm-authdirname` from the SiteMinder header when it receives the `HttpRequest` and a `SailPointContext` and matching it to an Identity Cube.

If the rule does not return an identity indicating that authentication failed, IdentityIQ displays the standard login page for manual authentication.

Synchronize IdentityIQ Server Time

Multiple IdentityIQ servers in the same deployment instance work in a cooperative fashion communicating with each other through the IdentityIQ database. It is imperative that the time set on all servers be synchronized for this to work effectively. Inaccurate times can lead to incorrect detection of server outage, can interfere with object locking designed to protect the integrity of system data, can cause distributed sequential operations to be run in incorrect order, or can cause distributed concurrent operations to be run multiple times.

IdentityIQ Upgrade

Do not perform the upgrade until you have completely read the Release Notes. Additional white papers and articles that can provide additional information to assist with the upgrade procedure are available on Compass, the SailPoint online customer and partner community.

> Refer to Supported Platforms to ensure that the requirements have not changed since your previous installation or upgrade. Update as needed to ensure that IdentityIQ continues to operate as expected.

Upgrade Considerations

> Make sure to store copies of the `iiq.dat` and `iiq.cfg` files in a safe place so they can be restored after the upgrade

> SailPoint strongly recommends that you obtain JDBC drivers from your database server vendor that match the database server version in use.

> SailPoint strongly recommends that you use the version of the `iiq.properties` file that came with the version of IdentityIQ to which you are upgrading do avoid complication due to variable changes.

Implementation of SailPoint IdentityIQ often requires substantial configurations, process changes, customizations, defect mitigations and testing. Upgrades and major releases resolve issues, improve features, and deprecate certain functionality. Review the latest *SailPoint IdentityIQ Release Notes* and user documentation, as well as the implementation procedures, to understand the impacts of each component.

Different Apache Commons DBCP and Commons Pool versions are used by the IdentityIQ server instead of the JDBC connector. Refer to important upgrade information in the Database Connection Pooling and Dialects section of the *Performance Management Guide for IdentityIQ* available on Compass, the SailPoint online customer and partner community.

The following are two specific and distinct upgrades occurring within this process:

- Upgrade Procedure - Upgrading the software in production requires a specific procedure.
- Upgrade Configurations - Client implementations of SailPoint IdentityIQ frequently customize features outside the configuration options that leverage IdentityIQ APIs and/or coding. These features within the configuration and coding APIs of the software change and improve over time. While most of these version-to-version changes are backwards compatible, any implementation specific scripts or rules that use the API features need to be ported, tested, and validated ahead of the actual upgrade procedure when the installation moves into production.

How to Upgrade IdentityIQ

> You can only upgrade from the latest released version of IdentityIQ. This can be from any patch level of the latest release including no patches applied.

Use the following information to upgrade SailPoint IdentityIQ to the latest version. After your upgrade is complete refer to the product documentation for information on the latest features and function.

You must stop all processes and shutdown IdentityIQ before performing the upgrade. It is recommended that you schedule the upgrade for a time when the application is not being used heavily.

> If IQService is installed, the IQService version must match the IdentityIQ server version. If you upgrade one you must upgrade the other.

The upgrade process contains the following parts:

- Shutdown IdentityIQ — See, Shutdown IdentityIQ.
- > It is very important that you perform the database and customization backup before performing the upgrade procedure.
- Backup your existing IdentityIQ databases — See Backup Your Existing Version of IdentityIQ.
- Download and expand the installation files — See To download and expand the installation Files:.
- Perform the upgrade to the latest version of IdentityIQ — See Upgrade the IdentityIQ Database,Upgrade the IdentityIQ Configuration andUpgrade the IdentityIQ External Components.
- Reapply any customization — See Reapply Customization to the Upgraded Installation
- Access IdentityIQ — See Access IdentityIQ.
- Perform the post-upgrade procedures — See Post-upgrade Procedures.

Shutdown IdentityIQ

You must stop all running processes and shutdown IdentityIQ before running the upgrade.

> Some of the items displayed on the Task Results page are approvals that are generated by workflows and are not running processes. These items are managed by IdentityIQ, should not be terminated, and will display on that page after the upgrade procedure is complete.

Check the Task Results and Report Results pages to ensure that no task-generated processes are running before stopping your application server.

If there are task or reports pending, terminate them within IdentityIQ before shutting down your application server. Processes terminated abnormally might not end gracefully.

To terminate a task or report, right-click on the name and select **Terminate**.

Backup Your Existing Version of IdentityIQ

> It is very important that you perform the database and customization backup before performing the upgrade procedure.

Backup your IdentityIQ database and any customization you performed on the current installation. Any customization involving additions or changes to files in the IdentityIQ installation directory is overwritten by the upgrade procedure and will need to be reapplied when it is complete.

Customizations that are stored in the IdentityIQ installation directory can include the .hbm.xml extended attribute configuration files in `WEB-INF\classes\sailpoint\object` and web-based content such as XHTML, JavaScript, and images. See How to Upgrade IdentityIQ for more information.

It is recommended that you create a `custom` directory within you IdentityIQ installation directory so that changes are not lost during future upgrades of the product. Copy any customized files into this directory so that you can copy and paste them back into the working directories after the upgrade is complete.

Customization also includes the task definitions, reports, rules, and workflows that are stored in the IdentityIQ database that were included as part of the product and that are overwritten during the upgrade with the latest version. If you have modified any of these, you will need to re-apply your modifications after the upgrade process.

See your database documentation for information on the proper backup procedure for database you are using.

Download and Expand the Installation Files

> You must have access to both the *identityiq_installation* and *identityiq_home* directories. where: *identityiq_installation* is the directory in which you download the installation files and *identityiq_home* is the directory in which you expand the `identityiq`.war file.

To download and expand the installation Files:

1. Delete any existing IdentityIQ installation files before downloading the newest version.
2. Download the IdentityIQ installation files to a temporary installation directory on your application server. For example, `C:\IdentityIQ_installation`.

   The installation files are contained in a .zip file.

   The IdentityIQ installation files are as follows:

   ```
   identityiq.war
   database
   doc
   integration
   ```

   > In Step 3, unlike in a patch release, the new version of IdentityIQ cannot be expanded into the same directory as an old version. This would not remove files that no longer exist in the new version and therefore would create conflicts that will cause unexpected failure.

3. Upgrade the IdentityIQ application on your application server.

   The IdentityIQ upgrade is packaged in the Java EE standard WAR format in the `identityiq.war` file. Use the tools provided by your application server for the upgrade.
   Example:
   This example is using Tomcat to deploy a Web application by unpacking the .war file into an application directory in the webapps directory of the Tomcat installation.
   a. Access the `webapps` directory.
      `cd` *Tomcat_home*`/webapps`

b. Access the IdentityIQ home directory.
   `cd identityiq`
c. Delete the existing files.
d. Expand the `identityiq.war` file to this directory.
   `jar -xvf identityiq_installation_directory\identityiq.war`
   where *identityiq_installation_directory* is the directory in which you downloaded the installation files.

4. Do not add any previously-used e-fixes in this upgraded installation. E-fixes are only compatible with the patch level for which they were delivered.

Reapply Customization to the Upgraded Installation

If you changed the default number of extended and searchable identity, account, or certification item attributes as part of your initial installation, copy your customized `IdentityExtended.hbm.xml`, `LinkExtended.hbm.xml`, and `CertificationItemExtended.hbm.xml` files into the new installation. Important notes about defining extended attributes in IdentityIQ can be found in `IdentityExtended.hbm.xml`. The notes there apply to all of the product's extended attributes.

If you make changes to the account attributes you must make the same changes to the certification item attributes. This enables searchable attributes from links to be stored with additional entitlements on certifications to enable searching and the display of account status icons.

The upgrade process modifies many different configuration objects in the system. If you have saved the XML representation of these objects externally for the purposes of change control or ease of deployment, you must refresh those objects by exporting them from the system after the upgrade process is complete. Any objects from a previous version of IdentityIQ that are imported into the current version might not have the required changes or enhancements applied. This can cause IdentityIQ to operate abnormally. A list of the objects that are affected is provided in the Release Notes.

Upgrade the IdentityIQ Database

Before running the upgrade script, import the database specific upgrade file and modify it as required. Modifications are required if the database name used is different than the default.

Execute the database specific upgrade script named `upgrade_identityiq_tables.database_type`. These files contain the DDL statements that modify the existing database schema to have the new columns/tables/etc… required for the newer version.

You must modify the upgrade script in the same way that the installation script was modified as part of the initial installation. For example, if you are using a different database name or database schema name than the default, you must modify the script to specify the alternate names.

> Verify the database client being used is set up to continue on error to ensure a successful upgrade. If IdentityIQ patches for the previous version were applied, database commands might have already run in the environment. Consult your database client documentation for further information on configuration settings.

If the extended attributes of the Identity, Account, or CertificationItem were modified during the previous installation, you might need to adjust the upgrade DDL for the `spt_identity, spt_link`, or `spt_certification_item` tables.

For example, in MySQL do the following:`mysql> source upgrade_identityiq_tables.mysql`.

If you want to use plugins that require independent persistence, you must create the plugin database. This database is not created as part of the standard database upgrade script. Customize and run the script contained in `WEB-INF/database/plugins/create_identityiq_plugins_db.database_type` using a database client.

If you installed a patch to the previous version of IdentityIQ, you could receive the following errors when running the upgrade DDL. Because the patch contains all of the required changes from the base revision, some of the changes could have been applied when a patch was installed.

- Duplicate column iiqlock
- Duplicate index spt_cert_item_att_name_ci

Upgrade the IdentityIQ Configuration

After the database is upgraded, run the IdentityIQ upgrade CLI. This example assumes that no custom modifications were made to the original installation.

To run the IdentityIQ Upgrade CLI:

1. Access the directory in which you extracted the identityiq.war file.
   `cd identityiq_home\WEB-INF\bin`
2. Run the script and command
   `iiq upgrade`

> This command applies changes to the configuration and managed data in the IdentityIQ database and only needs to be run one time for each installation of IdentityIQ regardless of the number of application servers that are in the installation.

Upgrade the IdentityIQ External Components

If external components, such as the IQService and Connector Gateway,were deployed, those components should be upgraded at the same time as the IdentityIQ server. Additionally, configuration in third party managed systems required for IdentityIQ integration should be upgraded.

Upgrade the IQService, if it is deployed:

1. Backup your existing IQService installation.
2. Ensure that the service is stopped, either from the Services Applet or from the command line by running: `IQService.exe -k`
3. Uninstall IQService using command: `IQService -u`
4. Extract latest IQService in installation directory
5. Install IQService using command: `IQService -i`
6. Start the service: `IQService.exe -s`

> If you have executed the IQService Public Key Exchange Task for existing IQService, it is recommended that you follow the instructions to install and register a new IQService.

Upgrade the Connector Gateway, if it is deployed:

1. Stop the connector gateway.
2. Save a copy of the `init.xml` configuration file located in the Connector Gateway installation directory
3. Remove the existing contents of the installation directory.
4. Extract the `ConnectorGateway.zip` file into the installation directory.
5. Move the saved `init.xml` configuration file into the installation directory.
6. Start the Connector Gateway service.

Access IdentityIQ

To access IdentityIQ from your Web browser and continue working with IdentityIQ:

1. Start the application server (or the IdentityIQ application if supported by the application server).
2. Launch a Web browser. See Supported Platforms.
3. Point the browser to your IdentityIQ installation.
   Example:
   If you performed the default Tomcat installation and are accessing IdentityIQ from the same machine on which it was installed, use `http://localhost:8080/identityiq`.
4. Log on to the IdentityIQ using the default user ID and Password.
   The default user ID is `spadmin` and the default password is `admin`.

Post-upgrade Procedures

After you complete the IdentityIQ upgrade and the product is up and running, perform the following procedures to ensure that all extraneous information is removed from your database tables and that all of the new features are configured to run correctly.

Clean up IdentityIQ Tables

Clean up the tables, columns, and indexes that are no longer used by IdentityIQ but that were required during the upgrade procedure with the `post_upgrade_identityiq_tables.`*database_type* script.

This can be done at any time, even while the application is running. The scripts for each database type are located in the *identityiq_home*`\WEB-INF\database` directory where *identityiq_home* is the directory where you extracted the IdentityIQ files, not the directory where you downloaded the files.

You must modify the post-upgrade script in the same way that the installation script modified as part of the initial installation. For example, if you are using a different database name or database schema name than the default, you must modify the script to specify the alternate names.

For example, in MySQL do the following:

mysql>`source post_upgrade_identityiq_tables.mysql`

Upgrade Data Export Tables

1. Modify the script matching the database on which your data export tables are stored. If necessary, alter the database name in the script. The scripts are named *upgrade_data_export_tables*.* and are stored in `WEB-INF/database/dataExport` folder of your IdentityIQ installation directory.
2. Using a database client, run the modified application script to upgrade the data export DDL.