The following is a deprecated sample from a 130 page installation guide written for a complex system that is typically installed by Professional Services.

1. Overview

This guide describes how to install XXXX for the following stacks:

Platform	Application Server	Database Server
Red Hat Enterprise Linux	JBoss	Oracle
Red Hat Enterprise Linux	WebSphere	Postgres

You can install XXXX in environments with different platform topologies, so information will differ between installations.

The XXXX installation can have multiple application server instances, databases, and web servers per site. These components are separated on dedicated servers. You can install multiple components on a shared server; however, this document will not describe this information.

1.1. XXXX installation components

Before you begin an installation, it is helpful to understand the components with which you will be working.

- **XXXXinstall.dat**: The application server configuration file.
- **auto_install directory**: The directory from where you will run the installation. It contains the files and folders required to perform the installation.
- **base package**: The mandatory applications that you must install as part of the XXXX installation. This includes the following:
  - **XXXX-XXXX**: Provides the administrative features for the <administrative user interface>.
  - **XXXX**: The <administrative user interface> application.
  - **XXXX** or **XXXX**: Provides authorization and authentication to applications and resources.
  - **XXXX**: Provides common services such as events and auditing.
  - **XXXX**: Moves data to and from XXX environments and performs updates to the runtime XXXXX datastore.
- **config.properties**: The general installation properties file that contains information like the type of database server, type of application server type, and number of nodes needed (if a clustered installation).
- **config.properties.overrides**: The properties file for the XXXX installation
- **Installation worksheet**: The spreadsheet used for manual installations prior to XXXX 1.0. Starting with XXX 1.0, <Company X> recommends that you use the `config.props-overrides.properties` file to specify the installation information. However, the installer still generates this spreadsheet and depends on its existence. This worksheet is deprecated in XXX 2.0 and will no longer be supported in XXX 2.1.
- **install.sh**: The script that you run with arguments to perform the installation or upgrade.
- **XXXXX datastore**: Used to retrieve data from the database and transform it for use at runtime so the data is optimized and efficient.
- **Optional applications**: An application that is not needed to support the base functionality but is available for use by anyone using XXX. An example is the XXXXXX which provides the <administrative user interface> for UI integration.

- **Plugins**: Add-on components that provide business-specific functionality that is built on the base package such as <abc> or <def>. Other solutions that use the <this product> might use plugins like <pluginA> or <pluginB>.
  - **<Plugin1>**: Use to manage industry-standard OpenAPIs to support financial services in <CompanyX> products and solutions. See the *Quick Start Guide* for more information.
  - **<Plugin2>**: Use to manage the APIs that were enabled during the installation, the applications that were registered by API Providers and when the consumer onboards.
  - **<Plugin3>**: Use to access the XXX search and reporting functionality for the <name of feature>.
  - **<Plugin4>**: Use to enable dashboards that have <application name> capabilities in the <administrative user interface>.

NOTE	This is applicable for traditional, non-containerized installations only.

  - **<Plugin5>**: Use to import and export APIs.
  - **<Plugin6>**: The web application that supports <feature>.
  - **<Plugin7>**: The web application that handles <feature> activities at runtime.
  - **<Plugin8>**: The XXX web application that integrates REST-based API information for entitlements
  - **<Plugin9>**: The XXX installs and configures the XXXX server. It is available in `{installdir}/standalone/xxx-server-app/`. See the *XXX Servcer Configuration Guide* for additional configuration details.
  - **XXX**: Gives end-to-end visibility of <items> and provides a single repository for all <items>.
- **Standalone applications**: Applications that run as standalone java processes rather than as web applications such as the <examples>.
- **XXXX Installer**: The install.sh script that is run from the auto_install directory.
- **XXX-Agent**: An agent that is deployed on each of the application nodes to handle the processing of the installation components.
- **XXX-CLI-Client**: An underlying installer CLI client tool that executes the installer commands. The install.sh script is a wrapper to the client that allows it to run in a non-interactive mode.
- **Utilities**
  - **XXXX XXXX**: Encrypts sensitive data like passwords and supports both AES-256 and Triple DES.

1.2. Terminology

- **Autoexposure** - Autoexposure is the process when you define entities in Configuration Builder that are automatically pushed to the Control Center as user interfaces so clients can view the data to perform CRUD operations.
- **Dual Control** - You can also enable Dual Control for pages. Dual Control is a workflow feature where actions submitted by users must be reviewed and approved by another user before the submitted action is completed.

- **Control Center Dashboard** - Use the Dashboard to show live data so you can monitor the performance and health of the system resources as standalone processes or aggregated by nodes.
- **Web applications** - The Web Application Framework is used to build secure web applications that expose user interfaces and APIs. You can integrate these web applications into the Control Center to provide a consistent user experience for your users.

1.3. Port Management overview

You must open multiple ports for APSF installations to complete the installation or upgrade. All nodes in the topology must be able to communicate with each other.

When the Installer validates the input data, it creates a file in `/-CLI-Client` that summarizes the ports assigned on each host. The default name is .

The following describes the use of ports for installations:

Port type	Description
Application server ports	If the nodes are on the same subnet, you do not have to expose the ports if you have a Web Server in front of the Application Servers. See Configure Application Server installation input ( ).
JBoss's Undertow module	Open the firewall for the HTTPs port configured for the Undertow system. See the *JBoss Application Server Installation Guide* for details.
Database port	Open the firewall for the database port if the database server is not in the same subnet as the application hosts so that applications can access the database. With the installer, you do not have to deploy a UPF-Installer-Agent on the database machine.
installer ports	If you plan to run the UPF installation from a machine that is in the same subnet as the application nodes, you do not have to explicitly open the firewall for additional ports. The -CLI-Client will communicate with the -Agents to install the applications. A range of ports to be used for the installation is typically provided as part of the setup performed in the Configure the installation properties (config.properties) section.

Port type	Description
Application ports	Each deployed application uses a port for BSI and a port for JMX communication for interapplication communication. If all your application nodes are in the same subnet, you do not have to open these ports for the firewall. The range is typically provided as part of the setup performed in the Configure the installation properties (config.properties) section. BSI ports only use this range. The JMX port base offset is [redacted] by default. You can override the offset when setting up the Artifact configuration [redacted]

1.4. Use TLS to secure database connections

Before you configure TLS to secure the database connection, your DBA must configure the database server, generate certificates, keystores, and truststores for the client side to which the [redacted] installation configuration will point.

1.4.1. Postgres

NOTE DBAs must see the base package [redacted] for instructions about how to enable SSL/TLS for the PostgreSQL database.

1. Go to the [redacted] section in the config-props-overrides.properties file in the `{installdir}/config/` directory.

   NOTE The {installdir} will be determined by the local mount points on the application and/or database servers where the installation is performed.

2. Uncommment the `[` [redacted] `]` properties for Postgres for all databases [redacted]) and update the values as needed.

```
ssl=true;sslmode=require;sslcert=/home/<username>/.postgresql/<yourhostname>_client.crt;sslkey=/home/<username>/.postgresql/<yourhostname>_client.pk8;sslrootcert=/home/<username>/.postgresql/<yourhostname>_root.crt;sslpassword=<your ssl pwd>
```

   NOTE Ensure <yourhostname>_client.crt, <yourhostname>_client.pk8, and <yourhostname>_root.crt are available in the path. Get the sslpassword from your DBA.

3. Save the file.

1.4.2. Oracle

1.4.2.1. Database server configuration

See [redacted] *Securing database connections Using SSL Guide for Oracle*.

See Oracle's official documentation:

https://oracle-base.com/articles/misc/configure-tcpip-with-ssl-and-tls-for-database-connections

http://www.oracle.com/technetwork/topics/wp-oracle-jdbc-thin-ssl-130128.pdf

1.4.2.2. Database server

See the official documentation: https://support.microsoft.com/en-us/help/316898/how-to-enable-ssl-encryption-for-an-instance-of-sql-server-by-using-mi

1.4.2.3. Basic data source connection

1. Open the [redacted] and go to the [redacted] properties.
2. Go to the following properties and uncomment the lines:

```
#[redacted]=jdbc:oracle:thin:@mydbhost.mydomain.com:[redacted]:[redacted]

#[redacted]]=jdbc:oracle:thin:@mydbhost.mydomain.com:1521:APSFDB

[redacted]=jdbc:oracle:thin:@mydbhost.mydomain.com:[redacted]
```

3. Replace the values using tcps protocol and a URL like this:

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=<yourservername>)(PORT=<portnumber>))(CONNECT_DATA=(SERVICE_NAME=<yourservicename>)))
```

4. Define additional driver properties in [redacted]] properties for Oracle for all databases ([redacted]). Depending on the type of SSL/TLS encryption that you need, set the following properties:
   - For anonymous authentication, specify the anonymous algorithm to use for encryption:

     ```
     jdbc.ssl.https.cipherSuites=SSL_DH_anon_WITH_3DES_EDE_CBC_SHA
     ```

   - For server authentication, specify truststore information:

```
javax.net.ssl.trustStore=D:/truststore/truststore.jks;javax.net.ssl.trustSto
reType=JKS;javax.net.ssl.trustStorePassword=<your password>
```

- For mutual authentication, specify both keystore and truststore information:

```
javax.net.ssl.trustStore=D:/truststore/truststore.jks;javax.net.ssl.trustSto
reType=JKS;javax.net.ssl.trustStorePassword=welcome123;javax.net.ssl.keyStor
e=D:/keystore/keystore.jks;javax.net.ssl.keyStoreType=JKS;javax.net.ssl.keyS
torePassword=welcome456
```

NOTE You can also define the same properties as JVM arguments (see the JNDI data source connection).

5. Save the file.

1.4.2.4. JNDI data source connection

Set the following properties in the Datasources sheet properties:

1. Uncomment the following lines to specify the URL defined in the DSURL. Use the same URL for the basic data source.

```
#[DSSheet].[APSF_DB].[apsfmain].[JNDI].[DSURL]=jdbc:oracle:thin:@mydbhost.mydom
ain.com:1521:APSFDB
#[DSSheet].[UTR_DB].[apsfUTR].[JNDI].[DSURL]=jdbc:oracle:thin:@mydbhost.mydomai
n.com:1521:APSFDB
```

2. Define additional JVM arguments in the [redacted] section. See the details in the [redacted] for the following properties:

   [redacted].jvmArgumentList]

   [redacted].jvmargs.value]

   Repeat the second line and replace [redacted] above with each of the following:

   - [redacted]
   - [redacted]
   - [redacted]
   - [redacted]
   - [redacted]

NOTE

1. You can include the database server certificates in the keystore/truststore of the application server, and not define additional JVM arguments for the keystore and truststore.
2. For improved security, you can use the Password Encryptor Tool and the [redacted], from [redacted] properties section, to encrypt any potential passwords in the above start script tokens JVM arguments, such as keystore and truststore passwords.
3. The [redacted] property overrides the data from other fields (hostname, port, database schema).
4. For WebSphere, see the *WebSphere Installation Guide*, section "Security Considerations" to perform manual configuration as a post-installation step.

1.4.3. Microsoft SQL Server

1.4.3.1. Database server

See Microsoft's official documentation: https://support.microsoft.com/en-us/help/316898/how-to-enable-ssl-encryption-for-an-instance-of-sql-server-byusing-mi

1.4.3.2. APSF basic datasource connection

In the [redacted] section, go to the JDBC URL section that has the [redacted] properties for the APSF_DB, TOKEN_DB, and UTR_DB.

- For Anonymous SSL append `encrypt=true;trustServerCertificate=true` to the URL.
- For Server Authentication SSL, append `encrypt=true;trustServerCertificate=false;trustStore=<path-to-the-jks-file-that-stores-the-certificate>;trustStorePassword=<trust-store-password-in-plain>` to the URL.

1. For SQL Server, define the truststore information for the server authentication as part of the URL, so you do not have to add additional JVM arguments, like you do for Oracle or DB2.
2. For improved security, you can use the Password Encryptor Tool and the [redacted], from the same group of properties mentioned above, to encrypt any potential passwords in the [redacted], such as keystore and truststore passwords.

1.4.3.3. JNDI data source connection

In the Datasources Sheet Properties section, configure the [redacted] using the same information that was used in the [redacted] section. The URL must contain everything including the truststore information.

NOTE

Like the basic datasource [redacted] property, keystore and truststore passwords in the DSURL property can be encrypted using the Password Encryptor Tool and the cipher password under the `CIPHERPASSWORD` property.

1.5. Prerequisites

- See the *Hardware & Software Requirements Guide* to set up the required hardware and software.
- See the *Pre-installation Guide* [redacted] to set up the environment.

1.6. Upgrades to [redacted]

If you are migrating from [redacted] and want to continue to use DES and using the default cipher password must also follow the reencrypt steps because the default cipher password has changed, and you must use the new default cipher to regenerate all encrypted passwords.

1.7. Updates to the installation procedure

This section describes changes to the installation procedure that are new only in this release. If you are familiar with an installation procedure from an earlier release earlier, see the past installation guides for a complete understanding of the installation procedures.

<DEPRECATED>

2. Clean install of single-site [redacted]

This chapter describes how to setup the database, obtain [redacted] artifacts and unpack them, enter the installation configuration input, and perform a clean installation of [redacted] in a single-site.

2.1. Clean, single-site [redacted] checklist

The high-level steps involved in the installation process for a single-site are:

- ☐ Get database information
- ☐ Configure passwordless SSH
- ☐ Confirm your system has the proper utilities like java and unzip.
- ☐ Obtain the [redacted] artifacts and installer required for the installation.
  - ☐ Configure the properties for an automated installation.
  - ☐ (Optional) Validate the configuration for all hosts.
  - ☐ Configure the installation configuration ([redacted]).
  - ☐ Configure Application Server ([redacted]).
  - ☐ Configure the overrides for the [redacted] installation default values of the ([redacted]).
- ☐ Trigger the silent installation process. Use the Installer to:
  - ☐ Setup the environment and [redacted] load
  - ☐ Validate the configuration and prepare the agent
  - ☐ Deploy the automated agent
  - ☐ Install the Application Server (single-node)
  - ☐ Perform the [redacted] Installation workflow
  - ☐ Retry failed installations
  - ☐ If necessary, update the JMS connection for standalone artifacts
  - ☐ Run post-installation steps

2.2. Get database information

For new installations, your DBA must create the databases, tablespaces, user, schemas and grant the necessary roles.

NOTE	DBAs must see the base package [redacted] for instructions.

Make sure you know the following:

- Database (DB) hostname
- Port

- Database home directory
- Tablespace root directory
- Database name
- Database user and password
- Database schema name.

You must have this information to configure the installer to run [redacted] with that database user and update the provided databases.

2.3. Configure passwordless SSH

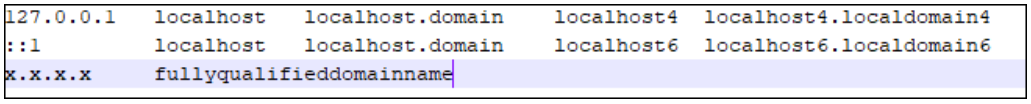
<DEPRECATED>

2.3.1. Steps to configure passwordless SSH from a Source to a Target system

1. Log on to the Source System.
2. Update the hosts file.
   a. Use the following commands to edit `/etc/hosts` with root access.

```
su root
gedit /etc/hosts
```

3. Add a new entry <IP> <HOSTNAME> <FQDN>. For example:

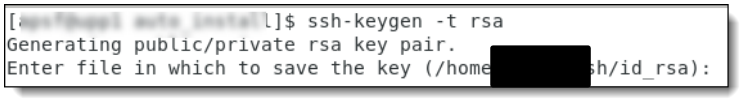


4. As a non-root user, execute `ssh-keygen -t rsa`. Press Enter at all prompts.

NOTE	If you have already configured SSH, you might see a message such as ".ssh/id_rsa already exists. Overwrite (y/n)?". Type `n` if you do not want to override the existing configuration. Otherwise, press Enter.

For example:

```
ssh-keygen -t rsa
```



5. Run the following command to change the directory to `$HOME/.ssh`.

```
cd $HOME/.ssh
```

6. Run the following command to add self to the `authorized_keys` file: `cat id_rsa.pub >>authorized_keys`
7. On the Target system, execute the following steps:
   a. Log on to the Target system.

      If the `$HOME/.ssh` directory exists, skip to the next step.

      If the `$HOME.ssh` directory does not exist, use the following commands to create it and set the permissions:

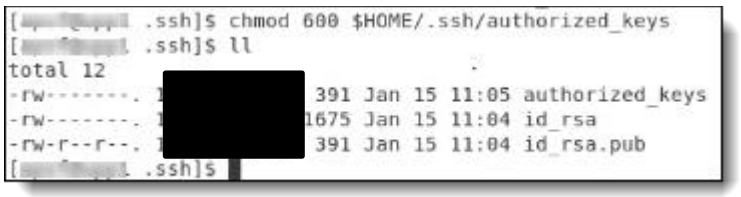
      ```
      mkdir $HOME/.ssh
      chmod go-w $HOME $HOME/.ssh
      ```

   b. Create the `authorized_keys` file if one does not exist and set its permissions:

      ```
      touch $HOME/.ssh/authorized_keys
      chmod 600 $HOME/.ssh/authorized_keys
      ```

   c. Copy the contents the `$HOME/.ssh/id_rsa.pub` file from the Source system machine and append it to the `authorized_keys` file on the Target system.
   d. Type the following command to check the permissions, which should be rw for read-write.

      ```
      ls -l
      ```



   e. Execute an SSH command, e.g. `ssh <target-system> who` from the source system to test

passwordless SSH. Verify the command will run without requiring a password.

```
ssh [target-system] who
```

The first time you execute the `ssh` command against a hostname, you will see the following message. Type `yes` to add the key to your `known_hosts` file. You will not see the message again.

f. If you have a domain attached to your host name, run the following command with the fully qualified domain name. For example:

```
ssh target-system-fullyQualifiedDomainName who
```

g. Use the following command to ensure that the ssh command also works with localhost on each of the app server nodes because the installer might communicate with the localhost or the hostname:

```
ssh localhost who
```

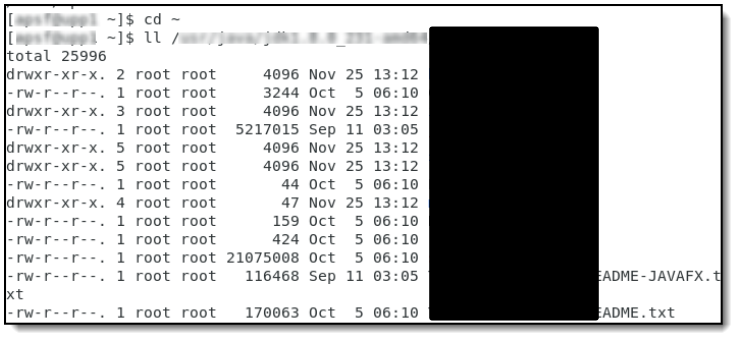
When prompted, type `yes` to permanently add the localhost to your known_hosts file.

2.4. Check required utilities

Oracle JDK (which must have the following utilities accessible through the system $PATH variable: java, javac, jar, keytool) and other Linux utilities like: unzip, sed, ssh, scp, tee, grep, awk, and sleep are required on each environment host where the [redacted] Installer Agents or the [redacted] CLI-Client will run.

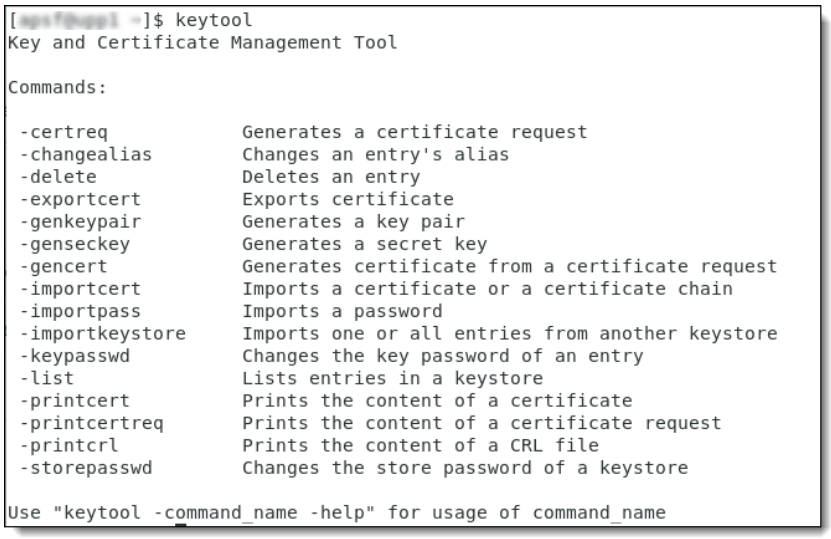
To verify Java, use the following commands to go to the home directory and verify the JDK in the latest location for java (typically /usr/java/jdkx.x.x):

```
cd~
ll /<java_location>/
```



To verify the Linux utilities, confirm that they have manual or help pages. For example, run the following command:

```
keytool
```



NOTE The [redacted] install automation uses bash for scripting. If you change the $PATH variable to include the JDK or other utilities, update the .bashrc file on the environment hosts. To do this:

a. Change to your home directory, `cd $HOME`
b. Edit bashrc: `vi /.bashrc`
c. Add the necessary lines such as `export PATH=/usr/java/<JDK directory>/bin:$PATH`

The [redacted] installer automation scripts use passwordless SSH to run commands remotely and updating the .bashrc file is the best way to ensure these utilities are available in those SSH connections. The .bashrc file should not be printing or echoing any data (such as the pwd).

IMPORTANT [redacted]

2.5. Obtain and prepare [redacted] artifacts

To simplify configuration, [redacted] recommends that you orchestrate the installation from the application server master node. [redacted] has validated this installation approach and it uses fewer configuration changes. The examples in this guide follow this use case.

NOTE If you do not follow the recommendation, [redacted]

Follow these steps on the application server master node.

1. Create a directory with a name of your choosing on the host. This directory will store the installer and installation files. For the rest of this document, this will be referred to as {installdir}.
2. Download the [redacted] artifacts for an [redacted] installation and copy them to {installdir}.
3. Unzip [redacted]`-x.x.x.x.zip`. After you unzip the file, delete it. The directory structure is as follows:

```
[{installdir}]
    |_[auto_install]
        |_[config]       <-Configuration input directory
        |_[patches]      <-[redacted])
        |_[plugins]      <-[redacted])
        |_[thirdparty]   <-JDBC drivers
        |_Readme.txt
        |_install.sh
```