Installing ThingWorx 8

V1.7

Document Revision History

Revision Date	Version	Description of Change
February 2020	1.7	Clarified setting the JVM heap values.
March 2020	1.6	Clarified Tomcat Ghostcat mitigation step.
December 2019	1.5	Added workaround for Ubuntu if Tomcat doesn't automatically start after reboot.
November 2019	1.4	Updated step for removing contents in Tomcat webapp folder.
September 2019	1.3	Updated for 8.5.0.
March 2019	1.2	Fixed broken links, clarified minimum password length, and fixed errors in PostgreSQL tables.
January 2019	1.1	Updated for 8.4.0.
May 2018	1.0	Initial version for ThingWorx 8.3

Contents

1

ThingWorx Installation Overview

Note

These installation steps were tested on ThingWorx 8.5.0 and Apache Tomcat 8.5.x and file names used in the process reflect this, but other versions may be supported. Refer to ThingWorx System Requirements for additional information. The general steps can be used for any version of ThingWorx 8.

Installation steps are available in the Help Center. PDF versions for earlier versions can be located using the Reference Documents page of the PTC Support Portal.

Upgrading

If you are upgrading to a newer version, refer to the Upgrading ThingWorx guide.

Installation Prerequisites

You must have Apache Tomcat and Oracle Java installed. PostgreSQL, InfluxDB, or MSSQL Server may be required if you are not using H2 for your database. Reference the ThingWorx Deployment Architecture Guide for more information about database and deployment options.

Supported Operating Systems

ThingWorx is currently supported on

- Windows on page 8
- Ubuntu on page 39

- RHEL on page 72

Database Options

There are several database options to consider before installing ThingWorx.

- H2 is an embedded database option
- PostgreSQL, MSSQL, Azure SQL, and InfluxDB are external databases that require additional configuration steps

For more information on database options, see the ThingWorx Deployment Architecture Guide, ThingWorx Sizing Guide, and ThingWorx Model and Data Best Practices.

Note

If you are not using PostgreSQL or H2 for your database, refer to the following for additional installation and configuration information:

- Microsoft SQL Server: Getting Started with MS SQL Server and ThingWorx Guide
- InfluxDB: (available in 8.4+): Using InfluxDB as the Persistence Provider
- AzureSQL: (available in 8.4+): Using AzureSQL as the Persistence Provider

For additional information on database options, see the Persistence Providers.

System Requirements

For detailed software and hardware requirements, refer to ThingWorx System Requirements.

This document provides the following server hardware and configuration requirements for running ThingWorx in a production environment:

- Core operating system software requirements
- Prerequisite software required by ThingWorx
- Minimum sizing requirements (for production use)

PostgreSQL High Availability (HA) Option

You can use PostgreSQL with an optional High Availability layer at the database level and/or at the ThingWorx level. Additional steps for HA are required and are located in the ThingWorx High Availability Guide.

2

Windows Installation

- H2/Azure SQL on page 9
- PostgreSQL on page

Note

See ThingWorx Installation Overview on page 6 for other options.

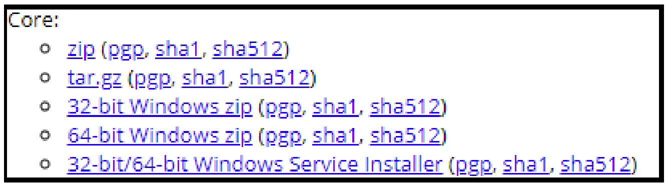
H2/Azure SQL

Install Java and Apache Tomcat (Windows)

1. If you are using AzureSQL for your database, go to Using Azure SQL Server as the Persistence Provider. Perform the steps in that section to set up the database, and you will be referred back to this section.
2. If you are using MSSQL for your database, go to Using MSSQL as the Persistence Provider. Perform the steps in that section to set up the database, and you will be referred back to this section.
3. Refer to the ThingWorx System Requirements for Java JDK version requirements.
4. Download and install the required version of the Java JDK from the Oracle website.
5. Ensure the Java environment variable is configured properly :
   -
     a. Locate your Java installation directory and copy the path. The default path is `C:\Program Files\Java\jdk_<version number>`.
     b. From the Windows start menu, navigate to **Advanced System Properties**. Your path to these properties will vary based on your version of Windows. For example, for Windows 10, search for **Environment Variables** then select **Edit the system environment variables**.
     c. Click **Environment Variables**.
     d. In the **System variables** section, click **New**.
     e. In the **Variable name** field, enter `JAVA_HOME`.
     f. In the **Variable value** field, enter the path to your Java installation as defined in step a.
     g. Click **OK**.
6. Refer to the ThingWorx System Requirements for Apache Tomcat version requirements.
7. Visit the Tomcat website to download the **32-bit/64-bit Windows Service Installer (pgp, sha1, sha512)**.

Note

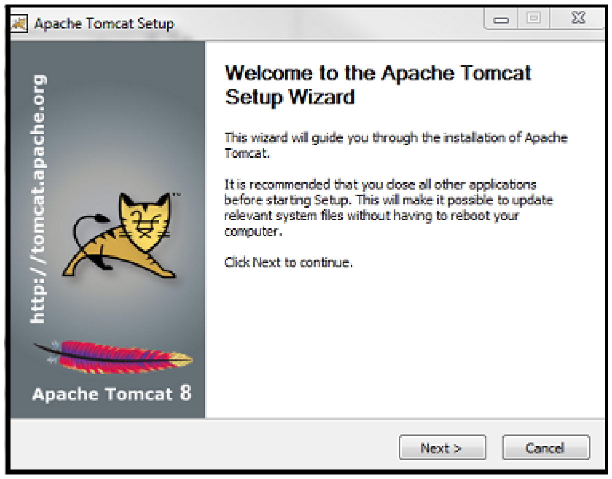
Best practice includes verifying the integrity of the Tomcat file by using the signatures or checksums for each release. Refer to Apache's documentation for more information.



Core:

- zip (pgp, sha1, sha512)
- tar.gz (pgp, sha1, sha512)
- 32-bit Windows zip (pgp, sha1, sha512)
- 64-bit Windows zip (pgp, sha1, sha512)
- 32-bit/64-bit Windows Service Installer (pgp, sha1, sha512)

8. The Apache Tomcat Setup Wizard launches. Click **Next**.



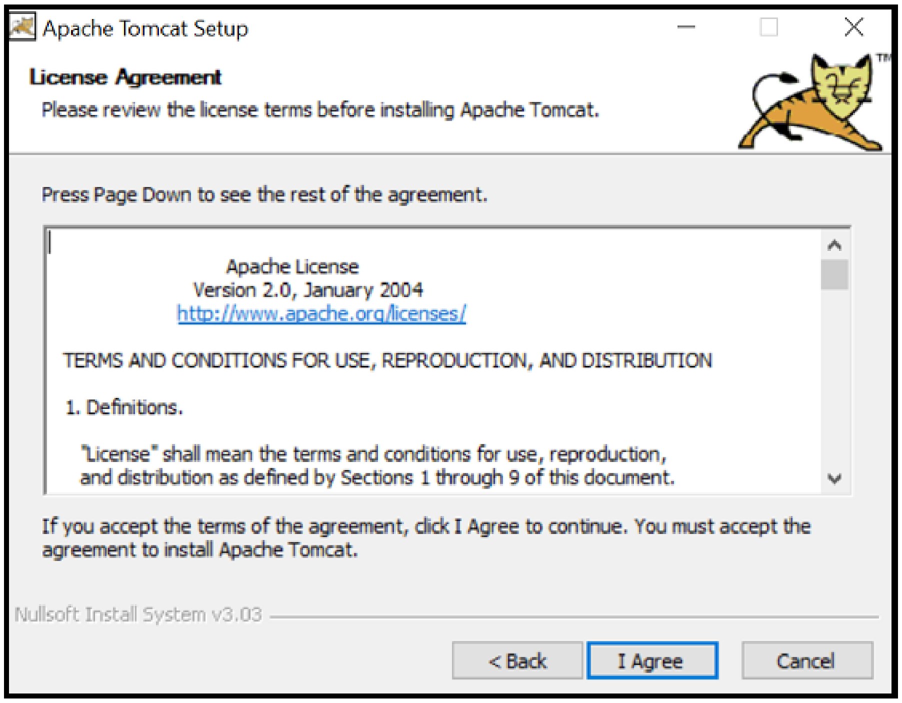
Welcome to the Apache Tomcat Setup Wizard

This wizard will guide you through the installation of Apache Tomcat.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

9. Click **I Agree**.



License Agreement

Please review the license terms before installing Apache Tomcat.

Press Page Down to see the rest of the agreement.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

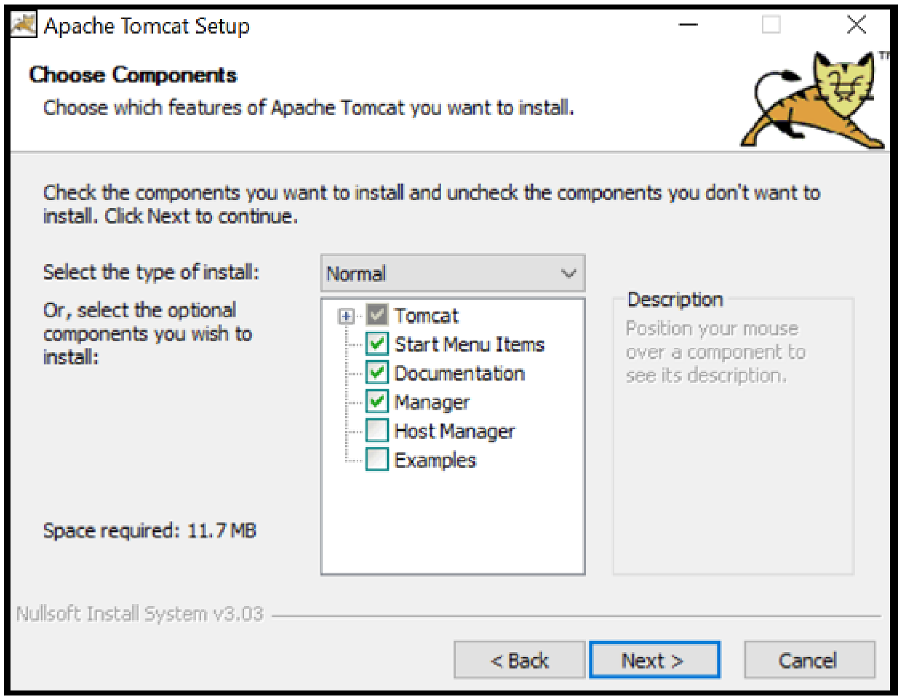
1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Apache Tomcat.

Nullsoft Install System v3.03

10. In the **Choose Components** section, use the default settings. Click **Next**.



Choose Components

Choose which features of Apache Tomcat you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select the type of install: Normal

Or, select the optional components you wish to install:

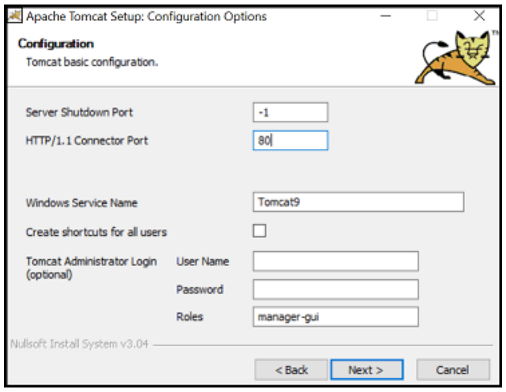
- [✓] Tomcat
- [✓] Start Menu Items
- [✓] Documentation
- [✓] Manager
- [ ] Host Manager
- [ ] Examples

Description

Position your mouse over a component to see its description.

Space required: 11.7 MB

11. In the **HTTP/1.1 Connector Port** field, type `80` (or other available port).
12. In the **Tomcat Administrator Login** fields, you must enter a `Tomcat user name` and a unique, secure password for Tomcat administration. In ThingWorx it is required, not optional.
13. Click **Next**.



Apache Tomcat Setup: Configuration Options

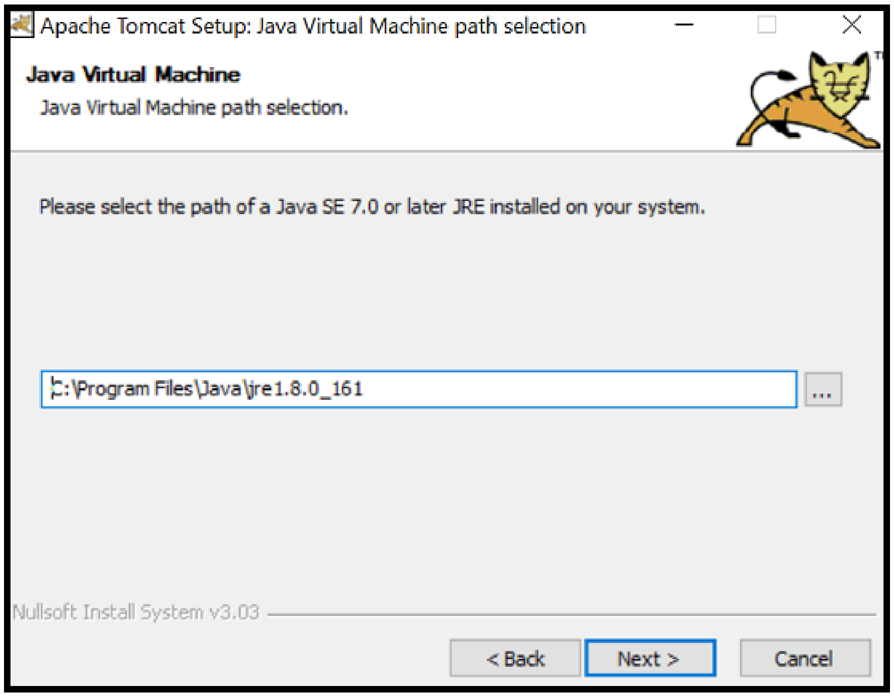
Configuration

Tomcat basic configuration.

Server Shutdown Port		-1
HTTP/1.1 Connector Port		80
Windows Service Name		Tomcat9
Create shortcuts for all users		☐
Tomcat Administrator Login (optional)	User Name	
	Password	
	Roles	manager-gui

Nullsoft Install System v3.04
< Back
Next >
Cancel

14. Click **Next**.


Apache Tomcat Setup: Java Virtual Machine path selection

Java Virtual Machine

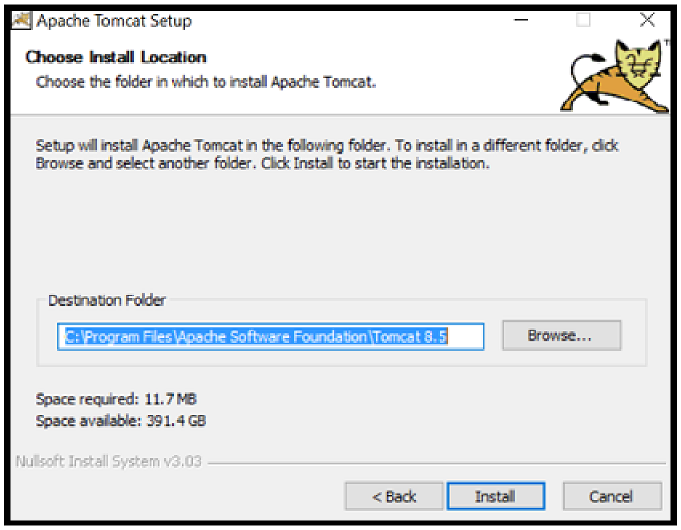
Java Virtual Machine path selection.

Please select the path of a Java SE 7.0 or later JRE installed on your system.

C:\Program Files\Java\jre1.8.0_161

Nullsoft Install System v3.03

15. Click **Install**.


Apache Tomcat Setup

Choose Install Location

Choose the folder in which to install Apache Tomcat.

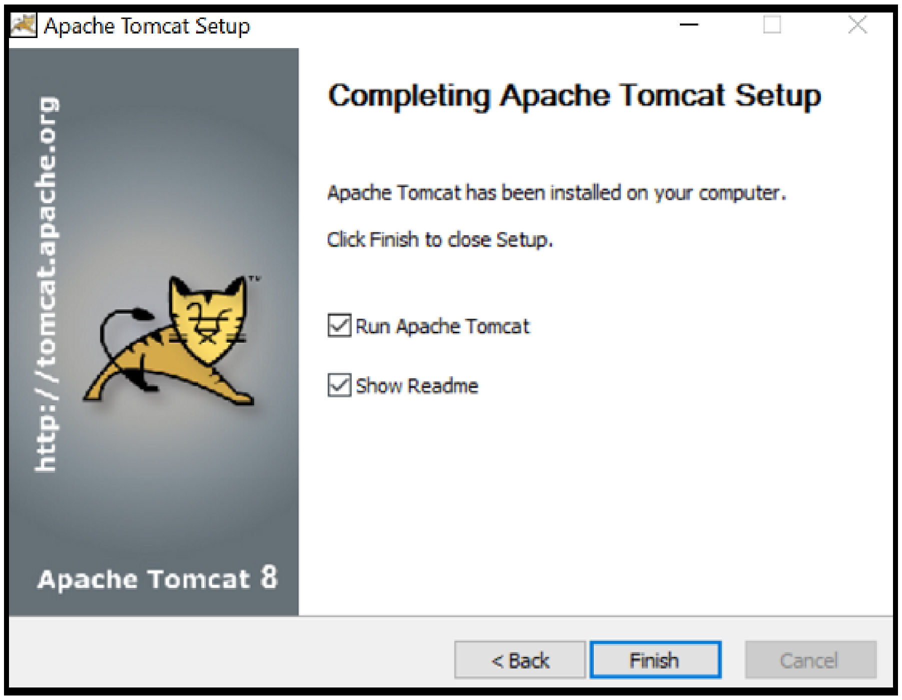
Setup will install Apache Tomcat in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder: C:\Program Files\Apache Software Foundation\Tomcat 8.5

Space required: 11.7 MB
Space available: 391.4 GB

Nullsoft Install System v3.03

16. Click **Finish**.


Apache Tomcat Setup

Completing Apache Tomcat Setup

Apache Tomcat has been installed on your computer.

Click Finish to close Setup.

☑ Run Apache Tomcat

☑ Show Readme

17. Launch Tomcat. Click **Configure Tomcat**. In the Configure Tomcat window, click the **Java** tab.
18. In the **Java Options** field, add the following to the end of the options field:

```
-Dserver -Dd64
-XX:+UseG1GC
-Dfile.encoding=UTF-8
-Djava.library.path=<path to Tomcat>\webapps\Thingworx\WEB-INF\extensions
```

Note

`Djava.library.path` example:

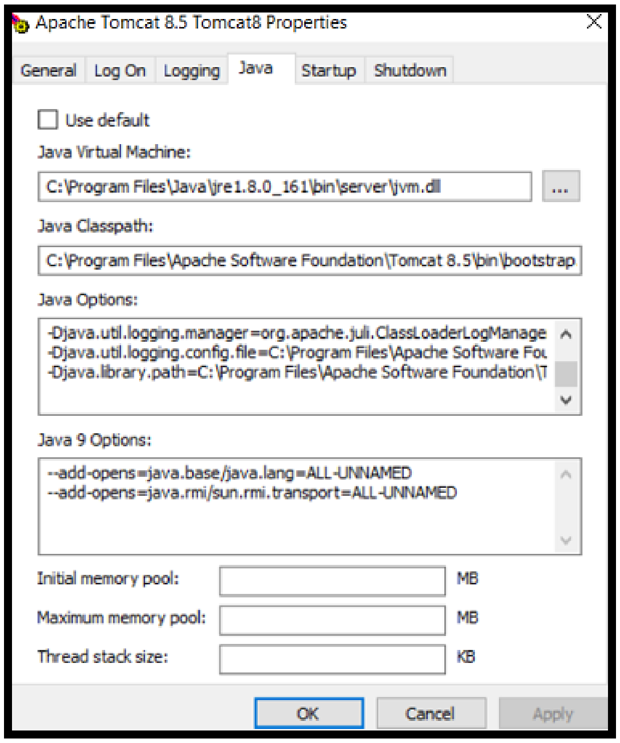
```
-Djava.library.path=C:\Program Files\Apache Software Foundation\Tomcat8.5\
webapps\
Thingworx\WEB-INF\extensions
```

If you are installing the ThingWorx Platform for the first time, the Java option `-Duser.timezone=UTC` should be set, where `UTC` does not recognize daylight savings time. Setting this option prevents overwriting data when daylight savings time changes occur. Existing customers should not update this setting at this time.

Note

For more information on these options and for additional options for hosted and/or public-facing environments, refer to the Apache Tomcat Java Option Settings on page 116.

19. Set the **Initial memory pool** and **Maximum memory pool** fields to 75% of the available OS memory (for example, 12GB for a 16GB RAM system). Refer to JVM Tuning for additional information.



☐ Use default

Java Virtual Machine:

```
C:\Program Files\Java\jre1.8.0_161\bin\server\jvm.dll
```

Java Classpath:

```
C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\bootstrap
```

Java Options:

```
-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManage
-Djava.util.logging.config.file=C:\Program Files\Apache Software Fou
-Djava.library.path=C:\Program Files\Apache Software Foundation\T
```

Java 9 Options:

```
--add-opens=java.base/java.lang=ALL-UNNAMED
--add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED
```

Initial memory pool:		MB
Maximum memory pool:		MB
Thread stack size:		KB

20. Click **OK**

21. In the location of the Tomcat installation, open `/conf/web.xml`. Replace the default error page (default is stacktrace) by adding the following into the `web.xml` file. Place the following within the `web-app` tag (after the `welcome-file-list` tag). A well-configured web application will override this default in `webapps/APP_NAME/WEB-INF/web.xml` so it won't cause problems.

```
<error-page><exception-type>java.lang.Throwable</exception-type><location>/
error.jsp</location></error-page>
```

22. In the location of the Tomcat installation, open `conf/server.xml`. Add the following inside the `<Host> </Host>` tags:

```
<Valve className="org.apache.catalina.valves.ErrorReportValve" showReport=
"false" showServerInfo="false" />
```

Note

For security reasons, it is critical that you disable the AJP connector, if not already done so by default, by performing the following step.

23. In the location of the Tomcat installation, open `conf/server.xml` and search for the following line. If found, comment it out and save the file:

    ```
    <Connector port ="8009" protocol="AJP/1.3" redirectPort="8443"/>
    ```

Note

In Apache Tomcat 9.0 and later, the **rejectIllegalHeader** attribute defaults to true. Manually modifying the `conf/web.xml` file to set this attribute to false is not recommended or supported by PTC.

24. Remove all the Tomcat webapps located in `/<path_to_tomcat>/webapps/`. Removing these apps prevents unnecessary access to Tomcat, specifically in the context that would allow users to view other users' cookies.
25. If your application requires a specific cipher suite, refer to the following documentation for configuration information:
    - https://www.jamf.com/jamf-nation/articles/384/configuring-supported-ciphers-for-tomcat-https-connections
26. PTC strongly recommends the use of TLS when running ThingWorx. For detailed instructions on setting up TLS, refer to this technical support article.
27. (OPTIONAL STEP) To increase the default cache settings that affect static file caching, add the following line within the `<context></context>` tags in the `$CATALINA_HOME/conf/context.xml` file:

    ```
    <Resources cacheMaxSize="501200" cacheObjectMaxSize="2048" cacheTtl="60000"/>
    ```

    Increasing this setting improves performance and avoids the following message in Tomcat:

    ```
    WARNING: Unable to add the resource at [/Common/jquery/jquery-ui.js] to the
    cache
     because there was insufficient
     free space available after evicting expired cache entries -
    consider increasing the maximum size of the cache
    ```
28. For H2 and Azure SQL: Go to Install ThingWorx on page 16.
29. For PostgreSQL: Go to Install and Configure PostgreSQL on page 28.

Install ThingWorx (Windows)

1. If you have not already done so, create a folder named `ThingworxPlatform` at the root of the drive where Tomcat was installed.

Note

Ensure the ThingWorx server has read and write access to the `ThingworxPlatform` and `ThingworxStorage` folders. Without these permissions, the server will fail to start.

2. If you have not already done so, obtain the `Thingworx.war` file from PTC Software Downloads.
3. Place the `platform-settings.json` in the `ThingworxPlatform` folder.
4. Configure the Administrator password. Add the following **AdministratorUserSettings** section (in **PlatformSettingsConfig**) to your `platform-settings.json` file along with a password that is at least 14 characters long. Reference platform-settings.json Configuration Details on page 121 for more information on placement. See Passwords for additional information on setting passwords. Do not copy and paste the sample below, as it may cause bad formatting in your `platform-settings.json`. Instead, click here and copy from the file.

```
{
    "PlatformSettingsConfig": {
        "AdministratorUserSettings": {
            "InitialPassword": "changeme"
        }
    }
}
```

Note

If Tomcat fails to start and reports the error message: `Check the InitialPassword setting in the AdministratorUserPassword section in platform-settings.json...`, check the following:

- The password setting exists in `platform-settings.json`
- The password is valid (14 or more characters by default)
- The `platform-settings.json` file is formatted correctly - bad formatting could lead to errors

5. Enable extension import. By default, extension import is disabled for all users. Add the following to the `platform-settings.json` file. Update the following *ExtensionPackageImportPolicy* parameters to `true` to allow extensions to be imported. See Importing Extensions for best practices on controlling extension import.

```
"ExtensionPackageImportPolicy": {
             "importEnabled": <true or false>,
             "allowJarResources": <true or false>,
             "allowJavascriptResources": <true or false>,
             "allowCSSResources": <true or false>,
             "allowJSONResources": <true or false>,
             "allowWebAppResources": <true or false>,
             "allowEntities": <true or false>,
             "allowExtensibleEntities": <true or false>
        },
```

6. Skip this step if you are not using H2 as a database. Add a username and password for H2 in the `platform-settings.json` file. See platform-settings.json Configuration Details on page 121 for more information.

Note

ThingWorx connections to the H2 database require a username and password defined by the user, or the server will not start. This design fully mitigates any potential vulnerability represented by CVE-2018-10054.

```
},
"PersistenceProviderPackageConfigs":{
  "H2PersistenceProviderPackage":{
      "ConnectionInformation":
{
    "password": "<addsecurepassword>",
     "username": "twadmin"
}
},
```

7. Skip this step if you are not using Azure SQL as a database. Open the `platform-settings.json` file and add the Azure SQL persistence provider parameters:

```
"PersistenceProviderPackageConfigs": {
"AzuresqlPersistenceProviderPackage": {
"ConnectionInformation": {
    "driverClass": "com.microsoft.sqlserver.jdbc.SQLServerDriver",
"jdbcUrl": "jdbc:sqlserver://<server name>:<port>;databaseName=thingworx;
applicationName=Thingworx;",
"password": "<database password>",
"username": "twadmin"
}
}
}
```

8. Configure licensing:

   - Open the `platform-settings.json` file and add the following to the `PlatformSettingsConfig` section (reference platform-settings.json Configuration Options on page 121 for more information on placement.)

   Note

   If you are performing a disconnected installation (no internet access), you do not need to add licensing information to the `platform-settings.json` file. Refer to the Licensing Guide for disconnected sites and skip this step.

   ```
   "LicensingConnectionSettings":{
         "username":"PTC Support site user name",
         "password":"PTC Support site password"
         }
   ```

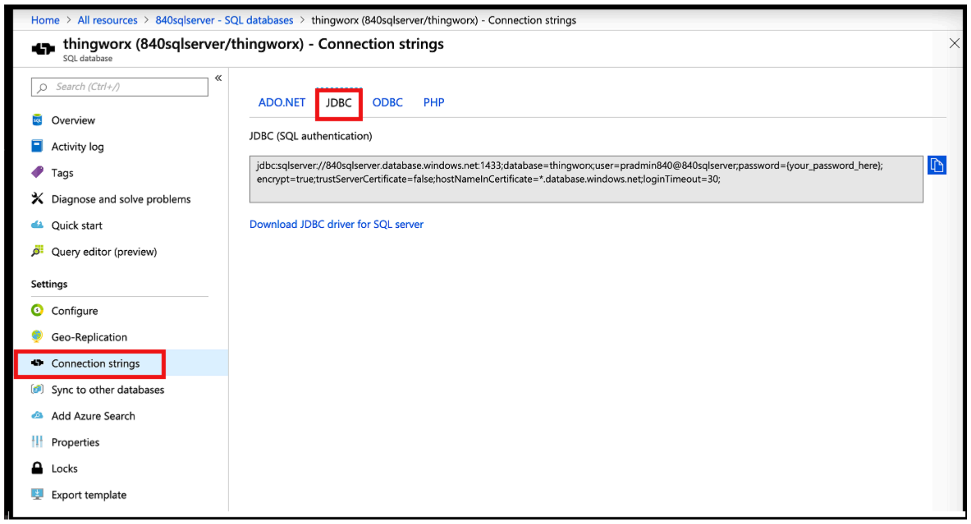
   - Stop Tomcat.
   - Copy the `Thingworx.war` file and place it in the following location of your Tomcat installation:
     `<Tomcat_Install_Location>\webapps`
   - Start Tomcat.
   - Verify that a license file (`successful_license_capability_response.bin`) is created in the `ThingworxPlatform` folder.

Note

If the settings are filled out incorrectly or if the server can't connect, a License Request text file (`licenseRequestFile.txt`) is created in the `ThingworxPlatform` folder. In this scenario, a license must be created manually. (If it is not created, ThingWorx will start in limited mode. Limited mode does not allow you to persist licensed entities to the database. Licensed entities are Things, Mashups, Masters, Gadgets, Users, and Persistence Providers).

More information on obtaining a ThingWorx disconnected site license through our License Management site can be found in the Licensing Guide for disconnected sites (no connection to PTC Support portal).

9. Encrypt the license server password by following the steps in Encrypting Passwords on page 118.
10. If you are using Azure SQL as your database, follow these steps to download the JDBC driver. Skip this step if you are not using Azure SQL.
    a. Go to the Azure portal and navigate to your ThingWorx database.
    b. Select **Connection strings**.
    c. Select the **JDBC** tab.


Home > All resources > 840sqlserver - SQL databases > thingworx (840sqlserver/thingworx) - Connection strings
thingworx (840sqlserver/thingworx) - Connection strings
SQL database
Search (Ctrl+/)
Overview
Activity log
Tags
Diagnose and solve problems
Quick start
Query editor (preview)
Settings
Configure
Geo-Replication
Connection strings
Sync to other databases
Add Azure Search
Properties
Locks
Export template
ADO.NET
JDBC
ODBC
PHP
JDBC (SQL authentication)
Download JDBC driver for SQL server

```
jdbc:sqlserver://840sqlserver.database.windows.net:1433;database=thingworx;user=pradmin840@840sqlserver;password={your_password_here};
encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;
```

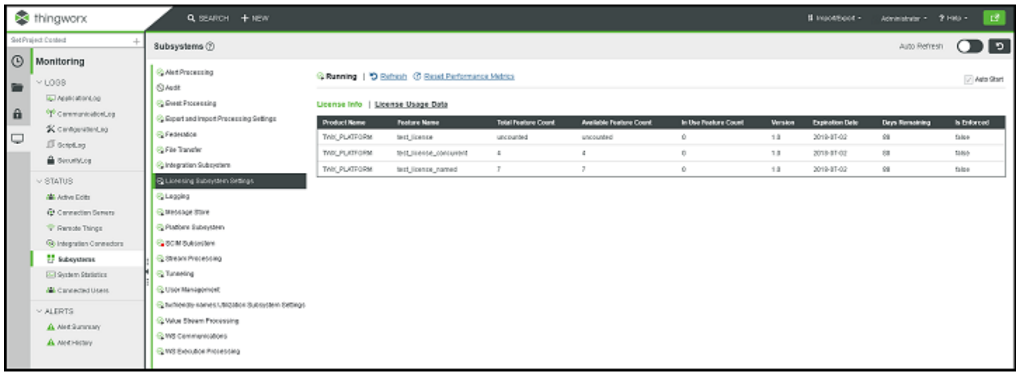
    d. Select **Download Microsoft JDBC Driver for SQL Server**.
    e. Select **Microsoft JDBC Driver 6.0 for SQL Server**.
    f. Extract and copy the downloaded binary in your ThingWorx VM to your Tomcat `lib` directory.
11. Start Tomcat.

12. To launch ThingWorx, go to `http://<servername>:<port>/Thingworx` in a Web browser.
13. Change the initial Administrator password.
    a. In Composer, select **Administrator** > **Change Password**.
    b. In the **Change Password** window, enter **Current Password**, **New Password**, and **Confirm Password**.

    Note

    The password, which should not be easily guessed or a known, common password, should be at least 14 characters in length and should include a mix of uppercase and lowercase letters, numbers, and special characters.

    c. Delete the initial password from the `platform-settings.json` file.
14. Select **Done**.
15. (OPTIONAL STEP) To determine the status of your license, open the **Monitoring**>**Subsystem**>**Licensing Subsystem Settings** in Composer to confirm the list of features (licensed entities) included with the license. If there are no licensed entities present, you are in limited mode.



Product Name	Feature Name	Total Feature Count	Available Feature Count	In Use Feature Count	Version	Expiration Date	Days Remaining	Is Enforced
TWX_PLATFORM	test_license	uncounted	uncounted	0	1.0	2019-07-02	98	false
TWX_PLATFORM	test_license_concurrent	4	4	0	1.0	2019-07-02	98	false
TWX_PLATFORM	test_license_named	7	7	0	1.0	2019-07-02	98	false

PostgreSQL

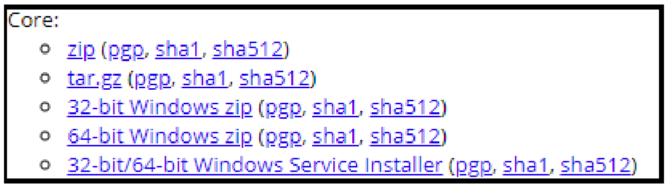
Install Java and Apache Tomcat (Windows)

1. If you are using AzureSQL for your database, go to Using Azure SQL Server as the Persistence Provider. Perform the steps in that section to set up the database, and you will be referred back to this section.

2. If you are using MSSQL for your database, go to Using MSSQL as the Persistence Provider. Perform the steps in that section to set up the database, and you will be referred back to this section.
3. Refer to the ThingWorx System Requirements for Java JDK version requirements.
4. Download and install the required version of the Java JDK from the Oracle website.
5. Ensure the Java environment variable is configured properly :
   -
     a. Locate your Java installation directory and copy the path. The default path is `C:\Program Files\Java\jdk_<version number>`.
     b. From the Windows start menu, navigate to **Advanced System Properties**. Your path to these properties will vary based on your version of Windows. For example, for Windows 10, search for **Environment Variables** then select **Edit the system environment variables**.
     c. Click **Environment Variables**.
     d. In the **System variables** section, click **New**.
     e. In the **Variable name** field, enter `JAVA_HOME`.
     f. In the **Variable value** field, enter the path to your Java installation as defined in step a.
     g. Click **OK**.
6. Refer to the ThingWorx System Requirements for Apache Tomcat version requirements.
7. Visit the Tomcat website to download the **32-bit/64-bit Windows Service Installer (pgp, sha1, sha512)**.

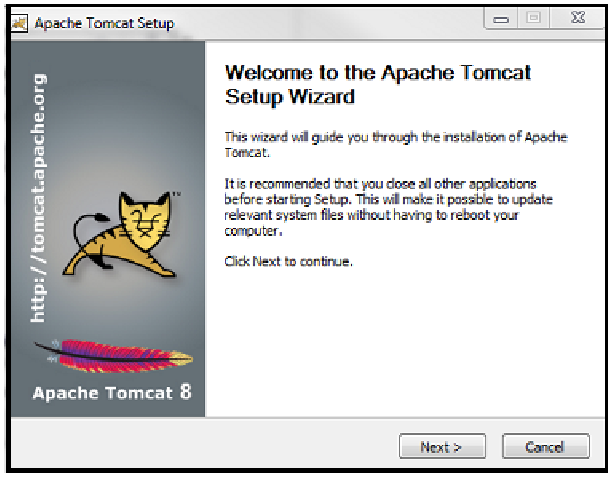
Note

Best practice includes verifying the integrity of the Tomcat file by using the signatures or checksums for each release. Refer to Apache's documentation for more information.



Core:
- zip (pgp, sha1, sha512)
- tar.gz (pgp, sha1, sha512)
- 32-bit Windows zip (pgp, sha1, sha512)
- 64-bit Windows zip (pgp, sha1, sha512)
- 32-bit/64-bit Windows Service Installer (pgp, sha1, sha512)

8. The Apache Tomcat Setup Wizard launches. Click **Next**.



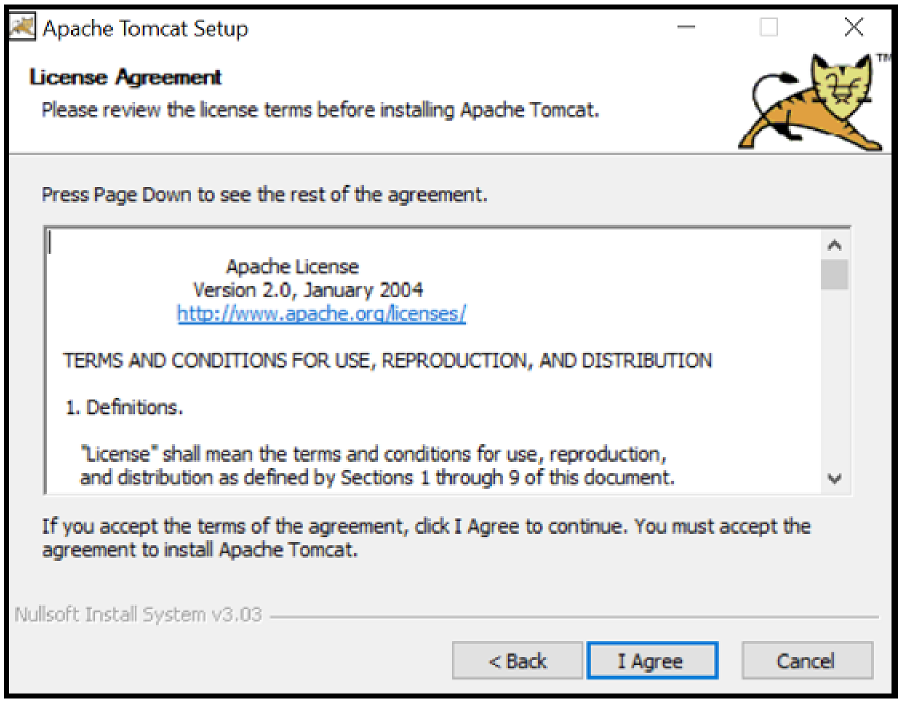
Welcome to the Apache Tomcat Setup Wizard

This wizard will guide you through the installation of Apache Tomcat.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

9. Click **I Agree**.



License Agreement

Please review the license terms before installing Apache Tomcat.

Press Page Down to see the rest of the agreement.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

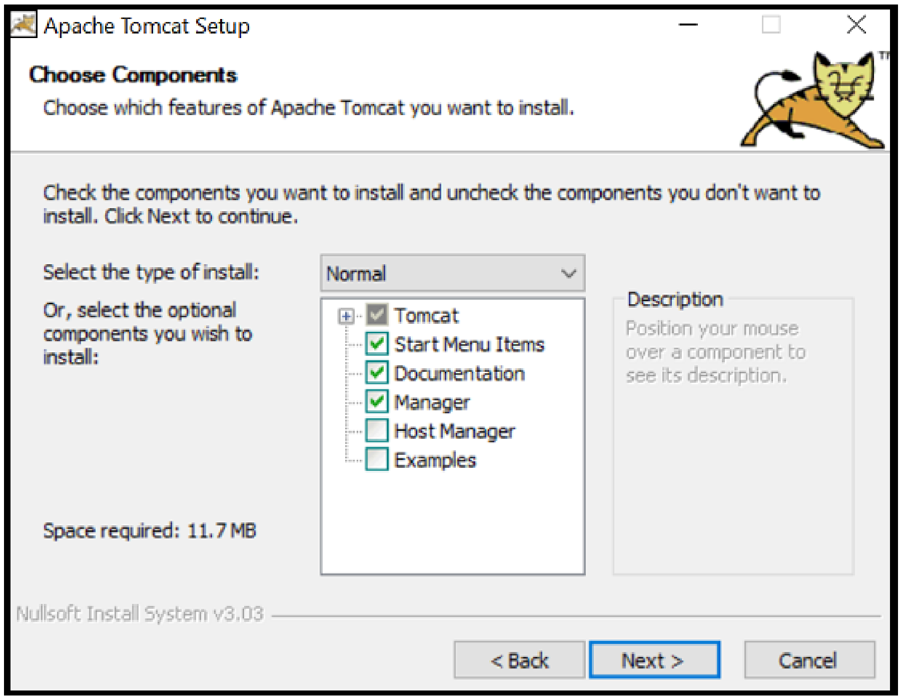
TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Apache Tomcat.

10. In the **Choose Components** section, use the default settings. Click **Next**.


Apache Tomcat Setup

Choose Components

Choose which features of Apache Tomcat you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select the type of install: Normal

Or, select the optional components you wish to install:

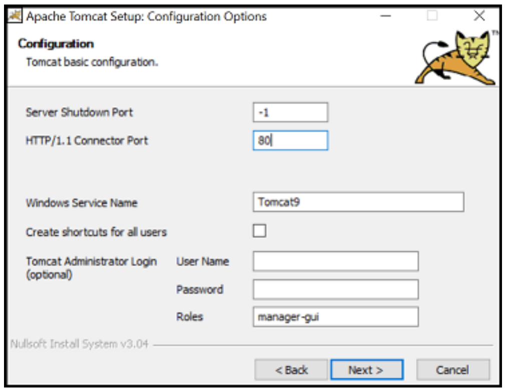
- [✓] Tomcat
- [✓] Start Menu Items
- [✓] Documentation
- [✓] Manager
- [ ] Host Manager
- [ ] Examples

Description: Position your mouse over a component to see its description.

Space required: 11.7 MB

Nullsoft Install System v3.03

11. In the **HTTP/1.1 Connector Port** field, type `80` (or other available port).
12. In the **Tomcat Administrator Login** fields, you must enter a `Tomcat user name` and a unique, secure password for Tomcat administration. In ThingWorx it is required, not optional.
13. Click **Next**.


Apache Tomcat Setup: Configuration Options

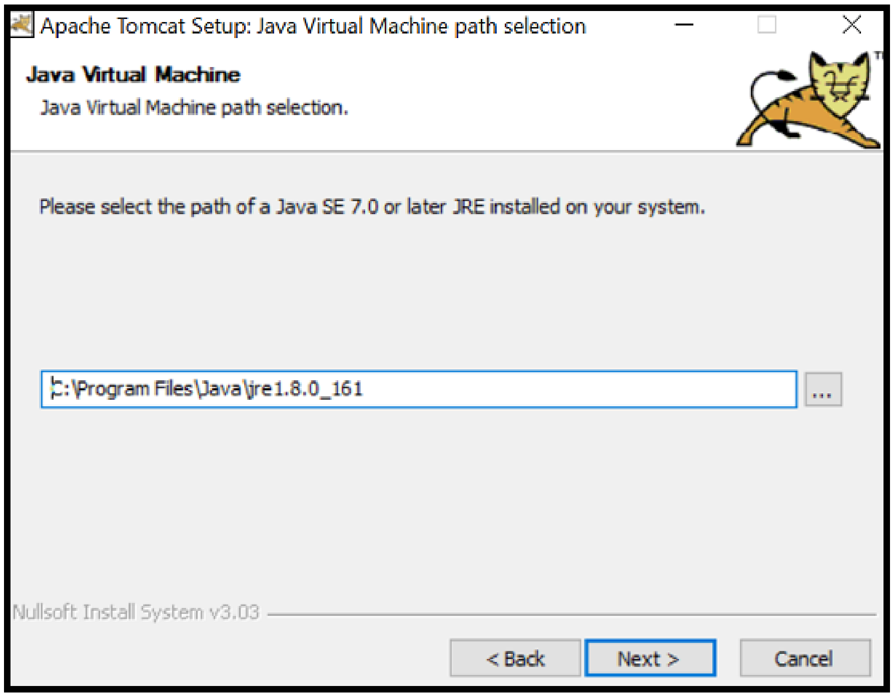
Configuration

Tomcat basic configuration.

Server Shutdown Port		-1
HTTP/1.1 Connector Port		80
Windows Service Name		Tomcat9
Create shortcuts for all users		☐
Tomcat Administrator Login (optional)	User Name	
	Password	
	Roles	manager-gui

Nullsoft Install System v3.04

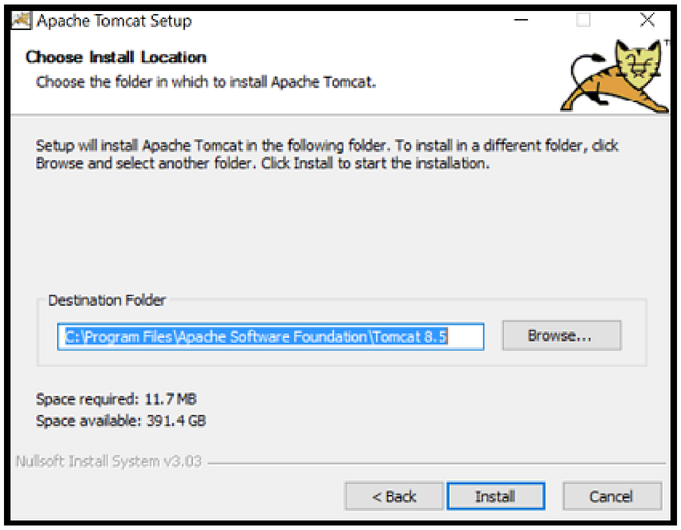
14. Click **Next**.


Apache Tomcat Setup: Java Virtual Machine path selection
Java Virtual Machine
Java Virtual Machine path selection.
Please select the path of a Java SE 7.0 or later JRE installed on your system.

C:\Program Files\Java\jre1.8.0_161

...
Nullsoft Install System v3.03
< Back
Next >
Cancel

15. Click **Install**.

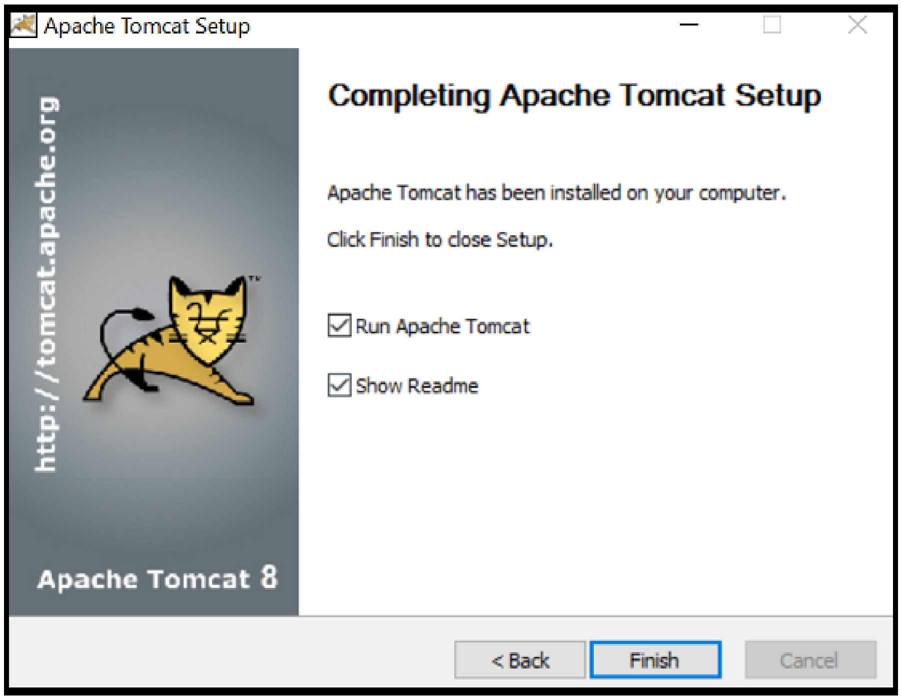

Apache Tomcat Setup
Choose Install Location
Choose the folder in which to install Apache Tomcat.
Setup will install Apache Tomcat in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder: C:\Program Files\Apache Software Foundation\Tomcat 8.5

Space required: 11.7 MB
Space available: 391.4 GB

Browse...
Nullsoft Install System v3.03
< Back
Install
Cancel

16. Click **Finish**.


Apache Tomcat Setup

Completing Apache Tomcat Setup

Apache Tomcat has been installed on your computer.

Click Finish to close Setup.

☑ Run Apache Tomcat

☑ Show Readme

17. Launch Tomcat. Click **Configure Tomcat**. In the Configure Tomcat window, click the **Java** tab.
18. In the **Java Options** field, add the following to the end of the options field:

```
-Dserver -Dd64
-XX:+UseG1GC
-Dfile.encoding=UTF-8
-Djava.library.path=<path to Tomcat>\webapps\Thingworx\WEB-INF\extensions
```

Note

`Djava.library.path` example:

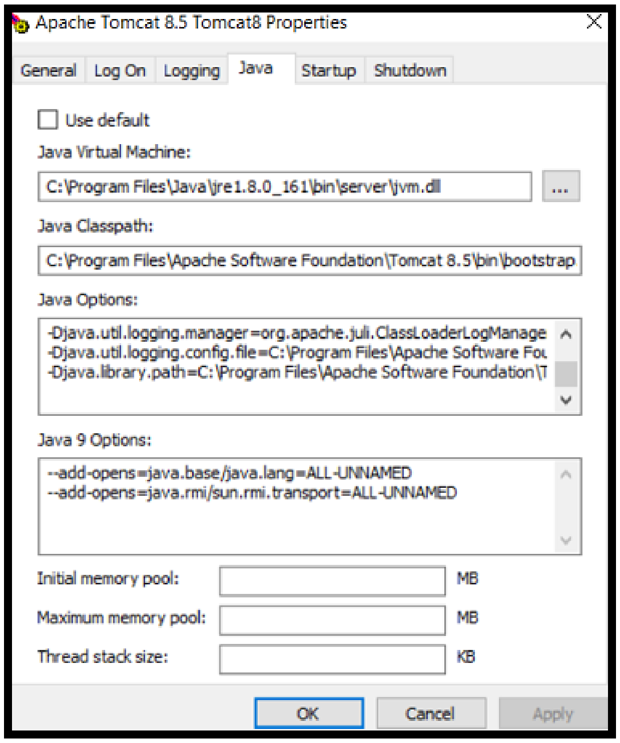
```
-Djava.library.path=C:\Program Files\Apache Software Foundation\Tomcat8.5\
webapps\
Thingworx\WEB-INF\extensions
```

If you are installing the ThingWorx Platform for the first time, the Java option `-Duser.timezone=UTC` should be set, where `UTC` does not recognize daylight savings time. Setting this option prevents overwriting data when daylight savings time changes occur. Existing customers should not update this setting at this time.

Note

For more information on these options and for additional options for hosted and/or public-facing environments, refer to the Apache Tomcat Java Option Settings on page 116.

19. Set the **Initial memory pool** and **Maximum memory pool** fields to 75% of the available OS memory (for example, 12GB for a 16GB RAM system). Refer to JVM Tuning for additional information.



Use default: ☐

Java Virtual Machine:
C:\Program Files\Java\jre1.8.0_161\bin\server\jvm.dll

Java Classpath:
C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\bootstrap

Java Options:
```
-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManage
-Djava.util.logging.config.file=C:\Program Files\Apache Software Fou
-Djava.library.path=C:\Program Files\Apache Software Foundation\T
```

Java 9 Options:
```
--add-opens=java.base/java.lang=ALL-UNNAMED
--add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED
```

Initial memory pool:		MB
Maximum memory pool:		MB
Thread stack size:		KB

20. Click **OK**

21. In the location of the Tomcat installation, open `/conf/web.xml`. Replace the default error page (default is stacktrace) by adding the following into the `web.xml` file. Place the following within the `web-app` tag (after the `welcome-file-list` tag). A well-configured web application will override this default in `webapps/APP_NAME/WEB-INF/web.xml` so it won't cause problems.

```
<error-page><exception-type>java.lang.Throwable</exception-type><location>/
error.jsp</location></error-page>
```

22. In the location of the Tomcat installation, open `conf/server.xml`. Add the following inside the `<Host> </Host>` tags:

```
<Valve className="org.apache.catalina.valves.ErrorReportValve" showReport=
"false" showServerInfo="false" />
```

Note

For security reasons, it is critical that you disable the AJP connector, if not already done so by default, by performing the following step.

23. In the location of the Tomcat installation, open `conf/server.xml` and search for the following line. If found, comment it out and save the file:

    ```
    <Connector port ="8009" protocol="AJP/1.3" redirectPort="8443"/>
    ```

Note

In Apache Tomcat 9.0 and later, the **rejectIllegalHeader** attribute defaults to true. Manually modifying the `conf/web.xml` file to set this attribute to false is not recommended or supported by PTC.

24. Remove all the Tomcat webapps located in `/<path_to_tomcat>/webapps/`. Removing these apps prevents unnecessary access to Tomcat, specifically in the context that would allow users to view other users' cookies.
25. If your application requires a specific cipher suite, refer to the following documentation for configuration information:
    - https://www.jamf.com/jamf-nation/articles/384/configuring-supported-ciphers-for-tomcat-https-connections
26. PTC strongly recommends the use of TLS when running ThingWorx. For detailed instructions on setting up TLS, refer to this technical support article.
27. (OPTIONAL STEP) To increase the default cache settings that affect static file caching, add the following line within the `<context></context>` tags in the `$CATALINA_HOME/conf/context.xml` file:

    ```
    <Resources cacheMaxSize="501200" cacheObjectMaxSize="2048" cacheTtl="60000"/>
    ```

    Increasing this setting improves performance and avoids the following message in Tomcat:

    ```
    WARNING: Unable to add the resource at [/Common/jquery/jquery-ui.js] to the
    cache
     because there was insufficient
     free space available after evicting expired cache entries -
    consider increasing the maximum size of the cache
    ```
28. For H2 and Azure SQL: Go to Install ThingWorx on page 16.
29. For PostgreSQL: Go to Install and Configure PostgreSQL on page 28.

Install and Configure PostgreSQL (Windows)

The instructions provided below are intended for the PostgreSQL administrator (not the DB host servers). If you are including the HA layer to your implementation, refer to the the ThingWorx High Availability Administrator's Guide.

Install PostgreSQL and Create a New User Role

1. Refer to ThingWorx System Requirements for information on supported PostgreSQL versions.

> **Note**
>
> The steps in this procedure use PostgreSQL version *x.x*, where *x.x* is the supported version.

2. Download and install the appropriate version of PostgreSQL from http://www.postgresql.org/download/.
3. Open PostgreSQL using PgAdmin. The PgAdmin tool is available in the PostgreSQL download.

> **Note**
>
> PgAdmin is an open source management tool for your databases that is included in the PostgreSQL download. The tool features full Unicode support, fast, multi-threaded query, and data editing tools and support for all PostgreSQL object types.

4. Create a new user role:
   a. Right click **PostgreSQLx.x (<IP or host name of the database>:<Port number of PostgreSQL>)**. Example: `PostgreSQLx.x (localhost:5432)`
   b. Select **New Object**>**New Login Role**. On the **Properties** tab, in the **Role name** field, enter the <PostgreSQL user role name> for PostgreSQL administration.
   c. On the **Definition** tab, in the **Password** field, enter a unique and secure password for PostgreSQL administration (you will be prompted to enter it twice).

Note

The password, which should not be easily guessed or a known, common password, should be at least 14 characters in length and include a mix of uppercase and lowercase letters, numbers, and special characters. You will need to re-enter this password in later steps.

5. Click **OK**. Note the user role name created in this step for later use.

Configure PostgreSQL Database Located on a Separate Server than ThingWorx

Note

This section is optional for development environments, but should be implemented in all production environments.

By default, the PostgreSQL server is installed in a locked-down state. The server will only listen for connections from the local machine. For ThingWorx to communicate to the PostgreSQL server, some configuration changes need to be made so that PostgreSQL knows to listen for connections from other users (thingworx user, default is twadmin) and/or other machines (ThingWorx installed on a separate server).

You will need to know where your PostgreSQL data directory resides for these steps. On Windows, the default data folder is `C:\Program Files\PostgreSQL\x.x\data`.

Modify the `pg_hba.conf` file and add the following lines based on your desired configuration:

If you want to allow all IPv4 addresses to connect:	`hostallall0.0.0.0/0md5`
If you want to allow only a specific IPv4 address to connect (Replace <*ipAddress*> with the IP address of the machine making the connection):	`hostallall<ipAddress>/32md5`
If you want to allow all IPv6 addresses to connect:	`hostallall::0/0md5`
If you want to allow only a specific IPv6 address to connect (Replace <ipv6Address> with the appropriate address):	`hostallall<ipv6Address>/128md5`

Any other combination is possible by using additional allowance lines (individual IPs or ranges) or subnet masks appropriate to the machines that require access to the PostgreSQL database.

Any change to this file requires a restart of the database service.

Note

For additional information about configuring the `pg_hba.conf`file, see the official PostgreSQL documentation.

Configure and Execute the PostgreSQL Database Script

To set up the PostgreSQL database and tablespace, the `thingworxPostgresDBSetup` script must be configured and executed.

1. Add the `<postgres-installation>/bin` folder to your system **PATH** variable.
2. Create a folder named `ThingworxPostgresqlStorage` on the drive that the `ThingworxStorage` folder is located (in the root directory by default). Note the following:
   - If you create the folder using the `-d<databasename>` command, you do not have to use the PostgreSQL user.
   - You must specify the `-l` option to a path that exists. For example, `-l D:\ThingworxPostgresqlStorage`. The script does not create the folder for you.
   - The folder must have appropriate ownership and access rights. It should be owned by the same user who runs the PostgreSQL service, and have Full Control assigned to that user - this user is generally NETWORK_ SERVICE, but may differ in your environment.
3. Obtain and open the `thingworxPostgresDBSetup` script from the ThingWorx software download package. ThingWorx downloads are available in PTC Software Downloads.
4. If necessary, configure the script. Reference the options in the table below.

thingworxPostgresDBSetup Script Options

Option	Parameter	Default	Description	Example
t or -T	tablespace	thingworx	Tablespace name	`-t thingworx`
-p or -P	port	5432	Port number of PostgreSQL	`-p 5432`

thingworxPostgresDBSetup Script Options (continued)

Option	Parameter	Default	Description	Example
-d or -D	database	thingworx	PostgreSQL Database name to create	`-d thingworx`
-h or -H	host	localhost	Name of the host	`-h localhost`
-l or -L	tablespace_location	/Thingworx-Postgresql-Storage	Required. Location in the file system where the files representing database objects are stored.	`-l` or `-L`
-a or -A	adminuser-name	postgres	Administrator Name	`-a postgres`
-u or -U	thingworxu-sername	twadmin	User name that has permissions to write to the database.	`-u twadmin`

5. Execute the script.

Configure and Execute the Model/Data Provider Schema Script

To set up the PostgreSQL model/data provider schema, the `thingworxPostgresSchemaSetup` script must be configured and executed. This script will set up the public schema under your database on the PostgreSQL instance installed on the localhost.

1. Obtain the `thingworxPostgresSchemaSetup.bat` from the ThingWorx software download package. ThingWorx downloads are available in PTC Software Downloads.
2. If necessary, configure the script. Reference the options in the table below.

thingworxPostgresSchemaSetup Script Options

Option	Parameter	Default	Description	Example
-h or -H	host	localhost	IP or host name of the database.	`-h localhost`
-p or -P	port	5432	Port number	`-p 5432`

thingworxPostgresSchemaSetup Script Options (continued)

Option	Parameter	Default	Description	Example
			of PostgreSQL.	
-d or -D	database	thingworx	Database name to use.	`-d thingworx`
-s or -S	schema	public	Schema name to use.	`-s mySchema`
-u or -U	username	twadmin	Username to update the database schema	`-u twadmin`
-o or -O	option	all	There are three options: • all: Sets up the model and data provider schemas into the specified database. • model: Sets up the model provider schema into the specified database. • data: Sets up the data provider schema into the specified database.	`-o data`

3. Execute the script.

Configure platform-settings.json

1. Create the folder `ThingworxPlatform` at the root of the drive where Tomcat was installed or set a system environment variable that points to the folder. Note the following:
   - To specify the location where ThingWorx stores its settings, you can set the `THINGWORX_PLATFORM_SETTINGS` environment variable to the desired location. Ensure that the folder referenced by `THINGWORX_PLATFORM_SETTINGS` exists and is writable by the Tomcat user. This environment variable should be configured as part of the system environment variables.
   - The ThingWorx server will fail to start if it does not have read and write access to this folder.
2. Place the `platform-settings.json` file into the `ThingworxPlatform` folder. This file is available in the software download.
3. Open `platform-settings.json` and configure as necessary. Refer to the configuration options in platform-settings.json Configuration Details on page 121.

Note

If your PostgreSQL server is not the same as your ThingWorx server, and you are having issues with your ThingWorx installation, review your Tomcat logs and `platform-settings.json` file. The default installation assumes both servers are on the same machine.

(Optional) Encrypt the PostgreSQL Password

Encrypt the password by following the steps in Encrypting Passwords on page 118.

(Optional) Installing the PostgreSQL Client Package and PostgreSQL User

To issue PostgreSQL commands from the client machine to the PostgreSQL server, do so from a PostgreSQL user. The `postgresql-client-x.x` package can be installed on the client machine. Refer to your PostgreSQL distribution documentation on how to install it. This package provides some administrative tools such as `psql`.

Install ThingWorx

Go to Install ThingWorx on page 34.Installation steps are also available in the Help Center..

Install ThingWorx (Windows)

1. If you have not already done so, create a folder named `ThingworxPlatform` at the root of the drive where Tomcat was installed.

   Note

   Ensure the ThingWorx server has read and write access to the `ThingworxPlatform` and `ThingworxStorage` folders. Without these permissions, the server will fail to start.

2. If you have not already done so, obtain the `Thingworx.war` file from PTC Software Downloads.
3. Place the `platform-settings.json` in the `ThingworxPlatform` folder.
4. Configure the Administrator password. Add the following **AdministratorUserSettings** section (in **PlatformSettingsConfig**) to your `platform-settings.json` file along with a password that is at least 14 characters long. Reference platform-settings.json Configuration Details on page 121 for more information on placement. Passwords for additional information on setting passwords. Do not copy and paste the sample below, as it may cause bad formatting in your `platform-settings.json`. Instead, click here and copy from the file.

```
{
    "PlatformSettingsConfig": {
        "AdministratorUserSettings": {
            "InitialPassword": "changeme"
        }
    }
}
```

Note

If Tomcat fails to start and reports the error message: `Check the InitialPassword setting in the AdministratorUserPassword section in platform-settings.json...`, check the following:

- The password setting exists in `platform-settings.json`
- The password is valid (14 or more characters by default)
- The `platform-settings.json` file is formatted correctly - bad formatting could lead to errors

5. Enable extension import. By default, extension import is disabled for all users. Add the following to the `platform-settings.json` file. Update the following *ExtensionPackageImportPolicy* parameters to `true` to allow extensions to be imported. Importing Extensions for best practices on configuration.

```
"ExtensionPackageImportPolicy": {
             "importEnabled": <true or false>,
             "allowJarResources": <true or false>,
             "allowJavascriptResources": <true or false>,
             "allowCSSResources": <true or false>,
             "allowJSONResources": <true or false>,
             "allowWebAppResources": <true or false>,
             "allowEntities": <true or false>,
             "allowExtensibleEntities": <true or false>
        },
```

6. Configure licensing:

   - Open the `platform-settings.json` file and add the following to the `PlatformSettingsConfig` section (reference platform-settings.json Configuration Options on page 121 for more information on placement.)

   Note

   If you are performing a disconnected installation (no internet access), you do not need to add to the `platform-settings.json` file. Refer to the Licensing Guide for disconnected sites and skip this step.

   ```
   "LicensingConnectionSettings":{
         "username":"PTC Support site user name",
         "password":"PTC Support site password"
         }
   ```

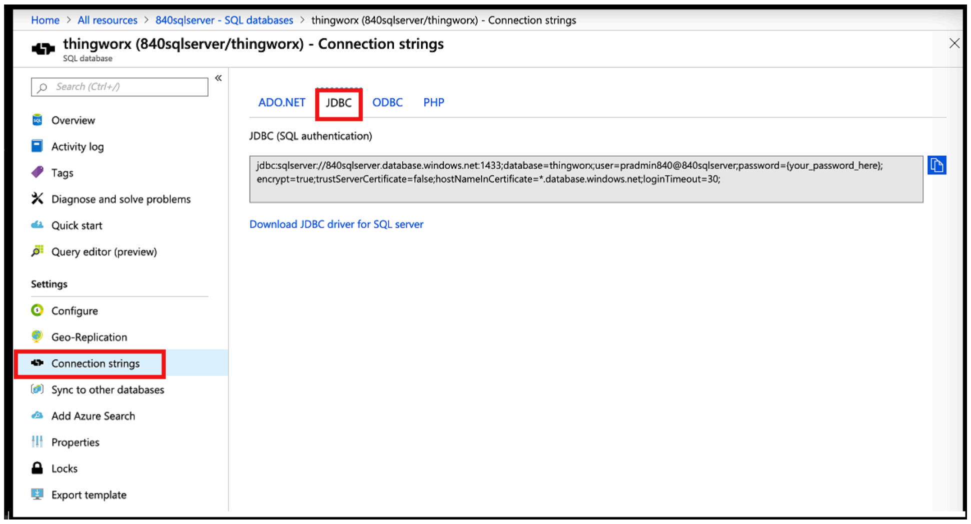
- Stop Tomcat.
- Copy the `Thingworx.war` file and place it in the following location of your Tomcat installation:
  `<Tomcat_Install_Location>\webapps`
- Start Tomcat.
- Verify that a license file (`successful_license_capability_response.bin`) is created in the `ThingworxPlatform` folder.

Note

If the settings are filled out incorrectly or if the server can't connect, a License Request text file (`licenseRequestFile.txt`) is created in the `ThingworxPlatform` folder. In this scenario, a license must be created manually. (If it is not created, ThingWorx will start in limited mode. Limited mode does not allow you to persist licensed entities to the database. Licensed entities are Things, Mashups, Masters, Gadgets, Users, and Persistence Providers).

More information on obtaining a ThingWorx disconnected site license through our License Management site can be found in the Licensing Guide.

7. Encrypt the license server password by following the steps in Encrypting Passwords on page 118.
8. If you are using Azure SQL as your database, follow these steps to download the JDBC driver. Skip this step if you are not using Azure SQL.
   a. Go to the Azure portal and navigate to your ThingWorx database.
   b. Select **Connection strings**.
   c. Select the **JDBC** tab.


Home > All resources > 840sqlserver - SQL databases > thingworx (840sqlserver/thingworx) - Connection strings
thingworx (840sqlserver/thingworx) - Connection strings
SQL database
Search (Ctrl+/)
Overview
Activity log
Tags
Diagnose and solve problems
Quick start
Query editor (preview)
Settings
Configure
Geo-Replication
Connection strings
Sync to other databases
Add Azure Search
Properties
Locks
Export template
ADO.NET
JDBC
ODBC
PHP
JDBC (SQL authentication)
Download JDBC driver for SQL server

```
jdbc:sqlserver://840sqlserver.database.windows.net:1433;database=thingworx;user=pradmin840@840sqlserver;password={your_password_here};
encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;
```

d. Select **Download Microsoft JDBC Driver for SQL Server**.

e. Select **Microsoft JDBC Driver 6.0 for SQL Server**.

f. Extract and copy the downloaded binary in your ThingWorx VM to your Tomcat `lib` directory.

9. Start Tomcat.

10. To launch ThingWorx, go to `http://<servername>:<port>/Thingworx` in a Web browser.

11. Change the default password:

    a. In Composer, select **Administrator** > **Change Password**.

    b. In the **Change Password** window, enter **Current Password**, **New Password**, and **Confirm Password**.

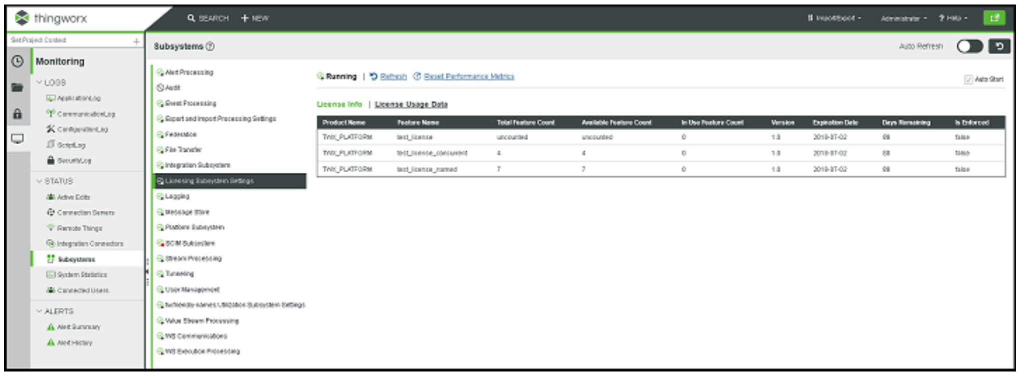
    Note

    The password, which should not be easily guessed or a known, common password, is recommended to be at least 14 characters in length and should include a mix of uppercase and lowercase letters, numbers, and special characters.

    c. Delete the initial password from the `platform-settings.json` file.

12. Select **Done**.

13. (OPTIONAL STEP) To determine the status of your license, open the **Monitoring**>**Subsystem**>**Licensing Subsystem Settings** in Composer to confirm the list of features (licensed entities) included with the license. If there are no licensed entities present, you are in limited mode.



Product Name	Feature Name	Total Feature Count	Available Feature Count	In Use Feature Count	Version	Expiration Date	Days Remaining	Is Enforced
TWX_PLATFORM	test_license	uncounted	uncounted	0	1.0	2019-07-02	98	false
TWX_PLATFORM	test_license_concurrent	4	4	0	1.0	2019-07-02	98	false
TWX_PLATFORM	test_license_named	7	7	0	1.0	2019-07-02	98	false

3

Ubuntu Installation

- H2/Azure SQL on page 40
- PostgreSQL on page 52

Note

See ThingWorx Installation Overview on page 6 for other options.

H2/Azure SQL

Install Java and Apache Tomcat (Ubuntu)

In the steps below, replace **xx** or **xxx** with the build number you are using.

1. If you are using AzureSQL for your database, go to Using Azure SQL Server as the Persistence Provider. Perform the steps in that section to set up the database, and return to this section.
2. If you are using MSSQL for your database, go to Using MSSQL as the Persistence Provider. Perform the steps in that section to set up the database, and you will be referred back to this section.
3. Update Ubuntu packages:

   ```
   $ sudo apt-get update
   ```
4. Install and Configure Network Time Protocol (NTP) settings for time synchronization. The default configuration for NTP is sufficient. For additional configuration information about NTP (beyond the scope of this documentation), refer to the following resources:
   - Time Synchronization with NTP
   - How do I use pool.ntp.org?

   ```
   $ sudo apt-get install ntp
   ```
5. Edit AUTHBIND properties to allow Tomcat to bind to ports below 1024:

   ```
   $ sudo apt-get install authbind
   ```
6. Refer to the ThingWorx System Requirements for version requirements. Download the appropriate Java JDK tar file from Oracle's website.
7. Extract the tar file:

   ```
   $ tar -xf jdk-8uxxx-linux-x64.tar.gz
   ```
8. Create the directory by moving the JDK to `/usr/lib/jvm`. If the directory is not empty, a warning message will display.

   ```
   $ sudo mkdir -p /usr/lib/jvm
   $ sudo mv jdk1.8.0_xxx/ /usr/lib/jvm/
   ```
9. Add alternatives to the system:

   ```
   $ sudo update-alternatives --install "/usr/bin/java" "java" "/usr/lib/jvm/
   jdk1.8.0_xxx/bin/java" 1
   $ sudo update-alternatives --install "/usr/bin/keytool" "keytool" "/usr/lib/
   jvm/jdk1.8.0_xxx/bin/keytool" 1
   ```
10. Change access permissions:

    ```
    $ sudo chmod a+x /usr/bin/java
    $ sudo chmod a+x /usr/bin/keytool
    ```
11. Change owner:

    ```
    $ sudo chown -R root:root /usr/lib/jvm/jdk1.8.0_xxx/
    ```
12. Configure master links:

```
$ sudo update-alternatives --config java
$ sudo update-alternatives --config keytool
```

`Nothing to configure` is a normal response to this command and is not an error. Additional executables in `/usr/lib/jvm/jdk1.8.0_xxx/bin/` can be installed using the previous set of steps.

13. Verify Java version:

    ```
    $ java -version
    ```

    This should return something similar to the following (build specifics may be different):

    ```
    java version "1.8.0_xxx"
    Java(TM) SE Runtime Environment (build 1.8.0_xxx-bxx)
    Java HotSpot(TM) 64-Bit Server VM (build 24.75-bxx, mixed mode)
    ```

14. Download Apache Tomcat. The steps in this process use Tomcat 8.5.*xx*, where *xx* is replaced with the version you are using.

    ```
    $ wget http://archive.apache.org/dist/tomcat/tomcat-8/v8.5.xx/bin/apache-
    tomcat-8.5.xx.tar.gz
    ```

Note

Best practice includes verifying the integrity of the Tomcat file by using the signatures or checksums for each release. Refer to Apache’s documentation for more information.

15. Extract the tar file:

    ```
    $ tar -xf apache-tomcat-8.5.xx.tar.gz
    ```

16. Create and change the owner for `/usr/share/tomcat8.5` and move Tomcat to the following location. Add user and group to the system:

    ```
    $ sudo mkdir -p /usr/share/tomcat8.5
    $ sudo mv apache-tomcat-8.5.xx /usr/share/tomcat8.5/8.5.xx
    $ sudo addgroup --system tomcat8.5 --quiet -force-badname
    $ sudo adduser --system --home /usr/share/tomcat8.5/ --no-create-home --ingroup
    tomcat8.5 --disabled-password --force-badname --shell /bin/false tomcat8.5
    $ sudo chown -R tomcat8.5:tomcat8.5 /usr/share/tomcat8.5
    ```

17. Define environment variables in `/etc/environment`:

    ```
    $ export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_xxx
    $ export CATALINA_HOME=/usr/share/tomcat8.5/8.5.xx
    ```

18. Change directory to `$CATALINA_HOME`:

    ```
    $ cd $CATALINA_HOME
    ```

19. Change owner and access permissions of `bin/`, `lib/`, and `webapps/`:

    ```
    $ sudo chown -Rh tomcat8.5:tomcat8.5 bin/ lib/ webapps/
    $ sudo chmod 775 bin/ lib/ webapps/
    ```

20. Change owner and access permissions of `usr/share/tomcat8.5/8.5xx`:

```
sudo chown -R tomcat8.5:tomcat8.5 /usr/share/tomcat8.5/8.5.xx
sudo chmod -R 775 /usr/share/tomcat8.5/8.5.xx
```

21. Change owner and access permissions of `conf/`:

    ```
    $ sudo chown -Rh root:tomcat8.5 conf/
    $ sudo chmod -R 650 conf/
    ```

22. Change access permissions of `logs/`, `temp/`, and `work/`:

    ```
    $ sudo chown -R tomcat8.5:adm logs/ temp/ work/
    $ sudo chmod 760 logs/ temp/ work/
    ```

23. Create a self-signed certificate:

    ```
    $ sudo $JAVA_HOME/bin/keytool -genkey -alias tomcat8.5 -keyalg RSA -keystore
    $CATALINA_HOME/conf/.keystore
    ```

24. Follow the instructions to complete the certificate creation process.

    - Set the keystore password.
    - Follow the prompts to set up your security certificate.
    - Set the tomcat8.5 user password to the same as the keystore password:

      ```
      $ sudo chown root:tomcat8.5 $CATALINA_HOME/conf/.keystore
      $ sudo chmod 640 $CATALINA_HOME/conf/.keystore
      ```

25. Uncomment the `Manager` element in `$CATALINA_HOME/conf/context.xml` to prevent sessions from persisting across restarts:

    ```
    <Manager pathname="" />
    ```

Note

For security reasons, it is critical that you disable the AJP connector, if not already done so by default, by performing the following step.

26. In the location of the Tomcat installation, open `conf/server.xml` and search for the following line. If found, comment it out and save the file:

    ```
    <Connector port ="8009" protocol="AJP/1.3" redirectPort="8443"/>
    ```

Note

In Apache Tomcat 9.0 and later, the **rejectIllegalHeader** attribute defaults to true. Manually modifying the `conf/web.xml` file to set this attribute to false is not recommended or supported by PTC.

Note

If you receive an error that the directory doesn't exist, use the following commands to ensure port 443 works:

```
sudo touch /etc/authbind/byport/443
sudo chmod 700 /etc/authbind/byport/443
sudo chown tomcat8.5:tomcat8.5 /etc/authbind/byport/443
```

27. Define a user in `$CATALINA_HOME/conf/tomcat-users.xml`:

    ```
    sudo vi $CATALINA_HOME/conf/tomcat-users.xml
    <user username="<Tomcat user name> " password="<Tomcat password> " roles=
    "manager"/>
    ```

28. Determine the uid of tomcat8.5 user:

    ```
    $ id -u tomcat8.5
    ```

29. Using this number, create an ID file in `/etc/authbind/byuid/`. Change the `<uid>` to the number that was returned in the previous step:

    ```
    $ sudo touch /etc/authbind/byuid/<uid>
    sudo vi /etc/authbind/byuid/<uid>
    ```

30. Edit the file from the step above and paste in the following:

    ```
    0.0.0.0/0:1,1023
    ```

31. Change owner and access permissions of `/etc/authbind/byuid/<uid>`:

    ```
    $ sudo chown tomcat8.5:tomcat8.5 /etc/authbind/byuid/<uid>
    $ sudo chmod 700 /etc/authbind/byuid/<uid>
    ```

32. Modify `$CATALINA_HOME/bin/startup.sh` to always use authbind:

    ```
    sudo vi $CATALINA_HOME/bin/startup.sh
    ```

    Comment the following in the file:

    ```
    #exec "$PRGDIR"/"$EXECUTABLE" start "$@"
    ```

33. Add the following to the end of the file:

    ```
    exec authbind --deep "$PRGDIR"/"$EXECUTABLE" start "$@"
    ```

34. In `/etc/init.d`, create tomcat8.5 file:

    ```
    $ sudo touch /etc/init.d/tomcat8.5
    ```

35. Edit the file and enter the following contents:

    ```
    $ sudo vi /etc/init.d/tomcat8.5

    CATALINA_HOME=/usr/share/tomcat8.5/8.5.xx

    case $1 in
      start)
        /bin/su -p -s /bin/sh tomcat8.5 $CATALINA_HOME/bin/startup.sh
      ;;
    ```

```
  stop)
    /bin/su -p -s /bin/sh tomcat8.5 $CATALINA_HOME/bin/shutdown.sh
  ;;

  restart)
    /bin/su -p -s /bin/sh tomcat8.5 $CATALINA_HOME/bin/shutdown.sh
    /bin/su -p -s /bin/sh tomcat8.5 $CATALINA_HOME/bin/startup.sh
  ;;

esac
exit 0
```

36. Change access permissions of `etc/init.d/tomcat8.5` and create symbolic links:

```
$ sudo chmod 755 /etc/init.d/tomcat8.5
$ sudo ln -s /etc/init.d/tomcat8.5 /etc/rc1.d/K99tomcat
$ sudo ln -s /etc/init.d/tomcat8.5 /etc/rc2.d/S99tomcat
```

37. Set up Tomcat as a service to start on boot. First, build JSVC. If JSVC is already installed on your system go to the next step.

```
$ sudo apt-get install gcc
```

38. Set up the Tomcat service on boot:

```
$ cd /usr/share/tomcat8.5/8.5.xx/bin/
$ sudo tar xvfz commons-daemon-native.tar.gz
$ cd commons-daemon-*-native-src/unix
$ sudo ./configure --with-java=$JAVA_HOME
$ sudo apt-get install make
$ sudo make
$ sudo cp jsvc ../..
```

39. Create the Tomcat service file:

```
sudo touch /etc/systemd/system/tomcat8.5.service
```

40. Open `/etc/systemd/system/tomcat8.5.service` in a text editor (as root):

```
sudo vi /etc/systemd/system/tomcat8.5.service
```

   a. Paste the following in the Tomcat service file:

Note

In the example below, set values for **-Xms** and **-Xmx** to 75% of the available OS memory (for example, 12GB for a 16GB RAM system). Refer to JVM Tuning for additional information.

```
[Unit]
Description=Apache Tomcat Web Application Container
After=network.target
```

```
[Service]
Type=forking
PIDFile=/var/run/tomcat.pid
Environment=CATALINA_PID=/var/run/tomcat.pid
Environment=JAVA_HOME=/usr/lib/jvm/jdk1.8.0_xxx
Environment=CATALINA_HOME=/usr/share/tomcat8.5/8.5.xx
Environment=CATALINA_BASE=/usr/share/tomcat8.5/8.5.xx
Environment=CATALINA_OPTS=

ExecStart=/usr/share/tomcat8.5/8.5.xx/bin/jsvc \
                              -Dcatalina.home=${CATALINA_HOME} \
                              -Dcatalina.base=${CATALINA_BASE} \
                              -Djava.awt.headless=true -Djava.net.
preferIPv4Stack=true -Dserver -Dd64 -XX:+UseNUMA \
                              -XX:+UseG1GC -Dfile.encoding=UTF-8 \
                              -Djava.library.path=${CATALINA_BASE}/webapps/
Thingworx/WEB-INF/extensions \
                              -Xms=<75% of available OS memory> \
                              -Xmx=<75% of available OS memory> \
                              -cp ${CATALINA_HOME}/bin/commons-daemon.jar:
${CATALINA_HOME}/bin/bootstrap.jar:${CATALINA_HOME}/bin/tomcat-juli.jar \
                              -user tomcat8.5 \
                              -java-home ${JAVA_HOME} \
                              -pidfile /var/run/tomcat.pid \
                              -errfile ${CATALINA_HOME}/logs/catalina.out \
                              -outfile ${CATALINA_HOME}/logs/catalina.out \
                              $CATALINA_OPTS \
                              org.apache.catalina.startup.Bootstrap

[Install]
WantedBy=multi-user.target
```

b. If the Tomcat service doesn't automatically start after reboot and you receive following error, on executing `sudo systemctl enable tomcat8.5.service`:

```
update-rc.d: error: tomcatx.x Default-Start contains no runlevels,
aborting.
```

Then the following step is required:

Remove the `tomcat8.5` file located at `/etc/init.d` and rerun following command:

```
sudo systemctl enable tomcat8.5.service
```

c. If you receive the following error:

```
insserv: warning: script 'tomcat8.5' missing LSB tags and override
```

Add the following to `/etc/systemd/system/tomcat8.5.service`:

```
#!/bin/sh

### BEGIN INIT INFO
# Provides: tomcat8.5
# Required-Start: $local_fs $network
# Required-Stop: $local_fs
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: tomcat8.5
# Description: tomcat8 service
### END INIT INFO
```

Run

```
sudo service tomcat8.5 start
```

41. If you are installing the ThingWorx Platform for the first time, the Java option `-Duser.timezone=UTC` should be added to the ExecStart block above, immediately following the line that begins with **`-Djava.library.path`**. The `UTC` timezone does not recognize daylight savings time. Setting this option prevents overwriting data when daylight savings time changes occur.

Caution

Existing customers should NOT update this setting at this time.

42. Create a new file in the tomcat /bin directory named `setenv.sh`:

```
cd $CATALINA_HOME/bin
sudo touch setenv.sh
sudo vi setenv.sh
CATALINA_OPTS="$CATALINA_OPTS -Djava.library.path=/usr/share/tomcat8.5/8.5.xx/
webapps/Thingworx/WEB-INF/extensions"
```

43. In the location of the Tomcat installation, open `CATALINA_HOME/conf/web.xml`. Replace the default error page (default is `stacktrace`) by adding the following into the `web.xml` file. Place the following within the `web-app` tag (after the `welcome-file-list` tag). A well-configured web application will override this default in `CATALINA_HOME/webapps/APP_NAME/WEB-INF/web.xml` so it won't cause problems.

```
<error-page><exception-type>java.lang.Throwable</exception-type><location>/
error.jsp</location></error-page>
```

44. In the location of the Tomcat installation, open `CATALINA_HOME/conf/server.xml`. Add the following inside the `<Host> </Host>` tags:

```
<Valve className="org.apache.catalina.valves.ErrorReportValve" showReport=
"false" showServerInfo="false" />
```

45. Remove all the Tomcat webapps located in `/<path_to_tomcat>/webapps/`. Removing these apps prevents unnecessary access to Tomcat, specifically in the context that would allow users to view other users' cookies.
46. PTC strongly recommends the use of TLS when running ThingWorx. For detailed instructions on setting up TLS, refer to this technical support article.
47. If your application requires a specific cipher suite, refer to the following documentation for configuration information:
    - https://www.jamf.com/jamf-nation/articles/384/configuring-supported-ciphers-for-tomcat-https-connections
48. (OPTIONAL STEP) To increase the default cache settings that affect static file caching, add the following line within the `<context></context>` tags in the `$CATALINA_HOME/conf/context.xml` file:

    ```
    <Resources cacheMaxSize="501200" cacheObjectMaxSize="2048" cacheTtl="60000"/>
    ```

    Increasing this setting improves performance and avoids the following message in Tomcat:

    ```
    WARNING: Unable to add the resource at [/Common/jquery/jquery-ui.js] to the
    cache because there was insufficient free space available after evicting
    expired cache entries - consider increasing the maximum size of the cache
    ```
49. H2 and Azure SQL: Go to Install ThingWorx on page 47.
50. PostgreSQL: Go to Install and Configure PostgreSQL on page 59.

Install ThingWorx (Ubuntu/RHEL)

1. Create `/ThingworxStorage` and `/ThingworxBackupStorage` directories. If you haven't already done so, create the `/ThingworxPlatform` directory as well:

   ```
   $ sudo mkdir /ThingworxStorage /ThingworxBackupStorage /ThingworxPlatform
   ```
2. Change owner and access permissions of `/ThingworxPlatform`, `/ThingworxStorage` and `/ThingworxBackupStorage`. Without these permissions, the server will fail to start.

   ```
   $ sudo chown tomcat8.5:tomcat8.5 /ThingworxStorage /ThingworxBackupStorage
   /ThingworxPlatform
   $ sudo chmod 775 /ThingworxStorage /ThingworxBackupStorage /ThingworxPlatform
   ```
3. If you have not already done so, obtain the `Thingworx.war` file from PTC Software Downloads.
4. Move the `Thingworx.war` to `$CATALINA_HOME/webapps`.

   ```
   $ sudo mv Thingworx.war $CATALINA_HOME/webapps
   $ sudo chown tomcat8.5:tomcat8.5 $CATALINA_HOME/webapps/Thingworx.war
   $ sudo chmod 775 $CATALINA_HOME/webapps/Thingworx.war
   ```
5. Place the `platform-settings.json` in the `ThingworxPlatform` folder.
6. Perform this step if you are using H2 as a database. If you are not using H2 as a database, go to the next step. Add a username and password for H2 in the

`platform-settings.json` file. See platform-settings.json Configuration Details on page 121 for more information.

Note

ThingWorx connections to the H2 database require a username and password defined by the user, or the server will not start. This design fully mitigates any potential vulnerability represented by CVE-2018-10054.

```
},
"PersistenceProviderPackageConfigs":{
  "H2PersistenceProviderPackage":{
      "ConnectionInformation":
{
    "password": "<addsecurepassword>",
     "username": "twadmin"
}
},
```

7. Perform this step if you are using Azure SQL as a database. If you are not using Azure SQL as a database, go to the next step. Open the `platform-settings.json` file and add the Azure SQL persistence provider parameters:

```
"PersistenceProviderPackageConfigs": {
"AzuresqlPersistenceProviderPackage": {
"ConnectionInformation": {
    "driverClass": "com.microsoft.sqlserver.jdbc.SQLServerDriver",
"jdbcUrl": "jdbc:sqlserver://<server name>:<port>;databaseName=thingworx;
applicationName=Thingworx;",
"password": "<database password>",
"username": "twadmin"
}
}
}
```

8. Configure the Administrator password. Add the following **AdministratorUserSettings** section (in **PlatformSettingsConfig**) to your `platform-settings.json` file along with a password that is at least 14 characters long. Reference platform-settings.json Configuration Details on page 121 for more information on placement. See Passwords for additional information on setting passwords. Do not copy and paste the sample below, as it may cause bad formatting in your `platform-settings.json`. Instead, click here and copy from the file.

```
{
    "PlatformSettingsConfig": {
        "AdministratorUserSettings": {
            "InitialPassword": "changeme"
        }
    }
}
```

Note

If Tomcat fails to start and reports the error message: `Check the InitialPassword setting in the AdministratorUserPassword section in platform-settings.json...`, check the following:

- The password setting exists in `platform-settings.json`
- The password is valid (14 or more characters by default)
- The `platform-settings.json` file is formatted correctly - bad formatting could lead to errors

:

9. Enable extension import. By default, extension import is disabled for all users. Add the following to the `platform-settings.json` file. Update the following *ExtensionPackageImportPolicy* parameters to `true` to allow extensions to be imported. See Importing Extensions for best practices on configuration.

```
"ExtensionPackageImportPolicy": {
             "importEnabled": <true or false>,
             "allowJarResources": <true or false>,
             "allowJavascriptResources": <true or false>,
             "allowCSSResources": <true or false>,
             "allowJSONResources": <true or false>,
             "allowWebAppResources": <true or false>,
             "allowEntities": <true or false>,
             "allowExtensibleEntities": <true or false>
        },
```

10. Configure licensing:

- Open the `platform-settings.json` file and add the following to the `PlatformSettingsConfig` section (reference platform-settings.json Configuration Options on page 121 for more information on placement.)

Note

If you are performing a disconnected installation (no internet access), you do not need to add the licensing information to the `platform-settings.json` file. Refer to the Licensing Guide for disconnected sites and skip this step.

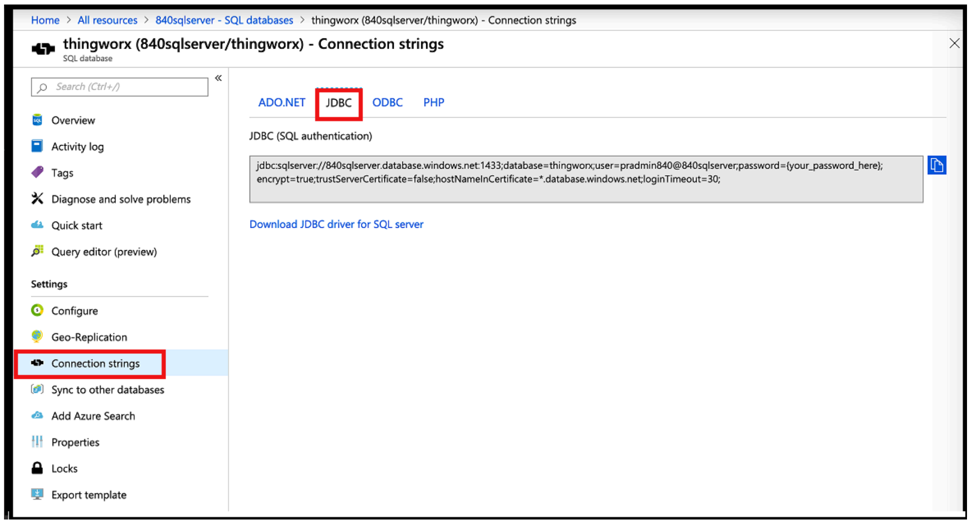
```
"LicensingConnectionSettings":{
      "username":"PTC Support site user name",
"password":"PTC Support site password"
}
```

Note

If the settings are filled out incorrectly or if the server can't connect, a License Request text file (`licenseRequestFile.txt`) is created in the `ThingworxPlatform` folder. In this scenario, a license must be created manually. (If it is not created, ThingWorx will start in limited mode. Limited mode does not allow you to persist licensed entities to the database. Licensed entities are Things, Mashups, Masters, Gadgets, Users, and Persistence Providers).

More information on obtaining a ThingWorx disconnected site license through our License Management site can be found in the Licensing Guide for disconnected sites (no connection to PTC Support portal).

11. Encrypt the license server password by following the steps in Encrypting Passwords on page 118.
12. If you are using Azure SQL as your database, follow these steps to download the JDBC driver. Skip this step if you are not using Azure SQL.
    a. Go to the Azure portal and navigate to your ThingWorx database.
    b. Select **Connection strings**.
    c. Select the **JDBC** tab.


Home > All resources > 840sqlserver - SQL databases > thingworx (840sqlserver/thingworx) - Connection strings
thingworx (840sqlserver/thingworx) - Connection strings
SQL database
Search (Ctrl+/)
Overview
Activity log
Tags
Diagnose and solve problems
Quick start
Query editor (preview)
Settings
Configure
Geo-Replication
Connection strings
Sync to other databases
Add Azure Search
Properties
Locks
Export template
ADO.NET
JDBC
ODBC
PHP
Download JDBC driver for SQL server

JDBC (SQL authentication)

```
jdbc:sqlserver://840sqlserver.database.windows.net:1433;database=thingworx;user=pradmin840@840sqlserver;password={your_password_here};
encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;
```

    d. Select **Download Microsoft JDBC Driver for SQL Server**.
    e. Select **Microsoft JDBC Driver 6.0 for SQL Server**.

f. Extract and copy the downloaded binary in your ThingWorx VM to your Tomcat `lib` directory.

13. Start Tomcat.

```
(UBUNTU) sudo service tomcat8.5 start

(RHEL) $ sudo systemctl start tomcat
```

Verify that a license file (`successful_license_capability_response.bin`) is created in the `ThingworxPlatform` folder.

14. To launch ThingWorx, go to `http://<servername>:<port>/Thingworx` in a Web browser.

15. Change the initial Administrator password:

a. In Composer, select **Administrator** > **Change Password**.

b. In the **Change Password** window, enter **Current Password**, **New Password**, and **Confirm Password**.

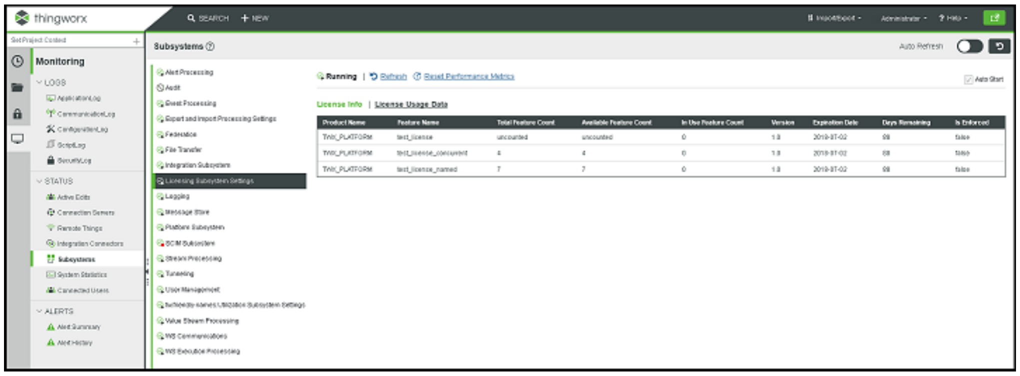
Note

The password, which should not be easily guessed or a known, common password, is recommended to be at least 14 characters in length and should include a mix of uppercase and lowercase letters, numbers, and special characters.

c. Delete the initial password from the `platform-settings.json` file.

16. Select **Done**.

17. (OPTIONAL STEP) To determine the status of your license, open the **Monitoring**>**Subsystem**>**Licensing Subsystem Settings** in Composer to confirm the list of features (licensed entities) included with the license. If there are no licensed entities present, you are in limited mode.



Product Name	Feature Name	Total Feature Count	Available Feature Count	In Use Feature Count	Version	Expiration Date	Days Remaining	Is Enforced
TWX_PLATFORM	test_license	uncounted	uncounted	0	1.0	2019-07-02	98	false
TWX_PLATFORM	test_license_concurrent	4	4	0	1.0	2019-07-02	98	false
TWX_PLATFORM	test_license_named	7	7	0	1.0	2019-07-02	98	false

PostgreSQL

Install Java and Apache Tomcat (Ubuntu)

1. If you are using AzureSQL for your database, go to Using Azure SQL Server as the Persistence Provider. Perform the steps in that section to set up the database, and you will be referred back to this section.
2. If you are using MSSQL for your database, go to Using MSSQL as the Persistence Provider. Perform the steps in that section to set up the database, and you will be referred back to this section.
3. Update Ubuntu packages:
   ```
   $ sudo apt-get update
   ```
4. Install and Configure Network Time Protocol (NTP) settings for time synchronization:
   ```
   $ sudo apt-get install ntp
   ```

Note

The default configuration for NTP is sufficient. For additional configuration information about NTP (beyond the scope of this documentation), refer to the following resources:

- Time Synchronization with NTP
- How do I use pool.ntp.org?

5. Edit AUTHBIND properties to allow Tomcat to bind to ports below 1024:
   ```
   $ sudo apt-get install authbind
   ```
6. Download the Java JDK tar file from Oracle's website, or run the following
   ```
   wget -c --header "Cookie: oraclelicense=accept-securebackup-cookie" http://
   download.oracle.com/otn-pub/java/jdk/8u131-b11/
   d54c1d3a095b4ff2b6607d096fa80163/jdk-8u131-linux-x64.tar.gz
   ```
7. Extract tar file:
   ```
   $ tar -xf jdk-8uxxx-linux-x64.tar.gz
   ```
8. Create the directory by moving the JDK to `/usr/lib/jvm`:

Note

If the directory is not empty, a warning message will display.

```
$ sudo mkdir -p /usr/lib/jvm
$ sudo mv jdk1.8.0_xxx/ /usr/lib/jvm/
```

9. Add alternatives to the system:

```
$ sudo update-alternatives --install "/usr/bin/java" "java" "/usr/lib/jvm/
jdk1.8.0_xxx/bin/java" 1
$ sudo update-alternatives --install "/usr/bin/keytool" "keytool" "/usr/lib/
jvm/jdk1.8.0_xxx/bin/keytool" 1
```

10. Change access permissions:

```
$ sudo chmod a+x /usr/bin/java
$ sudo chmod a+x /usr/bin/keytool
```

11. Change owner:

```
$ sudo chown -R root:root /usr/lib/jvm/jdk1.8.0_xxx/
```

12. Configure master links:

```
$ sudo update-alternatives --config java
$ sudo update-alternatives --config keytool
```

Note

`Nothing to configure` is a normal response to this command and is not an error. Additional executables in `/usr/lib/jvm/jdk1.8.0_xxx/bin/` can be installed using the previous set of steps.

13. Verify Java version:

```
$ java -version
```

This should return something similar to the following (build specifics may be different):

```
java version "1.8.0_xxx"
Java(TM) SE Runtime Environment (build 1.8.0_xxx-bxx)
Java HotSpot(TM) 64-Bit Server VM (build xx.xx-bxx, mixed mode)
```

14. Download Apache Tomcat: The steps in this process use Tomcat 8.5.*xx*, where *xx* is replaced with the version you are using.

```
$ wget http://archive.apache.org/dist/tomcat/tomcat-8/v8.5.xx/bin/apache-
tomcat-8.5.xx.tar.gz
```

Note

Best practice includes verifying the integrity of the Tomcat file by using the signatures or checksums for each release. Refer to Apache's documentation for more information.

15. Extract tar file:

```
$ tar -xf apache-tomcat-8.5.xx.tar.gz
```

16. Create and change the owner for `/usr/share/tomcat8.5` and move Tomcat to the following location. Add user and group to the system:

```
$ sudo mkdir -p /usr/share/tomcat8.5
$ sudo mv apache-tomcat-8.5.xx /usr/share/tomcat8.5/8.5.xx
$ sudo addgroup --system tomcat8.5 --quiet -force-badname
$ sudo adduser --system --home /usr/share/tomcat8.5/ --no-create-home --ingroup
tomcat8.5 --disabled-password --force-badname --shell /bin/false tomcat8.5
$ sudo chown -R tomcat8.5:tomcat8.5 /usr/share/tomcat8.5
```

17. Define environment variables in `/etc/environment`:

```
$ export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_xxx
$ export CATALINA_HOME=/usr/share/tomcat8.5/8.5.xx
```

18. Change directory to `$CATALINA_HOME`:

```
$ cd $CATALINA_HOME
```

19. Change owner and access permissions of `bin/`, `lib/`, and `webapps/`:

```
$ sudo chown -Rh tomcat8.5:tomcat8.5 bin/ lib/ webapps/
$ sudo chmod 775 bin/ lib/ webapps/
```

20. Change owner and access permissions of `usr/share/tomcat8.5/8.5xx`:

```
sudo chown -R tomcat8.5:tomcat8.5 /usr/share/tomcat8.5/8.5.xx
sudo chmod -R 775 /usr/share/tomcat8.5/8.5.xx
```

21. Change owner and access permissions of `conf/`:

```
$ sudo chown -Rh root:tomcat8.5 conf/
$ sudo chmod -R 650 conf/
```

22. Change access permissions of `logs/`, `temp/`, and `work/`:

```
$ sudo chown -R tomcat8.5:adm logs/ temp/ work/
$ sudo chmod 760 logs/ temp/ work/
```

23. Create self-signed certificate:

```
$ sudo $JAVA_HOME/bin/keytool -genkey -alias tomcat8.5 -keyalg RSA -keystore
$CATALINA_HOME/conf/.keystore
```

24. Follow the instructions to complete the certificate creation process.
    - Set the keystore password.
    - Follow the prompts to set up your security certificate.
    - Set the tomcat8.5 user password to the same as the keystore password:

    ```
    $ sudo chown root:tomcat8.5 $CATALINA_HOME/conf/.keystore
    $ sudo chmod 640 $CATALINA_HOME/conf/.keystore
    ```

25. Uncomment the `Manager` element in `$CATALINA_HOME/conf/context.xml` to prevent sessions from persisting across restarts:

```
<Manager pathname="" />
```

Note

For security reasons, it is critical that you disable the AJP connector, if not already done so by default, by performing the following step.

26. In the location of the Tomcat installation, open `conf/server.xml` and search for the following line. If found, comment it out and save the file:

    ```
    <Connector port ="8009" protocol="AJP/1.3" redirectPort="8443"/>
    ```

    Note

    In Apache Tomcat 9.0 and later, the **rejectIllegalHeader** attribute defaults to true. Manually modifying the `conf/web.xml` file to set this attribute to false is not recommended or supported by PTC.

    If you receive an error that the directory doesn’t exist, use the following commands to ensure port 443 works:

    ```
    sudo touch /etc/authbind/byport/443
    sudo chmod 700 /etc/authbind/byport/443
    sudo chown tomcat8.5:tomcat8.5 /etc/authbind/byport/443
    ```

27. Define a user in `$CATALINA_HOME/conf/tomcat-users.xml`:

    ```
    sudo vi $CATALINA_HOME/conf/tomcat-users.xml
    <user username="<Tomcat user name> " password="<Tomcat password> " roles=
    "manager"/>
    ```

28. Determine uid of tomcat8.5 user:

    ```
    $ id -u tomcat8.5
    ```

29. Using this number, create an ID file in `/etc/authbind/byuid/`:

    Note

    Change the `<uid>` to the number that was returned in the previous step.

    ```
    $ sudo touch /etc/authbind/byuid/<uid>
    sudo vi /etc/authbind/byuid/<uid>
    ```

30. Edit the file from the step above and paste in the following:

    ```
    0.0.0.0/0:1,1023
    ```

31. Change owner and access permissions of `/etc/authbind/byuid/<uid>`:

    ```
    $ sudo chown tomcat8.5:tomcat8.5 /etc/authbind/byuid/<uid>
    $ sudo chmod 700 /etc/authbind/byuid/<uid>
    ```

32. Modify `$CATALINA_HOME/bin/startup.sh` to always use authbind:

    ```
    sudo vi $CATALINA_HOME/bin/startup.sh
    ```

    Comment the following in the file:

    ```
    #exec "$PRGDIR"/"$EXECUTABLE" start "$@"
    ```

33. Add the following to the end of the file:

    ```
    exec authbind --deep "$PRGDIR"/"$EXECUTABLE" start "$@"
    ```

34. In `/etc/init.d`, create tomcat8.5 file:

```
$ sudo touch /etc/init.d/tomcat8.5
```

35. Edit the file and enter the following contents:

```
$ sudo vi /etc/init.d/tomcat8.5

CATALINA_HOME=/usr/share/tomcat8.5/8.5.xx

case $1 in
  start)
    /bin/su -p -s /bin/sh tomcat8.5 $CATALINA_HOME/bin/startup.sh
  ;;

  stop)
    /bin/su -p -s /bin/sh tomcat8.5 $CATALINA_HOME/bin/shutdown.sh
  ;;

  restart)
    /bin/su -p -s /bin/sh tomcat8.5 $CATALINA_HOME/bin/shutdown.sh
    /bin/su -p -s /bin/sh tomcat8.5 $CATALINA_HOME/bin/startup.sh
  ;;

esac
exit 0
```

36. Change access permissions of `etc/init.d/tomcat8.5` and create symbolic links:

```
$ sudo chmod 755 /etc/init.d/tomcat8.5
$ sudo ln -s /etc/init.d/tomcat8.5 /etc/rc1.d/K99tomcat
$ sudo ln -s /etc/init.d/tomcat8.5 /etc/rc2.d/S99tomcat
```

37. Set up Tomcat as a service to start on boot. Build JSVC if it is not already installed on your system. If it is already installed, skip and go to the next step:

```
$ sudo apt-get install gcc
```

38. Set up the Tomcat service on boot:

```
$ cd /usr/share/tomcat8.5/8.5.xx/bin/
$ sudo tar xvfz commons-daemon-native.tar.gz
$ cd commons-daemon-*-native-src/unix
$ sudo ./configure --with-java=$JAVA_HOME
$ sudo apt-get install make
$ sudo make
$ sudo cp jsvc ../..
```

39. Create the Tomcat service file:

```
sudo touch /etc/systemd/system/tomcat8.5.service
```

40. Open `/etc/systemd/system/tomcat8.5.service` in a text editor (as root):

```
sudo vi /etc/systemd/system/tomcat8.5.service
```

   a. Paste the following in the Tomcat service file:

Note

In the example below, set values for **-Xms** and **-Xmx** to 75% of the available OS memory (for example, 12GB for a 16GB RAM system). Refer to JVM Tuning for additional information.

```
[Unit]
Description=Apache Tomcat Web Application Container
After=network.target

[Service]
Type=forking
PIDFile=/var/run/tomcat.pid
Environment=CATALINA_PID=/var/run/tomcat.pid
Environment=JAVA_HOME=/usr/lib/jvm/jdk1.8.0_xxx
Environment=CATALINA_HOME=/usr/share/tomcat8.5/8.5.xx
Environment=CATALINA_BASE=/usr/share/tomcat8.5/8.5.xx
Environment=CATALINA_OPTS=

ExecStart=/usr/share/tomcat8.5/8.5.xx/bin/jsvc \
                              -Dcatalina.home=${CATALINA_HOME} \
                              -Dcatalina.base=${CATALINA_BASE} \
                              -Djava.awt.headless=true -Djava.net.
preferIPv4Stack=true -Dserver -Dd64 -XX:+UseNUMA \
                              -XX:+UseG1GC -Dfile.encoding=UTF-8 \
                              -Djava.library.path=${CATALINA_BASE}/webapps/
Thingworx/WEB-INF/extensions \
                              -Xms=<75% of available OS memory> \
                              -Xmx=<75% of available OS memory> \
                              -cp ${CATALINA_HOME}/bin/commons-daemon.jar:
${CATALINA_HOME}/bin/bootstrap.jar:${CATALINA_HOME}/bin/tomcat-juli.jar \
                              -user tomcat8.5 \
                              -java-home ${JAVA_HOME} \
                              -pidfile /var/run/tomcat.pid \
                              -errfile ${CATALINA_HOME}/logs/catalina.out \
                              -outfile ${CATALINA_HOME}/logs/catalina.out \
                              $CATALINA_OPTS \
                              org.apache.catalina.startup.Bootstrap

[Install]
WantedBy=multi-user.target
```

b. If the Tomcat service doesn't automatically start after reboot and you receive following error, on executing `sudo systemctl enable tomcat8.5.service`:

```
update-rc.d: error: tomcatx.x Default-Start contains no runlevels,
aborting.
```

Then the following step is required:

Remove the `tomcat8.5` file located at `/etc/init.d` and rerun following command:

```
sudo systemctl enable tomcat8.5.service
```

41. Create a new file in the tomcat `/bin` file named `setenv.sh`:

```
cd $CATALINA_HOME/bin
sudo touch setenv.sh
sudo vi setenv.sh
CATALINA_OPTS="$CATALINA_OPTS -Djava.library.path=/usr/share/tomcat8.5/8.5.xx/
webapps/Thingworx/WEB-INF/extensions"
```

42. In the location of the Tomcat installation, open `CATALINA_HOME/conf/web.xml`. Replace the default error page (default is stacktrace) by adding the following into the `web.xml` file. Place the following within the `web-app` tag (after the `welcome-file-list` tag). A well-configured web application will override this default in `CATALINA_HOME/webapps/APP_NAME/WEB-INF/web.xml` so it won't cause problems.

```
<error-page><exception-type>java.lang.Throwable</exception-type><location>/
error.jsp</location></error-page>
```

43. In the location of the Tomcat installation, open `CATALINA_HOME/conf/server.xml`. Add the following inside the `<Host> </Host>` tags:

```
<Valve className="org.apache.catalina.valves.ErrorReportValve" showReport=
"false" showServerInfo="false" />
```

44. Remove all the Tomcat webapps located in `/<path_to_tomcat>/webapps/`. Removing these apps prevents unnecessary access to Tomcat, specifically in the context that would allow users to view other users' cookies.
45. PTC strongly recommends the use of TLS when running ThingWorx. For detailed instructions on setting up TLS, refer to this technical support article.
46. If your application requires a specific cipher suite, refer to the following documentation for configuration information:
    - https://www.jamf.com/jamf-nation/articles/384/configuring-supported-ciphers-for-tomcat-https-connections
47. (OPTIONAL STEP) To increase the default cache settings that affect static file caching, add the following line within the `<context></context>` tags in the `$CATALINA_HOME/conf/context.xml` file:

```
<Resources cacheMaxSize="501200" cacheObjectMaxSize="2048" cacheTtl="60000"/>
```

Increasing this setting improves performance and avoids the following message in Tomcat:

```
WARNING: Unable to add the resource at [/Common/jquery/jquery-ui.js] to the
cache because there was insufficient free space available after evicting
expired cache entries - consider increasing the maximum size of the cache
```

48. H2 and Azure SQL: Go to Install ThingWorx on page 47.

49. PostgreSQL: Go to Install and Configure PostgreSQL on page 59.

Install and Configure PostgreSQL (Ubuntu)

The instructions provided below are intended for the PostgreSQL administrator (not the DB host servers). If you are including the HA layer to your implementation, refer to the the ThingWorx High Availability Administrator's Guide.

Install PostgreSQL and Create a New User Role

1. Refer to the ThingWorx System Requirements for information on supported PostgreSQL versions.

   Note

   The steps in this procedure use PostgreSQL version *x.x*, where *x.x* is the supported version.

2. Download and install the appropriate version of PostgreSQL.

   - The PostgreSQL repository can be added allowing the application to be installed directly from the package manager.

     Note

     To get the Ubuntu version name use the following command:

     ```
     $ lsb_release -sc
     ```

     ```
     $ sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ <YOUR_
     UBUNTU_VERSION_HERE>-pgdg main" '> /etc/apt/sources.list.d/pgdg.list
     $ sudo wget -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo
     apt-key add -

     $ sudo apt-get update

     $ sudo apt-get install postgresql-x.x -y
     ```

3. Install PgAdmin, the PostgreSQL admin tool:

   ```
   $ sudo apt-get install pgadmin4 -y
   ```

Note

To install PgAdmin via the command line, reference https://wiki.postgresql.org/wiki/Manual_Setup_at_the_Command_Line.

4. Set up the password for the PostgreSQL user:

```
$ sudo service postgresql restart
$ sudo -u postgres psql -c "ALTER ROLE postgres WITH password '<unique
PostgreSQL password>'"
```

5. Enter the password for the PostgreSQL user. You will use this password in later steps.

Note

The password, which should not be easily guessed or a known, common password, should be at least 14 characters in length and include a mix of uppercase and lowercase letters, numbers, and special characters.

6. Configure PgAdmin:

```
$ sudo pgadmin4
```

- In the PgAdmin GUI, click on **file->Open postgresql.conf**
- Open `/etc/postgresql/x.x/main/postgresql.conf`
- Put a check next to **listen addresses** and **port**. The default settings of **localhost** and **5432** are usually sufficient.
- Save and close.
- Click on **file->Open pg_hba.conf**
- Open `/etc/postgresql/x.x/main/pg_hba.conf`
- Double-click on the database ‘all’ line with address 127.0.0.1/32
- Set Method to **md5**
- Click **OK**
- Save and exit
- Close PgAdmin.

7. Restart the PostgreSQL service:

```
$ sudo service postgresql restart
```

8. Set up PgAdmin to connect to the database:

```
$ sudo pgadmin4
```

9. Click the plug icon to add a connection to a server in the top left corner and fill out the following:

```
Name: PostgreSQL x.x
```

```
Host: localhost
Port: 5432
Service: <blank>
Maintenance DB: postgres
Username: postgres
Password: <unique PostgreSQL password as set previously>
Store password: Checked
Group: Servers
```

10. Click **OK**.
11. Create a new user role:
    a.

    Note

    The following command can be used if you are not using PgAdmin:

    ```
    sudo -u postgres psql -c "CREATE USER twadmin WITH PASSWORD '<unique
    postgres password>';"
    ```

    b. Right click **PostgreSQLx.x (<IP or host name of the database>:<Port number of PostgreSQL>)**. Example: `PostgreSQLx.x (localhost:5432).`
    c. Select **NewObject**>**NewLogin Role**. On the **Properties** tab, enter a name in the **Role** name field.
    d. On the **Definition** tab, in the **Password** field, enter a unique password (you will be prompted to enter it twice). You will need to re-enter this password in later steps.

    Note

    The password, which should not be easily guessed or a known, common password, should be at least 14 characters in length and include a mix of uppercase and lowercase letters, numbers, and special characters.

    e. Click **OK**.

Configure PostgreSQL Database Located on a Separate Server than ThingWorx

Note

This section is optional for development environments, but should be implemented in all production environments.

By default, the PostgreSQL server is installed in a locked-down state. The server will only listen for connections from the local machine In order to get ThingWorx to communicate to the PostgreSQL server, some configuration changes need to be made so that PostgreSQL knows to listen for connections from other users (thingworx user, default is twadmin) and/or other machines (ThingWorx installed on a separate server).

You will need to know where your PostgreSQL data directory resides for these steps. On Linux, the location of the data folder, or even the configuration files can change based on distribution and installation method (download or package manager install). This location will be referred to as `<PGDATA>` in these instructions.

Note

On Ubuntu, when installed via apt-get, the configuration files are located at `/etc/postgresql/x.x/main/`

Modify the `pg_hba.conf` file and add the following lines based on your desired configuration:

If you want to allow all IPv4 addresses to connect:	`hostallall0.0.0.0/0md5`
If you want to allow only a specific IPv4 address to connect (Replace <*ipAddress*> with the IP address of the machine making the connection):	`hostallall<ipAddress>/32md5`
If you want to allow all IPv6 addresses to connect:	`hostallall::0/0md5`
If you want to allow only a specific IPv6 address to connect (Replace <ipv6Address> with the appropriate address):	`hostallall<ipv6Address>/128md5`

Any other combination is possible by using additional allowance lines (individual IPs or ranges) or subnet masks appropriate to the machines that require access to the PostgreSQL database.

Any change to this file requires a restart of the database service.

Note

For additional information about configuring the `pg_hba.conf` file, see the PostgreSQL documentation.

Enabling PostgreSQL to listen for all Connections

On Linux installations of PostgreSQL, there is an additional configuration step required to configure the PostgreSQL server to listen for connections.

1. In the `postgresql.conf` file, uncomment and update the `listen_addresses` line:

   ```
   Uncomment the listen_addresses line and change localhost to '*'
   # Listen on all addresses. Requires restart.
   listen_addresses = '*'
   ```

2. Restart the PostgreSQL server.

Configure and Execute the PostgreSQL Database Script

To set up the PostgreSQL database and tablespace, the `thingworxPostgresDBSetup` script must be configured and executed.

1. Create the `ThingworxPostgresqlStorage` folder on the drive that the `ThingworxStorage` folder is located (in the root directory by default). Note the following:

   - If you create the folder using the `-d<databasename>` command, you do not have to use the PostgreSQL user.
   - You must specify the `-l` option to a path that exists. For example, `-l D:\ThingworxPostgresqlStorage`. The script does not create the folder for you.
   - The folder must have appropriate ownership and access rights. It should be owned by the same user who runs the PostgreSQL service, and have Full Control assigned to that user - this user is generally `NETWORK_SERVICE`, but may differ in your environment.

   ```
   $ sudo mkdir /ThingworxPostgresqlStorage
   $ sudo chown postgres:postgres /ThingworxPostgresqlStorage
   $ sudo chmod 755 /ThingworxPostgresqlStorage
   ```

2. Obtain the `thingworxPostgresDBSetup` script from the ThingWorx software download package. The script is located in the `install` folder. ThingWorx downloads are available in PTC Software Downloads.
3. If necessary, configure the script. Reference the options in the table below.

Note

This example uses the 8.x.x download from the PTC site. If necessary, change the file name to the version you are using.

```
$ sudo unzip MED-61111-CD-084_ThingWorx-Platform-Postgres-8-x-x.zip
$ cd install
```

4. To set up the database and tablespace with a default PostgreSQL installation that has a PostgreSQL database and a PostgreSQL user name, enter:

```
$ sudo sh thingworxPostgresDBSetup.sh -a postgres -u <user role name> -l
/ThingworxPostgresqlStorage
```

thingworxPostgresDBSetup Script Options

Option	Parameter	Default	Description	Example
t or -T	tablespace	thingworx	Tablespace name	`-t thingworx`
-p or -P	port	5432	Port number of PostgreSQL	`-p 5432`
-d or -D	database	thingworx	PostgreSQL Database name to create	`-d thingworx`
-h or -H	host	localhost	Name of the host	`-h localhost`
-l or -L	tablespace_ location	/Thingworx-Postgresql-Storage	Required. Location in the file system where the files representing database objects are stored.	`-l or -L`

thingworxPostgresDBSetup Script Options (continued)

Option	Parameter	Default	Description	Example
-a or -A	adminuser-name	postgres	Administrator Name	`-a postgres`
-u or -U	thingworxu-sername	twadmin	User name that has permissions to write to the database.	`-u twadmin`

5. Execute the script.

Configure and Execute the Model/Data Provider Schema Script

To set up the PostgreSQL model/data provider schema, the `thingworxPostgresSchemaSetup` script must be configured and executed. This will set up the public schema under your database on the PostgreSQL instance installed on the localhost.

1. Obtain and open the `thingworxPostgresSchemaSetup` file from the ThingWorx software download package. The script is located in the `install` folder.
2. If necessary, configure the script. Reference the options in the table below.

thingworxPostgresSchemaSetup Script Options

Option	Parameter	Default	Description	Example
-h or -H	host	localhost	IP or host name of the database.	`-h localhost`
-p or -P	port	5432	Port number of PostgreSQL.	`-p 5432`
-d or -D	database	thingworx	Database name to use.	`-d thingworx`
-s or -S	schema	public	Schema name to use.	`-s mySchema`

thingworxPostgresSchemaSetup Script Options (continued)

Option	Parameter	Default	Description	Example
-u or -U	username	twadmin	Username to update the database schema	`-u twadmin`
-o or -O	option	all	There are three options: • all: Sets up the model and data provider schemas into the specified database. • model: Sets up the model provider schema into the specified database. • data: Sets up the data provider schema into the specified database.	`-o data`

3. Execute the script. The script can be run with the default parameters as:
   ```
   $ sudo sh thingworxPostgresSchemaSetup.sh
   ```
   The username should match the PostgreSQL username that was previously created.

Configure platform-settings.json

1. Create the folder `ThingworxPlatform` at the root of the drive where Tomcat was installed or as a system variable. Note the following:

- To specify the location where ThingWorx stores its settings, you can set the `THINGWORX_PLATFORM_SETTINGS` environment variable to the desired location. Ensure that the folder referenced by `THINGWORX_PLATFORM_SETTINGS` exists and is writable by the Tomcat user. This environment variable should be configured as part of the system environment variables. Ubuntu example: `THINGWORX_PLATFORM_SETTINGS=/data/ThingworxPlatform`.
- The ThingWorx server will fail to start if it does not have read and write access to this folder.

```
$ sudo mkdir /ThingworxPlatform
```

2. Place the `platform-settings.json` file into the `ThingworxPlatform` folder. This file is available in the ThingWorx software download.

```
$ sudo cp platform-settings.json /ThingworxPlatform/
```

3. Open `platform-settings.json` and configure as necessary. Refer to the configuration options in platform-settings.json Configuration Details on page 121.

Note

If your PostgreSQL server is not the same as your ThingWorx server, and you are having issues with your ThingWorx installation, review your Tomcat logs and `platform-settings.json` file. The default installation assumes both servers are on the same machine. While it is optional to have the PostgreSQL database on the same server as ThingWorx in a development environment, it should be separate in all production environments.

Encrypt the PostgreSQL Password

- Encrypt the password by following the steps in Encrypting Passwords on page 118

(Optional) Installing the PostgreSQL Client Package and PostgreSQL User

In order to issue PostgreSQL commands from the client machine to the PostgreSQL server, do so from a PostgreSQL user. The `postgresql-client-x.x` package can be installed on the client machine, refer to your distributions documentation on how to install it. This package provides some administration tools such as `psql`.

Install ThingWorx

Go to Install ThingWorx on page 68.

Install ThingWorx (Ubuntu/RHEL)

1. Create `/ThingworxStorage` and `/ThingworxBackupStorage` directories. If you haven't already done so, create the `/ThingworxPlatform` directory as well:
```
$ sudo mkdir /ThingworxStorage /ThingworxBackupStorage /ThingworxPlatform
```
2. Change owner and access permissions of `/ThingworxPlatform`, `/ThingworxStorage` and `/ThingworxBackupStorage`. Without these permissions, the server will fail to start.
```
$ sudo chown tomcat8.5:tomcat8.5 /ThingworxStorage /ThingworxBackupStorage
/ThingworxPlatform
$ sudo chmod 775 /ThingworxStorage /ThingworxBackupStorage /ThingworxPlatform
```
3. If you have not already done so, obtain the `Thingworx.war` file from PTC Software Downloads.
4. Move the `Thingworx.war` to `$CATALINA_HOME/webapps`.
```
$ sudo mv Thingworx.war $CATALINA_HOME/webapps
$ sudo chown tomcat8.5:tomcat8.5 $CATALINA_HOME/webapps/Thingworx.war
$ sudo chmod 775 $CATALINA_HOME/webapps/Thingworx.war
```
5. Place the `platform-settings.json` in the `ThingworxPlatform` folder.
6. Configure the Administrator password. Add the following **AdministratorUserSettings** section (in **PlatformSettingsConfig**) to your `platform-settings.json` file along with a password that is at least 14 characters long. Reference platform-settings.json Configuration Details on page 121 for more information on placement. See the section, Passwords, in the ThingWorx Help Center for additional information on setting passwords. Do not copy and paste the sample below, as it may cause bad formatting in your `platform-settings.json`. Instead, click here and copy from the file.

```
{
    "PlatformSettingsConfig": {
        "AdministratorUserSettings": {
            "InitialPassword": "changeme"
        }
    }
}
```

Note

If Tomcat fails to start and reports the error message: `Check the InitialPassword setting in the AdministratorUserPassword section in platform-settings.json...`, check the following:

- The password setting exists in `platform-settings.json`
- The password is valid (14 or more characters by default)
- The `platform-settings.json` file is formatted correctly - bad formatting could lead to errors

7. Enable extension import. By default, extension import is disabled for all users. Add the following to the `platform-settings.json` file. Update the following *ExtensionPackageImportPolicy* parameters to `true` to allow extensions to be imported. See here if you are viewing the PDF) for best practices on configuration.

```
"ExtensionPackageImportPolicy": {
             "importEnabled": <true or false>,
             "allowJarResources": <true or false>,
             "allowJavascriptResources": <true or false>,
             "allowCSSResources": <true or false>,
             "allowJSONResources": <true or false>,
             "allowWebAppResources": <true or false>,
             "allowEntities": <true or false>,
             "allowExtensibleEntities": <true or false>
        },
```

8. Configure licensing:

- Open the `platform-settings.json` file and add the following to the `PlatformSettingsConfig` section (reference platform-settings.json Configuration Options on page 121 for more information on placement.)

Note

If you are performing a disconnected installation (no internet access), you do not need to add to the `platform-settings.json` file. Refer to the Licensing Guide for disconnected sites and skip this step.

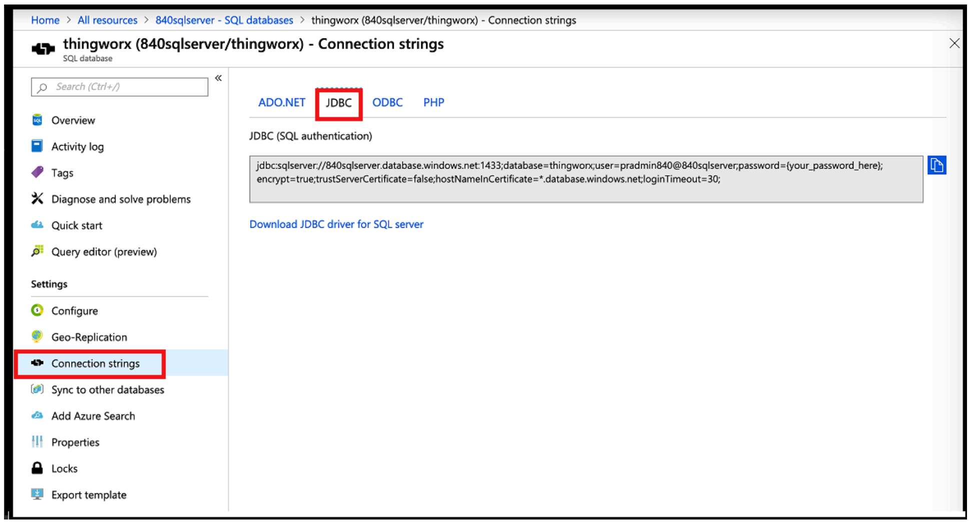
```
"LicensingConnectionSettings":{
      "username":"PTC Support site user name",
      "password":"PTC Support site password"
      }
```

Note

If the settings are filled out incorrectly or if the server can't connect, a License Request text file (`licenseRequestFile.txt`) is created in the `ThingworxPlatform` folder. In this scenario, a license must be created manually. (If it is not created, ThingWorx will start in limited mode. Limited mode does not allow you to persist licensed entities to the database. Licensed entities are Things, Mashups, Masters, Gadgets, Users, and Persistence Providers).

More information on obtaining a ThingWorx disconnected site license through our License Management site can be found in the Licensing Guide for disconnected sites (no connection to PTC Support portal).

9. Encrypt the license server password by following the steps in Encrypting Passwords on page 118.
10. If you are using Azure SQL as your database, follow these steps to download the JDBC driver. Skip this step if you are not using Azure SQL.
    a. Go to the Azure portal and navigate to your ThingWorx database.
    b. Select **Connection strings**.
    c. Select the **JDBC** tab.


Home > All resources > 840sqlserver - SQL databases > thingworx (840sqlserver/thingworx) - Connection strings
thingworx (840sqlserver/thingworx) - Connection strings
SQL database
Search (Ctrl+/)
Overview
Activity log
Tags
Diagnose and solve problems
Quick start
Query editor (preview)
Settings
Configure
Geo-Replication
Connection strings
Sync to other databases
Add Azure Search
Properties
Locks
Export template
ADO.NET
JDBC
ODBC
PHP
Download JDBC driver for SQL server

JDBC (SQL authentication)

```
jdbc:sqlserver://840sqlserver.database.windows.net:1433;database=thingworx;user=pradmin840@840sqlserver;password={your_password_here};
encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;
```

    d. Select **Download Microsoft JDBC Driver for SQL Server**.
    e. Select **Microsoft JDBC Driver 6.0 for SQL Server**.
    f. Extract and copy the downloaded binary in your ThingWorx VM to your Tomcat `lib` directory.
11. Start Tomcat.

```
(UBUNTU) sudo service tomcat8.5 start
```

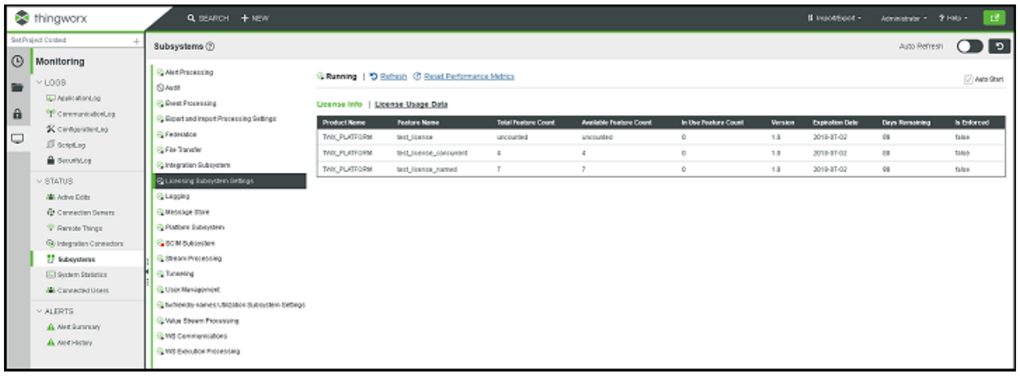
```
(RHEL) $ sudo systemctl start tomcat
```

Verify that a license file (`successful_license_capability_response.bin`) is created in the `ThingworxPlatform` folder.

12. To launch ThingWorx, go to `http://<servername>:<port>/Thingworx` in a Web browser.
13. Change the default password:
    a. In Composer, select **Administrator** > **Change Password**.
    b. In the **Change Password** window, enter **Current Password**, **New Password**, and **Confirm Password**.

> **Note**
>
> The password, which should not be easily guessed or a known, common password, is recommended to be at least 14 characters in length and should include a mix of uppercase and lowercase letters, numbers, and special characters.

14. Select **Done**.
15. (OPTIONAL STEP) To determine the status of your license, open the **Monitoring**>**Subsystem**>**Licensing Subsystem Settings** in Composer to confirm the list of features (licensed entities) included with the license. If there are no licensed entities present, you are in limited mode.



License Info | License Usage Data

Product Name	Feature Name	Total Feature Count	Available Feature Count	In Use Feature Count	Version	Expiration Date	Days Remaining	Is Enforced
TWX_PLATFORM	test_license	uncounted	uncounted	0	1.0	2019-07-02	98	false
TWX_PLATFORM	test_license_concurrent	4	4	0	1.0	2019-07-02	98	false
TWX_PLATFORM	test_license_named	7	7	0	1.0	2019-07-02	98	false

4

RHEL Installation

- H2/Azure SQL on page 73
- PostgreSQL on page 83

Note

See ThingWorx Installation Overview on page 6 for other options.

H2/Azure SQL

Install Java and Apache Tomcat (RHEL)

In the steps below, replace **xx** or **xxx** with the build number you are using.

1. If you are using AzureSQL for your database, go to Using Azure SQL Server as the Persistence Provider. Perform the steps in that section to set up the database, and you will be referred back to this section.
2. If you are using MSSQL for your database, go to Using MSSQL as the Persistence Provider. Perform the steps in that section to set up the database, and you will be referred back to this section.
3. Refer to the ThingWorx System Requirements for version requirements and then download the Java (JDK) RPM file from Oracle's website.
4. Run the Java installer:
   ```
   $ sudo rpm -i jdk-8uxxx-linux-x64.rpm
   ```
5. Create the directory and move the JDK:
   ```
   $ sudo mkdir -p /usr/lib/jvm
   $ sudo mv /usr/java/jdk1.8.0_xxx/ /usr/lib/jvm/
   ```
6. Set the Java alternatives:
   ```
   $ sudo alternatives --install /usr/bin/java java /usr/lib/jvm/jdk1.8.0_xxx/bin/
   java 1
   $ sudo alternatives --install /usr/bin/keytool keytool /usr/lib/jvm/jdk1.8.0_
   xxx/bin/keytool 1
   ```
7. Change access permissions:
   ```
   $ sudo chmod a+x /usr/bin/java
   $ sudo chmod a+x /usr/bin/keytool
   ```

   If you receive an error, use the following command:
   ```
   $ sudo chmod -f a+x /usr/bin/keytool
   ```
8. Change Owner:
   ```
   $ sudo chown -R root:root /usr/lib/jvm/jdk1.8.0_xxx/
   ```
9. Configure master links:
   ```
   $ sudo alternatives --config java
   ```
   Select the option that contains `/usr/lib/jvm/jdk1.8.0_xxx/bin/java`
   ```
   $ sudo rm /usr/java/latest
   $ sudo ln -s /usr/lib/jvm/jdk1.8.0_xxx /usr/java/latest
   $ sudo ln -s /usr/lib/jvm/jdk1.8.0_xxx/bin/keytool /usr/bin/keytool
   ```

   If you receive a `File Exists` error, ignore and continue.
   ```
   $ sudo alternatives --config keytool
   ```
10. Verify Java version. Your version may not be the version in the example that follows:
    ```
    $ java -version
    java version "1.8.0_xxx"
    Java(TM) SE Runtime Environment (build 1.8.0_xxx-bxx)
    Java HotSpot(TM) 64-Bit Server VM (build xx.xx-bxx, mixed mode)
    ```

11. Install Tomcat. Download the Tomcat installer. The steps in this process use Tomcat 8.5.*xx*, where *xx* is replaced with the version you are using.
```
$ wget https://archive.apache.org/dist/tomcat/tomcat-8/v8.5.xx/bin/apache-
tomcat-8.5.xx.tar.gz
```

Note

Best practice includes verifying the integrity of the Tomcat file by using the signatures or checksums for each release. Refer to Apache's documentation for more information.

12. Extract the contents:
```
$ tar -xf apache-tomcat-8.5.xx.tar.gz
```
13. Move Tomcat to `/usr/share/tomcat8.5`:
```
$ sudo mkdir -p /usr/share/tomcat8.5
$ sudo mv apache-tomcat-8.5.xx /usr/share/tomcat8.5/8.5.xx
```
14. Define environment variables in `/etc/environment`:
```
$ export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_xxx
$ export CATALINA_HOME=/usr/share/tomcat8.5/8.5.xx
```
15. Change directory to `/usr/share/tomcat8.5/8.5.xx`:
```
$ cd /usr/share/tomcat8.5/8.5.xx
```
16. Add user and group to the system:
```
$ sudo groupadd -r tomcat8.5
$ sudo useradd -r -d /usr/share/tomcat8.5 -g tomcat8.5 -s /bin/false tomcat8.5
$ sudo chown -R tomcat8.5:tomcat8.5 /usr/share/tomcat8.5
```
17. Change owner and access permissions of `bin/`, `lib/`, and `webapps/`:
```
$ sudo chown -Rh tomcat8.5:tomcat8.5 bin/ lib/ webapps/
$ sudo chmod 775 bin/ lib/ webapps/
```
18. Change owner and access permissions of `usr/share/tomcat8.5/8.5xx`:
```
sudo chown -R tomcat8.5:tomcat8.5 /usr/share/tomcat8.5/8.5.xx
sudo chmod -R 775 /usr/share/tomcat8.5/8.5.xx
```
19. Change owner and access permissions of `conf/`:
```
$ sudo chown -Rh root:tomcat8.5 conf/
$ sudo chmod -R 640 conf
sudo chown -R tomcat8.5:tomcat8.5 /usr/share/tomcat8.5/8.5.xx
sudo chmod -R 777 /usr/share/tomcat8.5/8.5.xx
```
Permissions and ownership should be revisited for a production system to increase security on a operating system level.
20. Change access permissions of `logs/`, `temp/`, and `work/`:
```
$ sudo chown -R tomcat8.5:adm logs/ temp/ work/
$ sudo chmod 760 logs/ temp/ work/
```
21. Create self-signed certificate:
```
$ /usr/lib/jvm/jdk1.8.0_xxx/jre/bin/keytool -genkey -alias tomcat8.5 -keyalg
RSA
```
22. Follow the instructions to complete the certificate creation process.

- Set the keystore password.
- Follow the prompts to set up your security certificate.
- Set the tomcat8.5 user password to be the same as the keystore password.

```
$ sudo cp ~/.keystore /usr/share/tomcat8.5/8.5.xx/conf/
$ sudo chown root:tomcat8.5 /usr/share/tomcat8.5/8.5.xx/conf/.keystore
$ sudo chmod 640 /usr/share/tomcat8.5/8.5.xx/conf/.keystore
```

23. Uncomment the `Manager` element in `context.xml` to prevent sessions from persisting across restarts. Open `/usr/share/tomcat8.5/8.5.xx/conf/context.xml` in a text editor (as root) and remove the '`<!—`' before '`<Manager pathname="" />`' and the '`-->`' after.
24. Save the file.
25. Define an Apache Manager user in `tomcat-users.xml`. Open `/usr/share/tomcat8.5/8.5.xx/conf/tomcat-users.xml` in a text editor (as root). Just above the final line `(</tomcat-users>)` add the following line:

```
<user username="<Tomcat username> " password="<Tomcat password> " roles=
"manager,manager-gui"/>
```

26. Save the file.

Note

The roles included are for ease of testing and can be removed if security is a concern.

Note

For security reasons, it is critical that you disable the AJP connector, if not already done so by default, by performing the following step.

27. In the location of the Tomcat installation, open `conf/server.xml` and search for the following line. If found, comment it out and save the file:

```
<Connector port ="8009" protocol="AJP/1.3" redirectPort="8443"/>
```

Note

In Apache Tomcat 9.0 and later, the **rejectIllegalHeader** attribute defaults to true. Manually modifying the `conf/web.xml` file to set this attribute to false is not recommended or supported by PTC.

28. Set up Tomcat as a service to start on boot. First, build JSVC if it is not already installed on your system:

```
$ sudo yum install gcc
$ cd /usr/share/tomcat8.5/8.5.xx/bin/
$ sudo tar xvfz commons-daemon-native.tar.gz
$ cd commons-daemon-*-native-src/unix
$ sudo ./configure --with-java=$JAVA_HOME

$ sudo yum install make
$ sudo make
$ sudo cp jsvc ../..
```

29. Create the Tomcat service file:

```
$ sudo touch /usr/lib/systemd/system/tomcat.service
```

30. Open `/usr/lib/systemd/system/tomcat.service` in a text editor (as root) and paste in the following:

Note

In the example below, set values for **-Xms** and **-Xmx** to 75% of the available OS memory (for example, 12GB for a 16GB RAM system). Refer to JVM Tuning for additional information.

```
[Unit]
Description=Apache Tomcat Web Application Container
After=network.target

[Service]
Type=forking
PIDFile=/var/run/tomcat.pid
Environment=CATALINA_PID=/var/run/tomcat.pid
Environment=JAVA_HOME=/usr/lib/jvm/jdk1.8.0_xxx
Environment=CATALINA_HOME=/usr/share/tomcat8.5/8.5.xx
Environment=CATALINA_BASE=/usr/share/tomcat8.5/8.5.xx
Environment=CATALINA_OPTS=

ExecStart=/usr/share/tomcat8.5/8.5.xx/bin/jsvc \
                              -Dcatalina.home=${CATALINA_HOME} \
                              -Dcatalina.base=${CATALINA_BASE} \
                              -Djava.awt.headless=true -Djava.net.
preferIPv4Stack=true -Dserver -XX:+UseNUMA \
                              -XX:+UseG1GC -Dfile.encoding=UTF-8 \
                              -Djava.library.path=${CATALINA_BASE}/webapps/
Thingworx/WEB-INF/extensions \
                              -Xms=<75% of available OS memory> \
                              -Xmx=<75% of available OS memory> \
                              -cp ${CATALINA_HOME}/bin/commons-daemon.jar:
${CATALINA_HOME}/bin/bootstrap.jar:${CATALINA_HOME}/bin/tomcat-juli.jar \
                              -user tomcat8.5 \
                              -java-home ${JAVA_HOME} \
                              -pidfile /var/run/tomcat.pid \
                              -errfile ${CATALINA_HOME}/logs/catalina.out \
                              -outfile ${CATALINA_HOME}/logs/catalina.out \
                              $CATALINA_OPTS \
                              org.apache.catalina.startup.Bootstrap

[Install]
```

```
WantedBy=multi-user.target
```

31. If you are installing the ThingWorx Platform for the first time, the Java option `-Duser.timezone=UTC` should be added to the ExecStart block above, immediately following the line that begins with **`-Djava.library.path`**. The `UTC` timezone does not recognize daylight savings time. Setting this option prevents overwriting data when daylight savings time changes occur.

Caution

Existing customers should NOT update this setting at this time.

32. Create a new file in the Tomcat `usr/share/tomcat8.5/8.5.xx/bin` file, `setenv.sh`:

```
CATALINA_OPTS="$CATALINA_OPTS -Djava.library.path=/usr/share/tomcat8.5/8.5.xx/
webapps/Thingworx/WEB-INF/extensions"
```

33. Set Tomcat to run on system start up:

```
$ sudo systemctl enable tomcat.service
```

This will allow the user to control the Tomcat service with the following commands:

```
sudo systemctl start tomcat
sudo systemctl stop tomcat
sudo systemctl restart tomcat
sudo systemctl status tomcat
```

34. In the location of the Tomcat installation, open `CATALINA_HOME/conf/web.xml`. Replace the default error page (default is stacktrace) by adding the following into the `web.xml` file. Place the following within the `web-app` tag (after the `welcome-file-list` tag). A well-configured web application will override this default in `CATALINA_HOME/webapps/APP_NAME/WEB-INF/web.xml` so it won't cause problems.

```
<error-page><exception-type>java.lang.Throwable</exception-type><location>/
error.jsp</location></error-page>
```

35. In the location of the Tomcat installation, open `CATALINA_HOME/conf/server.xml`. Add the following inside the `<Host> </Host>` tags:

```
<Valve className="org.apache.catalina.valves.ErrorReportValve" showReport=
"false" showServerInfo="false" />
```

36. Remove all the Tomcat webapps located in `/<path_to_tomcat>/webapps/`. Removing these apps prevents unnecessary access to Tomcat, specifically in the context that would allow users to view other users' cookies.

37. PTC strongly recommends the use of TLS when running ThingWorx. For detailed instructions on setting up TLS, refer to this technical support article.

38. If your application requires a specific cipher suite, refer to the following documentation for configuration information:

- https://www.jamf.com/jamf-nation/articles/384/configuring-supported-ciphers-for-tomcat-https-connections

Configuring Ulimit Settings

Running the Tomcat application server processes as the "root" user compromises the overall system security and violates industry standard best practices. To avoid this, PTC recommends that you modify the `/etc/security/limits.d/80-nofiles.conf` file to include settings specific to the user by which the application servers are intended to be run.

Configuration File Example

The following configuration is an example of the default Redhat 7.1 OS configuration located at `/etc/security/limits.d/80-nofiles.conf` with the needed changes. In the following example, `thingworx` is the name of the user for the app server.

```
thingworx                soft    nofile          30720
thingworx                hard    nofile          30720
```

To commit this change, log out and then log into your system.

Install ThingWorx/PostgreSQL

1. H2: Go to Install ThingWorx on page 78.
2. PostgreSQL: Go to Install and Configure PostgreSQL on page 88.

Install ThingWorx (Ubuntu/RHEL)

1. Create `/ThingworxStorage` and `/ThingworxBackupStorage` directories. If you haven't already done so, create the `/ThingworxPlatform` directory as well:
   ```
   $ sudo mkdir /ThingworxStorage /ThingworxBackupStorage /ThingworxPlatform
   ```
2. Change owner and access permissions of `/ThingworxPlatform`, `/ThingworxStorage` and `/ThingworxBackupStorage`. Without these permissions, the server will fail to start.
   ```
   $ sudo chown tomcat8.5:tomcat8.5 /ThingworxStorage /ThingworxBackupStorage
   /ThingworxPlatform
   $ sudo chmod 775 /ThingworxStorage /ThingworxBackupStorage /ThingworxPlatform
   ```
3. If you have not already done so, obtain the `Thingworx.war` file from PTC Software Downloads.
4. Move the `Thingworx.war` to `$CATALINA_HOME/webapps`.
   ```
   $ sudo mv Thingworx.war $CATALINA_HOME/webapps
   $ sudo chown tomcat8.5:tomcat8.5 $CATALINA_HOME/webapps/Thingworx.war
   $ sudo chmod 775 $CATALINA_HOME/webapps/Thingworx.war
   ```
5. Place the `platform-settings.json` in the `ThingworxPlatform` folder.

6. If you are using H2 as a database perform this step. If you are not using H2 as a database, go to the next step. Add a username and password for H2 in the `platform-settings.json` file. See platform-settings.json Configuration Details on page 121 for more information.

   Note

   ThingWorx connections to the H2 database require a username and password defined by the user, or the server will not start. This design fully mitigates any potential vulnerability represented by CVE-2018-10054.

```
},
"PersistenceProviderPackageConfigs":{
  "H2PersistenceProviderPackage":{
      "ConnectionInformation":
{
    "password": "<addsecurepassword>",
     "username": "twadmin"
}
},
```

7. Configure the Administrator password:Add the following **AdministratorUserSettings** section (in **PlatformSettingsConfig**) to your `platform-settings.json` file along with a password that is at least 14 characters long. Reference platform-settings.json Configuration Details on page 121 for more information on placement. See Passwords for additional information on setting passwords. Do not copy and paste the sample below, as it may cause bad formatting in your `platform-settings.json`. Instead, click here and copy from the file.

```
{
    "PlatformSettingsConfig": {
        "AdministratorUserSettings": {
            "InitialPassword": "changeme"
        }
    }
}
```

Note

If Tomcat fails to start and reports the error message: `Check the InitialPassword setting in the AdministratorUserPassword section in platform-settings.json...`, check the following:

- The password setting exists in `platform-settings.json`
- The password is valid (14 or more characters by default)
- The `platform-settings.json` file is formatted correctly - bad formatting could lead to errors

8. Enable extension import. By default, extension import is disabled for all users. Add the following to the `platform-settings.json` file. Update the following *ExtensionPackageImportPolicy* parameters to `true` to allow extensions to be imported. See Importing Extensions for best practices on configuration.

```
"ExtensionPackageImportPolicy": {
             "importEnabled": <true or false>,
             "allowJarResources": <true or false>,
             "allowJavascriptResources": <true or false>,
             "allowCSSResources": <true or false>,
             "allowJSONResources": <true or false>,
             "allowWebAppResources": <true or false>,
             "allowEntities": <true or false>,
             "allowExtensibleEntities": <true or false>
        },
```

9. Configure licensing:

   - Open the `platform-settings.json` file and add the following to the `PlatformSettingsConfig` section (reference platform-settings.json Configuration Options on page 121 for more information on placement.)

   Note

   If you are performing a disconnected installation (no internet access), you do not need to add to the `platform-settings.json` file. Refer to the Licensing Guide for disconnected sites and skip this step.

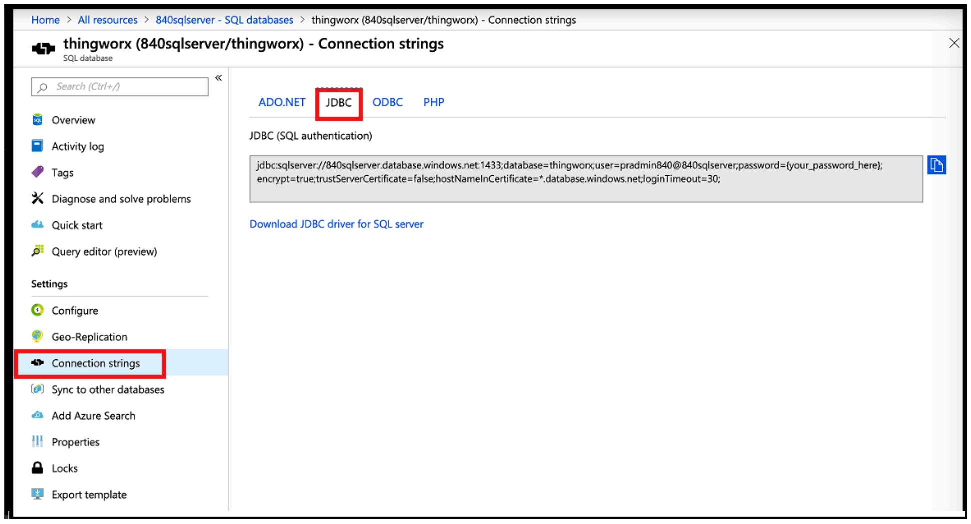
```
"LicensingConnectionSettings":{
      "username":"PTC Support site user name",
      "password":"PTC Support site password"
}
```

Note

If the settings are filled out incorrectly or if the server can’t connect, a License Request text file (`licenseRequestFile.txt`) is created in the `ThingworxPlatform` folder. In this scenario, a license must be created manually. (If it is not created, ThingWorx will start in limited mode. Limited mode does not allow you to persist licensed entities to the database. Licensed entities are Things, Mashups, Masters, Gadgets, Users, and Persistence Providers).

More information on obtaining a ThingWorx disconnected site license through our License Management site can be found in the the Licensing Guide for disconnected sites (no connection to PTC Support portal).

10. Encrypt the license server password by following the steps in Encrypting Passwords on page 118.
11. If you are using Azure SQL as your database, follow these steps to download the JDBC driver. Skip this step if you are not using Azure SQL.
    a. Go to the Azure portal and navigate to your ThingWorx database.
    b. Select **Connection strings**.
    c. Select the **JDBC** tab.


Home > All resources > 840sqlserver - SQL databases > thingworx (840sqlserver/thingworx) - Connection strings
thingworx (840sqlserver/thingworx) - Connection strings
SQL database
Search (Ctrl+/)
Overview
Activity log
Tags
Diagnose and solve problems
Quick start
Query editor (preview)
Settings
Configure
Geo-Replication
Connection strings
Sync to other databases
Add Azure Search
Properties
Locks
Export template
ADO.NET
JDBC
ODBC
PHP
Download JDBC driver for SQL server

JDBC (SQL authentication)

```
jdbc:sqlserver://840sqlserver.database.windows.net:1433;database=thingworx;user=pradmin840@840sqlserver;password={your_password_here};
encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;
```

    d. Select **Download Microsoft JDBC Driver for SQL Server**.
    e. Select **Microsoft JDBC Driver 6.0 for SQL Server**.

f. Extract and copy the downloaded binary in your ThingWorx VM to your Tomcat `lib` directory.

12. Start Tomcat.

```
(UBUNTU) sudo service tomcat8.5 start

(RHEL) $ sudo systemctl start tomcat
```

Verify that a license file (`successful_license_capability_response.bin`) is created in the `ThingworxPlatform` folder.

13. To launch ThingWorx, go to `http://<servername>:<port>/Thingworx` in a Web browser.

14. Change the default password:

a. In Composer, select **Administrator** > **Change Password**.

b. In the **Change Password** window, enter **Current Password**, **New Password**, and **Confirm Password**.

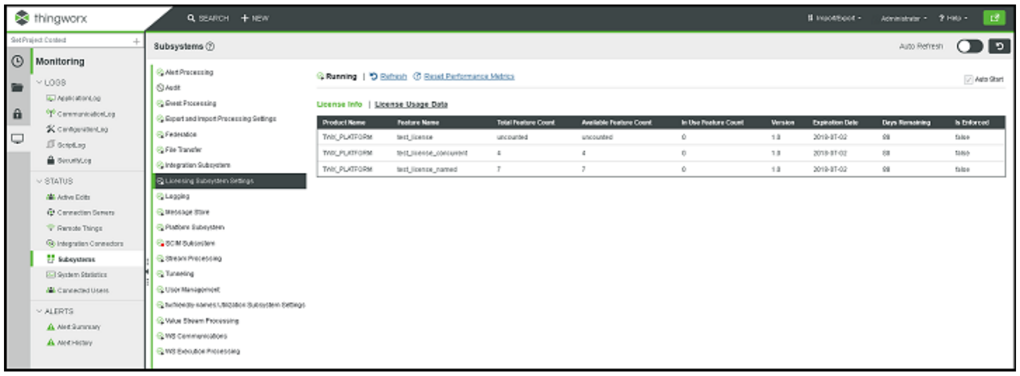
Note

The password, which should not be easily guessed or a known, common password, is recommended to be at least 14 characters in length and should include a mix of uppercase and lowercase letters, numbers, and special characters.

c. Delete the initial password from the `platform-settings.json` file.

15. Select **Done**.

16. (OPTIONAL STEP) To determine the status of your license, open the **Monitoring**>**Subsystem**>**Licensing Subsystem Settings** in Composer to confirm the list of features (licensed entities) included with the license. If there are no licensed entities present, you are in limited mode.



Product Name	Feature Name	Total Feature Count	Available Feature Count	In Use Feature Count	Version	Expiration Date	Days Remaining	Is Enforced
TWX_PLATFORM	test_license	uncounted	uncounted	0	1.0	2019-07-02	98	false
TWX_PLATFORM	test_license_concurrent	4	4	0	1.0	2019-07-02	98	false
TWX_PLATFORM	test_license_named	7	7	0	1.0	2019-07-02	98	false

PostgreSQL

Install Java and Apache Tomcat (RHEL)

In the steps below, replace **xx** or **xxx** with the build number you are using.

1. If you are using AzureSQL for your database, go to Using Azure SQL Server as the Persistence Provider. Perform the steps in that section to set up the database, and you will be referred back to this section.
2. If you are using MSSQL for your database, go to Using MSSQL as the Persistence Provider. Perform the steps in that section to set up the database, and you will be referred back to this section.
3. Download the Java (JDK) RPM file from Oracle's website.
4. Run the Java installer:
```
$ sudo rpm -i jdk-8uxxx-linux-x64.rpm
```
5. Create the directory and move the JDK:
```
$ sudo mkdir -p /usr/lib/jvm
$ sudo mv /usr/java/jdk1.8.0_xxx/ /usr/lib/jvm/
```
6. Set the Java alternatives:
```
$ sudo alternatives --install /usr/bin/java java /usr/lib/jvm/jdk1.8.0_xxx/bin/
java 1
$ sudo alternatives --install /usr/bin/keytool keytool /usr/lib/jvm/jdk1.8.0_
xxx/bin/keytool 1
```
7. Change access permissions:
```
$ sudo chmod a+x /usr/bin/java
$ sudo chmod a+x /usr/bin/keytool
```

Note

If you receive an error, use the following:

```
$ sudo chmod -f a+x /usr/bin/keytool
```

8. Change Owner:
```
$ sudo chown -R root:root /usr/lib/jvm/jdk1.8.0_xxx/
```
9. Configure master links:
```
$ sudo alternatives --config java
```

Note

Select the option that contains `/usr/lib/jvm/jdk1.8.0_xxx/bin/java`

```
$ sudo rm /usr/java/latest
$ sudo ln -s /usr/lib/jvm/jdk1.8.0_xxx /usr/java/latest
$ sudo ln -s /usr/lib/jvm/jdk1.8.0_xxx/bin/keytool /usr/bin/keytool
```

Note

This may return a `File Exists` error. If so, ignore and continue.

```
$ sudo alternatives --config keytool
```

10. Verify Java version. Your build version may differ.
    ```
    $ java -version
    java version "1.8.0_xxx"
    Java(TM) SE Runtime Environment (build 1.8.0_xxx-bxx)
    Java HotSpot(TM) 64-Bit Server VM (build 25.45-bxx, mixed mode)
    ```
11. Install Tomcat. Download the Tomcat installer. The steps in this process use Tomcat 8.5.*xx*, where *xx* is replaced with the version you are using.
    ```
    $ wget https://archive.apache.org/dist/tomcat/tomcat-8/v8.5.xx/bin/apache-
    tomcat-8.5.xx.tar.gz
    ```

Note

Best practice includes verifying the integrity of the Tomcat file by using the signatures or checksums for each release. Refer to Apache's documentation for more information.

12. Extract the contents:
    ```
    $ tar -xf apache-tomcat-8.5.xx.tar.gz
    ```
13. Move Tomcat to `/usr/share/tomcat8.5`:
    ```
    $ sudo mkdir -p /usr/share/tomcat8.5
    $ sudo mv apache-tomcat-8.5.xx /usr/share/tomcat8.5/8.5.xx
    ```
14. Define environment variables in `/etc/environment`:
    ```
    $ export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_xxx
    $ export CATALINA_HOME=/usr/share/tomcat8.5/8.5.xx
    ```
15. Change directory to `/usr/share/tomcat8.5/8.5.xx`:
    ```
    $ cd /usr/share/tomcat8.5/8.5.xx
    ```
16. Add user and group to the system:
    ```
    $ sudo groupadd -r tomcat8.5
    $ sudo useradd -r -d /usr/share/tomcat8.5 -g tomcat8.5 -s /bin/false tomcat8.5
    $ sudo chown -R tomcat8.5:tomcat8.5 /usr/share/tomcat8.5
    ```
17. Change owner and access permissions of `bin/`, `lib/`, and `webapps/`:
    ```
    $ sudo chown -Rh tomcat8.5:tomcat8.5 bin/ lib/ webapps/
    $ sudo chmod 775 bin/ lib/ webapps/
    ```
18. Change owner and access permissions of `usr/share/tomcat8.5/8.5xx`:
    ```
    sudo chown -R tomcat8.5:tomcat8.5 /usr/share/tomcat8.5/8.5.xx
    sudo chmod -R 775 /usr/share/tomcat8.5/8.5.xx
    ```

19. Change owner and access permissions of `conf/`. Permissions and ownership should be revisited for a production system to increase security on a operating system level.
```
$ sudo chown -Rh root:tomcat8.5 conf/
$ sudo chmod -R 640 conf
sudo chown -R tomcat8.5:tomcat8.5 /usr/share/tomcat8.5/8.5.xx
sudo chmod -R 777 /usr/share/tomcat8.5/8.5.xx
```
20. Change access permissions of `logs/`, `temp/`, and `work/`:
```
$ sudo chown -R tomcat8.5:adm logs/ temp/ work/
$ sudo chmod 760 logs/ temp/ work/
```
21. Create self-signed certificate:
```
$ /usr/lib/jvm/jdk1.8.0_xxx/jre/bin/keytool -genkey -alias tomcat8.5 -keyalg
RSA
```
22. Follow the instructions to complete the certificate creation process.
    - Set the keystore password.
    - Follow the prompts to set up your security certificate.
    - Set the tomcat8.5 user password to the same as the keystore password.
    ```
    $ sudo cp ~/.keystore /usr/share/tomcat8.5/8.5.xx/conf/
    $ sudo chown root:tomcat8.5 /usr/share/tomcat8.5/8.5.xx/conf/.keystore
    $ sudo chmod 640 /usr/share/tomcat8.5/8.5.xx/conf/.keystore
    ```
23. Uncomment the `Manager` element in `context.xml` to prevent sessions from persisting across restarts. Open `/usr/share/tomcat8.5/8.5.xx/conf/context.xml` in a text editor (as root) and remove the '`<!—`' before '`<Manager pathname="" />`' and the '`-->`' after.
24. Save the file.
25. Define an Apache Manager user in `tomcat-users.xml`. Open `/usr/share/tomcat8.5/8.5.xx/conf/tomcat-users.xml` in a text editor (as root). Just above the final line `(</tomcat-users>)` add the following line:
```
<user username="<Tomcat username> " password="<Tomcat password> " roles=
"manager,manager-gui"/>
```
26. Save the file.

Note

The roles included are for ease of testing and can be removed if security is a concern.

Note

For security reasons, it is critical that you disable the AJP connector, if not already done so by default, by performing the following step.

27. In the location of the Tomcat installation, open `conf/server.xml` and search for the following line. If found, comment it out and save the file:
```
<Connector port ="8009" protocol="AJP/1.3" redirectPort="8443"/>
```

Note

In Apache Tomcat 9.0 and later, the **rejectIllegalHeader** attribute defaults to true. Manually modifying the `conf/web.xml` file to set this attribute to false is not recommended or supported by PTC.

28. Set up Tomcat as a service to start on boot. First, build JSVC if it is not already installed on your system.
```
$ sudo yum install gcc
$ cd /usr/share/tomcat8.5/8.5.xx/bin/
$ sudo tar xvfz commons-daemon-native.tar.gz
$ cd commons-daemon-*-native-src/unix
$ sudo ./configure --with-java=$JAVA_HOME

$ sudo yum install make
$ sudo make
$ sudo cp jsvc ../..
```
29. Create the Tomcat service file:
```
$ sudo touch /usr/lib/systemd/system/tomcat.service
```
30. Open `/usr/lib/systemd/system/tomcat.service` in a text editor (as root) and paste in the following:

Note

In the example below, set values for **-Xms** and **-Xmx** to 75% of the available OS memory (for example, 12GB for a 16GB RAM system). Refer to JVM Tuning for additional information.

```
[Unit]
Description=Apache Tomcat Web Application Container
After=network.target

[Service]
Type=forking
PIDFile=/var/run/tomcat.pid
Environment=CATALINA_PID=/var/run/tomcat.pid
Environment=JAVA_HOME=/usr/lib/jvm/jdk1.8.0_xxx
```

```
Environment=CATALINA_HOME=/usr/share/tomcat8.5/8.5.xx
Environment=CATALINA_BASE=/usr/share/tomcat8.5/8.5.xx
Environment=CATALINA_OPTS=

ExecStart=/usr/share/tomcat8.5/8.5.xx/bin/jsvc \
                              -Dcatalina.home=${CATALINA_HOME} \
                              -Dcatalina.base=${CATALINA_BASE} \
                              -Djava.awt.headless=true -Djava.net.
preferIPv4Stack=true -Dserver -XX:+UseNUMA \
                              -XX:+UseG1GC -Dfile.encoding=UTF-8 \
                              -Djava.library.path=${CATALINA_BASE}/webapps/
Thingworx/WEB-INF/extensions \
                              -Xms=<75% of available OS memory> \
                              -Xmx=<75% of available OS memory> \
                              -cp ${CATALINA_HOME}/bin/commons-daemon.jar:
${CATALINA_HOME}/bin/bootstrap.jar:${CATALINA_HOME}/bin/tomcat-juli.jar \
                              -user tomcat8.5 \
                              -java-home ${JAVA_HOME} \
                              -pidfile /var/run/tomcat.pid \
                              -errfile ${CATALINA_HOME}/logs/catalina.out \
                              -outfile ${CATALINA_HOME}/logs/catalina.out \
                              $CATALINA_OPTS \
                              org.apache.catalina.startup.Bootstrap

[Install]
WantedBy=multi-user.target
```

31. Create a new file in the Tomcat `usr/share/tomcat8.5/8.5.xx/bin` file named `setenv.sh`:

```
CATALINA_OPTS="$CATALINA_OPTS -Djava.library.path=/usr/share/tomcat8.5/8.5.xx/
webapps/Thingworx/WEB-INF/extensions"
```

32. Set Tomcat to run on system start up and allow the user to control the Tomcat service:

```
$ sudo systemctl enable tomcat.service
sudo systemctl start tomcat
sudo systemctl stop tomcat
sudo systemctl restart tomcat
sudo systemctl status tomcat
```

33. In the location of the Tomcat installation, open `CATALINA_HOME/conf/web.xml`. Replace the default error page (default is stacktrace) by adding the following into the `web.xml` file. Place the following within the `web-app` tag (after the `welcome-file-list` tag). A well-configured web application will override this default in `CATALINA_HOME/webapps/APP_NAME/WEB-INF/web.xml` so it won't cause problems.

```
<error-page><exception-type>java.lang.Throwable</exception-type><location>/
error.jsp</location></error-page>
```

34. In the location of the Tomcat installation, open `CATALINA_HOME/conf/server.xml`. Add the following inside the `<Host> </Host>` tags:

```
<Valve className="org.apache.catalina.valves.ErrorReportValve" showReport=
"false" showServerInfo="false" />
```

35. Remove all the Tomcat webapps located in `/<path_to_tomcat>/webapps/`. Removing these apps prevents unnecessary access to Tomcat, specifically in the context that would allow users to view other users' cookies.

36. PTC strongly recommends the use of TLS when running ThingWorx. For detailed instructions on setting up TLS, refer to this technical support article.
37. If your application requires a specific cipher suite, refer to the following documentation for configuration information:
    - https://www.jamf.com/jamf-nation/articles/384/configuring-supported-ciphers-for-tomcat-https-connections

Configuring Ulimit Settings

Running the Tomcat application server processes as the "root" user compromises the overall system security and violates industry standard best practices. To avoid this, PTC recommends that you modify the `/etc/security/limits.d/80-nofiles.conf` file to include settings specific to the user by which the application servers are intended to be run.

Configuration File Example

The following configuration is an example of the default Redhat 7.1 OS configuration located at `/etc/security/limits.d/80-nofiles.conf` with the needed changes. In the following example, `thingworx` is the name of the user for the app server.

```
thingworx                soft    nofile          30720
thingworx                hard    nofile          30720
```

To commit this change, log out and then log into your system.

Install ThingWorx/PostgreSQL

1. H2: Go to Install ThingWorx (Ubuntu/RHEL) on page 78
2. PostgreSQL: Go to Install and Configure PostgreSQL on page 88.

Install and Configure PostgreSQL (RHEL)

The instructions provided below are intended for the PostgreSQL administrator (not the DB host servers). If you are including the HA layer to your implementation, refer to the the ThingWorx High Availability Administrator's Guide.

Install PostgreSQL and Create a New User Role

Note

These steps assume a version of RHEL with a GUI (X11) and an active account with access to the RHEL software repositories. If you are working without a GUI, skip installing PgAdmin and refer to this support article for alternate instructions. If you do not have access to the official RHEL software sources, you can set up a free open source repository from the EPEL team. (this site is not provided or controlled by PTC).

1. Refer to the ThingWorx System Requirements for information on supported PostgreSQL versions.

   Note

   The steps in this procedure use PostgreSQL version *x.x*, where *x.x* is the supported version.

2. Add the PostgreSQL repository to Yum and install.
3. Install PgAdmin, the PostgreSQL admin tool:

   ```
   $ sudo yum install pgadmin4
   ```

   Note

   To install PgAdmin via the command line, reference https://wiki.postgresql.org/wiki/Manual_Setup_at_the_Command_Line.

4. Initialize and launch the database:

   ```
   $ sudo /usr/pgsql-x.x/bin/postgresqlx.x-setup initdb
   ```

   Set the PostgreSQL service to start on boot:

   ```
   $ sudo chkconfig postgresql-x.x on
   $ sudo service postgresql-x.x start
   ```

5. Set up the password for the PostgreSQL user:

   ```
   $ sudo passwd postgres
   ```

6. Enter the password for the PostgreSQL user. You will use this password in later steps.

Note

The password, which should not be easily guessed or a known, common password, should be at least 14 characters in length and include a mix of uppercase and lowercase letters, numbers, and special characters.

7. Set up the PostgreSQL user in psql. The <*unique PostgreSQL password*> value is what you entered above.

Note

If the PostgreSQL database is not located on the same server as ThingWorx, then refer to the section Configure PostgreSQL Database Located on a Separate Server than ThingWorx on page 92 and skip the next two steps. While it is optional to have the PostgreSQL database on the same server as ThingWorx in a development environment, it should be separate in all production environments.

```
$ sudo -u postgres psql -c "ALTER ROLE postgres WITH password '<unique
PostgreSQL password>'"
```

8. If using the command line, open the following files and edit as noted. Skip this step if using PgAdmin.
   - `/var/lib/pgsql/x.x/data/postgresql.conf/postgresql.conf`: Uncomment `listen addresses` and `port`. The default settings of `localhost` and `5432` are usually sufficient.
   - `/var/lib/pgsql/x.x/data/pg_hba.conf`: Set **Method** to `md5`
9. Configure PgAdmin. Skip this step if you are not using PgAdmin.

   ```
   $ sudo pgadmin3
   ```

   - In the PgAdmin GUI, click on **file->Open postgresql.conf**
   - Open `/var/lib/pgsql/x.x/data/postgresql.conf`
   - Put a check next to **listen addresses** and **port**. The default settings of **localhost** and **5432** are usually sufficient.
   - Save and close.
   - Click on **file->Open pg_hba.conf**
   - Open `/var/lib/pgsql/x.x/data/pg_hba.conf`
   - Double-click on the database ‘all’ line with address 127.0.0.1/32
   - Set Method to **md5**
   - Click **OK**
   - Save and exit

- Close PgAdmin.

10. Restart the PostgreSQL service:

```
$ sudo service postgresql-x.x restart
```

11. Set up PgAdmin to connect to the database:

```
$ sudo pgadmin3
```

12. Click the plug icon to add a connection to a server in the top left corner and fill out the following:

```
Name: PostgreSQL x.x
Host: localhost
Port: 5432
Service: <blank>
Maintenance DB: postgres
Username: postgres
Password: <unique PostgreSQL password as set previously>
Store password: Checked
Group: Servers
```

13. Click **OK**.
14. Create a new user role:

Note

The following command can be used if you are not using PgAdmin:

```
sudo -u postgres psql -c "CREATE USER twadmin WITH PASSWORD '<unique postgres
password>';"
```

a. Right click PostgreSQLx.x(localhost:5432).

b. Select **NewObject**>**NewLogin Role**. On the **Properties** tab, enter a name in the **Role** name field.

c. On the **Definition** tab, in the **Password** field, enter a unique password (you will be prompted to enter it twice).

Note

The password, which should not be easily guessed or a known, common password, should be at least 14 characters in length and include a mix of uppercase and lowercase letters, numbers, and special characters. You will need to re-enter this password in later steps.

d. Click **OK**.

Configure PostgreSQL Database Located on a Separate Server than ThingWorx

Note

This section is optional for development environments, but should be implemented in all production environments.

By default, the PostgreSQL server is installed in a locked-down state. The server will only listen for connections from the local machine In order to get ThingWorx to communicate to the PostgreSQL server, some configuration changes need to be made so that PostgreSQL knows to listen for connections from other users (thingworx user, default is twadmin) and/or other machines (ThingWorx installed on a separate server).

You will need to know where your PostgreSQL data directory resides for these steps. On Linux, the location of the data folder, or even the configuration files can change based on distribution and installation method (download or package manager install). This location will be referred to as `<PGDATA>` in these instructions.

Modify the `pg_hba.conf` file and add the following lines based on your desired configuration:

If you want to allow all IPv4 addresses to connect:	`hostallall0.0.0.0/0md5`
If you want to allow only a specific IPv4 address to connect (Replace <*ipAddress*> with the IP address of the machine making the connection):	`hostallall<ipAddress>/32md5`
If you want to allow all IPv6 addresses to connect:	`hostallall::0/0md5`
If you want to allow only a specific IPv6 address to connect (Replace <ipv6Address> with the appropriate address):	`hostallall<ipv6Address>/128md5`

Any other combination is possible by using additional allowance lines (individual IPs or ranges) or subnet masks appropriate to the machines that require access to the PostgreSQL database.

Any change to this file requires a restart of the database service.

Note

For additional information about configuring the `pg_hba.conf` file, see the PostgreSQL documentation.

Enabling PostgreSQL to Listen for all Connections

On Linux installations of PostgreSQL, there is an additional configuration step required to configure the PostgreSQL server to listen for connections.

1. In the `postgresql.conf` file, uncomment and update the `listen_addresses` line:

   ```
   Uncomment the listen_addresses line and change localhost to '*'
   # Listen on all addresses. Requires restart.
   listen_addresses = '*'
   ```

2. Restart the PostgreSQL server.

Configure and Execute the PostgreSQL Database Script

To set up the PostgreSQL database and tablespace, the `thingworxPostgresDBSetup` script must be configured and executed.

1. Create the `ThingworxPostgresqlStorage` folder on the drive that the `ThingworxStorage` folder is located (in the root directory by default). Note the following:

   - If you create the folder using the `-d<databasename>` command, you do not have to use the PostgreSQL user
   - You must specify the `-l` option to a path that exists. For example, `-l D:\ThingworxPostgresqlStorage`. The script does not create the folder for you.
   - The folder must have appropriate ownership and access rights. It should be owned by the same user who runs the PostgreSQL service, and have Full Control assigned to that user - this user is generally NETWORK_SERVICE, but may differ in your environment.

   ```
   $ sudo mkdir /ThingworxPostgresqlStorage
   $ sudo chown postgres:postgres /ThingworxPostgresqlStorage
   $ sudo chmod 755 /ThingworxPostgresqlStorage
   ```

2. Obtain the `thingworxPostgresDBSetup` script from the ThingWorx software download package. The script is located in the `install` folder. ThingWorx downloads are available in PTC Software Downloads.
3. If necessary, configure the script. Reference the options in the table below.

Note

This example uses the x.x.x download from the PTC site. Change the file name to the version you are using.

```
$ sudo unzip MED-61111-CD-08x_ThingWorx-Platform-Postgres-x-x-x.zip
$ cd install
```

thingworxPostgresDBSetup Script Options

Option	Parameter	Default	Description	Example
t or -T	tablespace	thingworx	Tablespace name	`-t thingworx`
-p or -P	port	5432	Port number of PostgreSQL	`-p 5432`
-d or -D	database	thingworx	PostgreSQL Database name to create	`-d thingworx`
-h or -H	host	localhost	Name of the host	`-h localhost`
-l or -L	tablespace_location	/Thingworx-Postgresql-Storage	Required. Location in the file system where the files representing database objects are stored.	`-l` or `-L`
-a or -A	adminuser-name	postgres	Administrator Name	`-a postgres`
-u or -U	thingworxu-sername	twadmin	User name that has permissions to write to the database.	`-u twadmin`

4. To set up the database and tablespace with a default PostgreSQL installation that has a PostgreSQL database and a PostgreSQL user name, enter:

```
$ sudo sh thingworxPostgresDBSetup.sh -a postgres -u <user role name> -l
/ThingworxPostgresqlStorage
```

5. Execute the script.

Configure and Execute the Model/Data Provider Schema Script

To set up the PostgreSQL model/data provider schema, the `thingworxPostgresSchemaSetup` script must be configured and executed. This will set up the public schema under your database on the PostgreSQL instance installed on the localhost.

1. Obtain and open the `thingworxPostgresSchemaSetup` file from the ThingWorx software download package. The script is located in the `install` folder.
2. If necessary, configure the script. Reference the options in the table below.

Note

The script can be run with the default parameters as:

```
$ sudo sh thingworxPostgresSchemaSetup.sh
```

thingworxPostgresSchemaSetup Script Options

Option	Parameter	Default	Description	Example
-h or -H	host	localhost	IP or host name of the database.	`-h localhost`
-p or -P	port	5432	Port number of PostgreSQL.	`-p 5432`
-d or -D	database	thingworx	Database name to use.	`-d thingworx`
-s or -S	schema	public	Schema name to use.	`-s mySchema`

thingworxPostgresSchemaSetup Script Options (continued)

Option	Parameter	Default	Description	Example
-u or -U	username	twadmin	Username to update the database schema	`-u twadmin`
-o or -O	option	all	There are three options: • all: Sets up the model and data provider schemas into the specified database. • model: Sets up the model provider schema into the specified database. • data: Sets up the data provider schema into the specified database.	`-o data`

3. Execute the script.

Note

The username should match the PostgreSQL username that was previously created.

Configure platform-settings.json

1. Create the `ThingworxPlatform` folder at the root of the drive where Tomcat was installed or as a system variable. Note the following:
   - To specify the location where ThingWorx stores its settings, you can set the `THINGWORX_PLATFORM_SETTINGS` environment variable to the desired location. Ensure that the folder referenced by `THINGWORX_PLATFORM_SETTINGS` exists and is writable by the Tomcat user. This environment variable should be configured as part of the system environment variables.
   - The ThingWorx server will fail to start if it does not have read and write access to this folder.
2. Place the `platform-settings.json` file into the `ThingworxPlatform` folder. This file is available in the software download.

   ```
   $ sudo cp platform-settings.json /ThingworxPlatform/
   ```
3. Open `platform-settings.json` and configure as necessary. Refer to the configuration options in platform-settings.json Configuration Details on page 121.

Note

If your PostgreSQL server is not the same as your ThingWorx server, and you are having issues with your ThingWorx installation, review your Tomcat logs and `platform-settings.json` file. The default installation assumes both servers are on the same machine.

Encrypt the PostgreSQL Password

- Encrypt the password by following the steps in Encrypting Passwords on page 118

(Optional) Installing the PostgreSQL Client Package and PostgreSQL User

In order to issue PostgreSQL commands from the client machine to the PostgreSQL server, do so from a PostgreSQL user. The `postgresql-client-x.x` package can be installed on the client machine, refer to your distributions documentation on how to install it. This package provides some administration tools such as `psql`.

Install ThingWorx

Go to Install ThingWorx on page 98.

Install ThingWorx (Ubuntu/RHEL)

1. Create `/ThingworxStorage` and `/ThingworxBackupStorage` directories. If you haven't already done so, create the `/ThingworxPlatform` directory as well:
```
$ sudo mkdir /ThingworxStorage /ThingworxBackupStorage /ThingworxPlatform
```
2. Change owner and access permissions of `/ThingworxPlatform`, `/ThingworxStorage` and `/ThingworxBackupStorage`. Without these permissions, the server will fail to start.
```
$ sudo chown tomcat8.5:tomcat8.5 /ThingworxStorage /ThingworxBackupStorage
/ThingworxPlatform
$ sudo chmod 775 /ThingworxStorage /ThingworxBackupStorage /ThingworxPlatform
```
3. If you have not already done so, obtain the `Thingworx.war` file from PTC Software Downloads.
4. Move the `Thingworx.war` to `$CATALINA_HOME/webapps`.
```
$ sudo mv Thingworx.war $CATALINA_HOME/webapps
$ sudo chown tomcat8.5:tomcat8.5 $CATALINA_HOME/webapps/Thingworx.war
$ sudo chmod 775 $CATALINA_HOME/webapps/Thingworx.war
```
5. Place the `platform-settings.json` in the `ThingworxPlatform` folder.
6. Configure the Administrator password:Add the following **AdministratorUserSettings** section (in **PlatformSettingsConfig**) to your `platform-settings.json` file along with a password that is at least 14 characters long. Reference platform-settings.json Configuration Details on page 121 for more information on placement. See this topic, Passwords, in the ThingWorx Help Center for additional information on setting passwords. Do not copy and paste the sample below, as it may cause bad formatting in your `platform-settings.json`. Instead, click here and copy from the file.

```
{
    "PlatformSettingsConfig": {
        "AdministratorUserSettings": {
            "InitialPassword": "changeme"
        }
    }
}
```

Note

If Tomcat fails to start and reports the error message: `Check the InitialPassword setting in the AdministratorUserPassword section in platform-settings.json...`, check the following:

- The password setting exists in `platform-settings.json`
- The password is valid (14 or more characters by default)
- The `platform-settings.json` file is formatted correctly - bad formatting could lead to errors

7. Enable extension import. By default, extension import is disabled for all users. Add the following to the `platform-settings.json` file. Update the following *ExtensionPackageImportPolicy* parameters to `true` to allow extensions to be imported. See Importing Extensions in the ThingWorx Help Center for best practices on configuration.

```
"ExtensionPackageImportPolicy": {
             "importEnabled": <true or false>,
             "allowJarResources": <true or false>,
             "allowJavascriptResources": <true or false>,
             "allowCSSResources": <true or false>,
             "allowJSONResources": <true or false>,
             "allowWebAppResources": <true or false>,
             "allowEntities": <true or false>,
             "allowExtensibleEntities": <true or false>
        },
```

8. Configure licensing:

   - Open the `platform-settings.json` file and add the following to the `PlatformSettingsConfig` section (reference platform-settings.json Configuration Options on page 121 for more information on placement.)

   Note

   If you are performing a disconnected installation (no internet access), you do not need to add to the `platform-settings.json` file. Refer to the Licensing Guide for disconnected sites and skip this step.

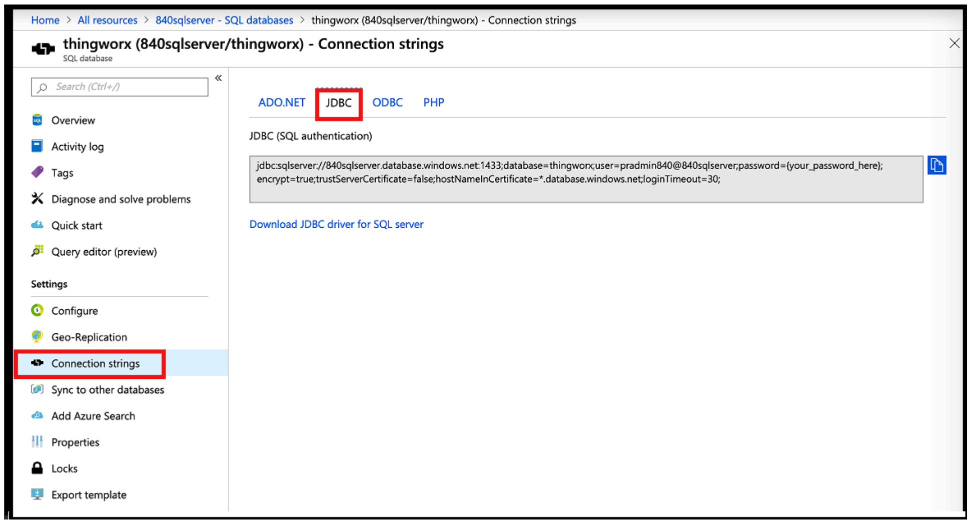
```
"LicensingConnectionSettings":{
      "username":"PTC Support site user name",
      "password":"PTC Support site password"
      }
```

Note

If the settings are filled out incorrectly or if the server can't connect, a License Request text file (`licenseRequestFile.txt`) is created in the `ThingworxPlatform` folder. In this scenario, a license must be created manually. (If it is not created, ThingWorx will start in limited mode. Limited mode does not allow you to persist licensed entities to the database. Licensed entities are Things, Mashups, Masters, Gadgets, Users, and Persistence Providers).

More information on obtaining a ThingWorx disconnected site license through our License Management site can be found in the Licensing Guide for disconnected sites (no connection to PTC Support portal).

9. Encrypt the license server password by following the steps in Encrypting Passwords on page 118.
10. If you are using Azure SQL as your database, follow these steps to download the JDBC driver. Skip this step if you are not using Azure SQL.
    a. Go to the Azure portal and navigate to your ThingWorx database.
    b. Select **Connection strings**.
    c. Select the **JDBC** tab.


Home > All resources > 840sqlserver - SQL databases > thingworx (840sqlserver/thingworx) - Connection strings
thingworx (840sqlserver/thingworx) - Connection strings
SQL database
Search (Ctrl+/)
Overview
Activity log
Tags
Diagnose and solve problems
Quick start
Query editor (preview)
Settings
Configure
Geo-Replication
Connection strings
Sync to other databases
Add Azure Search
Properties
Locks
Export template
ADO.NET
JDBC
ODBC
PHP
Download JDBC driver for SQL server

JDBC (SQL authentication)

```
jdbc:sqlserver://840sqlserver.database.windows.net:1433;database=thingworx;user=pradmin840@840sqlserver;password={your_password_here};
encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;
```

    d. Select **Download Microsoft JDBC Driver for SQL Server**.
    e. Select **Microsoft JDBC Driver 6.0 for SQL Server**.
    f. Extract and copy the downloaded binary in your ThingWorx VM to your Tomcat `lib` directory.
11. Start Tomcat.

```
(UBUNTU) sudo service tomcat8.5 start
```

```
(RHEL) $ sudo systemctl start tomcat
```

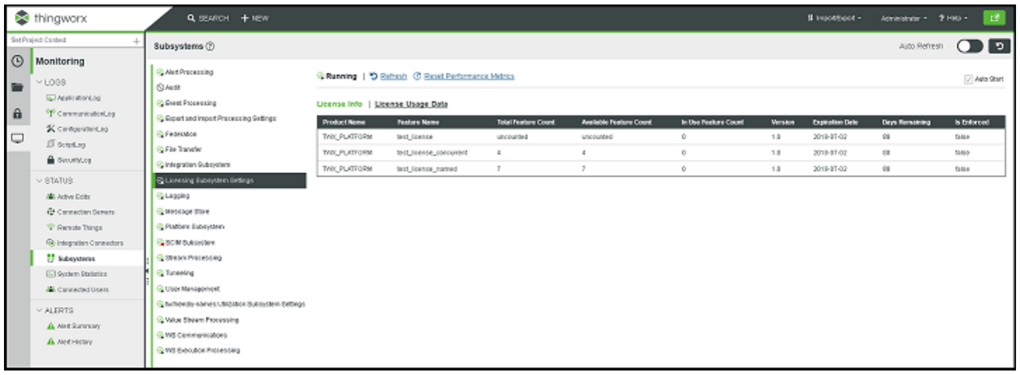
Verify that a license file (`successful_license_capability_response.bin`) is created in the `ThingworxPlatform` folder.

12. To launch ThingWorx, go to `http://<servername>:<port>/Thingworx` in a Web browser.
13. Change the default password:
    a. In Composer, select **Administrator** > **Change Password**.
    b. In the **Change Password** window, enter **Current Password**, **New Password**, and **Confirm Password**.

    Note

    The password, which should not be easily guessed or a known, common password, is recommended to be at least 14 characters in length and should include a mix of uppercase and lowercase letters, numbers, and special characters.

    c. Delete the initial password from the `platform-settings.json` file.
14. Select **Done**.
15. (OPTIONAL STEP) To determine the status of your license, open the **Monitoring**>**Subsystem**>**Licensing Subsystem Settings** in Composer to confirm the list of features (licensed entities) included with the license. If there are no licensed entities present, you are in limited mode.



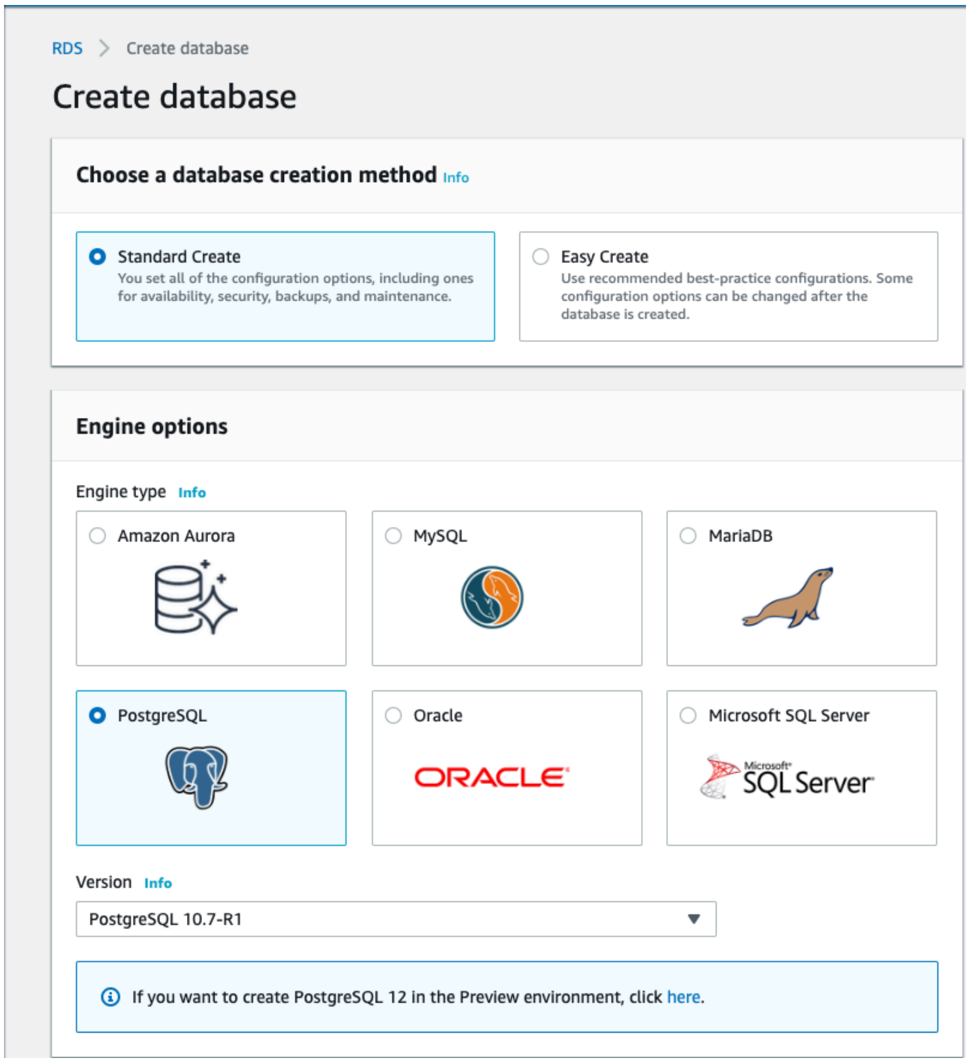
Product Name	Feature Name	Total Feature Count	Available Feature Count	In Use Feature Count	Version	Expiration Date	Days Remaining	Is Enforced
TWX_PLATFORM	test_license	uncounted	uncounted	0	1.0	2019-07-02	98	false
TWX_PLATFORM	test_license_concurrent	4	4	0	1.0	2019-07-02	98	false
TWX_PLATFORM	test_license_named	7	7	0	1.0	2019-07-02	98	false

5

Amazon RDS Installation

Step 1: Install Amazon RDS and Configure the Specify DB Details Page in Amazon Web Services (AWS)

1. Based on your operating system, perform the steps from one of the topics that follow:
   - Install Java and Apache Tomcat (Windows) on page 20
   - Install Java and Apache Tomcat (Ubuntu) on page 40
   - Install Java and Apache Tomcat (RHEL) on page 73
2. Follow the steps outlined in the Amazon RDS installation guide. The steps below provide supplemental guidance when you are ready to configure the **Settings** in AWS.
3. Specify the following information:
   a. In the **Version** field, select the appropriate PostgreSQL version available (10 in this example).



RDS > Create database

Create database

Choose a database creation method Info

[✓ Standard Create] You set all of the configuration options, including ones for availability, security, backups, and maintenance.

Easy Create: Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

Engine options

Engine type Info

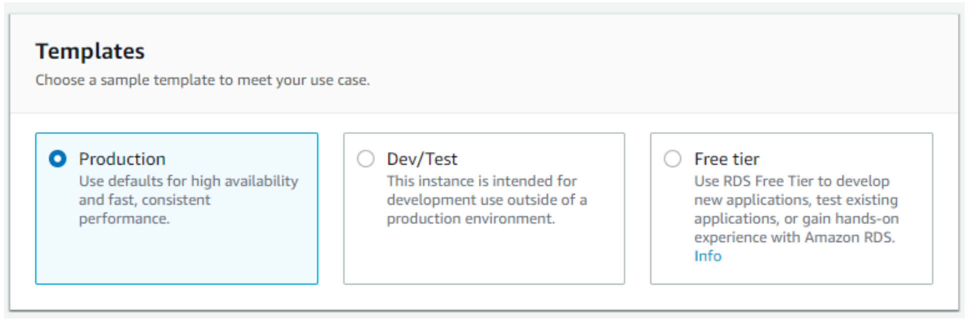
Amazon Aurora | MySQL | MariaDB | [✓ PostgreSQL] | Oracle | Microsoft SQL Server

Version Info

PostgreSQL 10.7-R1

If you want to create PostgreSQL 12 in the Preview environment, click here.

b. Select the type of Template to create.



Templates

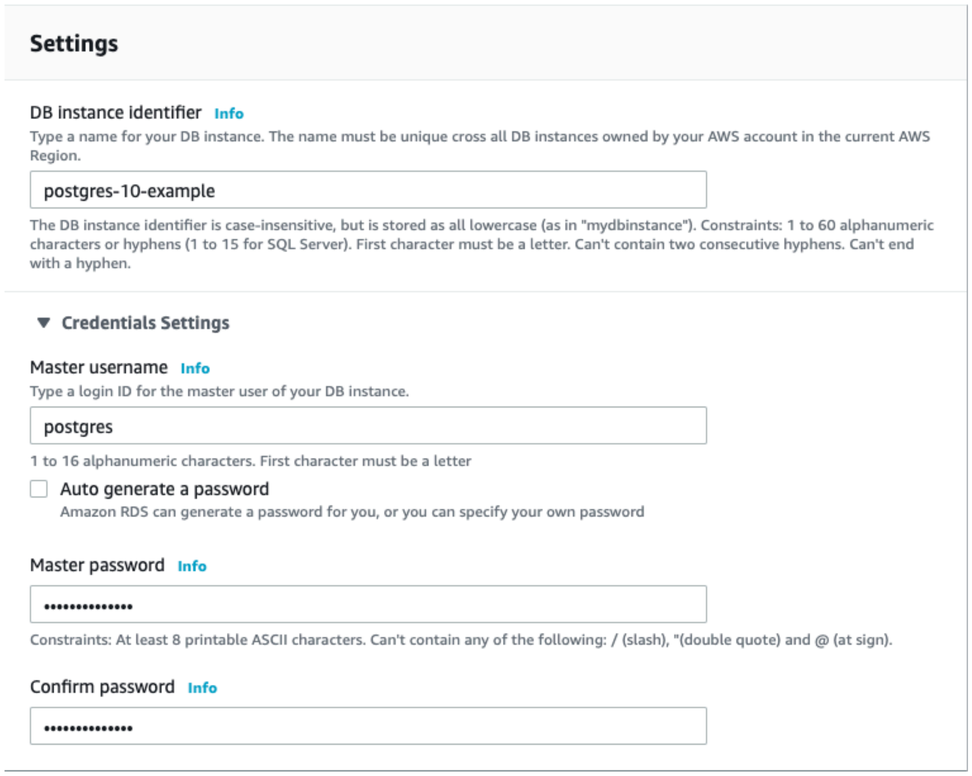
Choose a sample template to meet your use case.

[✓ Production] Use defaults for high availability and fast, consistent performance.

Dev/Test: This instance is intended for development use outside of a production environment.

Free tier: Use RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with Amazon RDS. Info

c. Note the **DB instance identifier** and **Master username** for later use.


Settings

DB instance identifier Info

Type a name for your DB instance. The name must be unique cross all DB instances owned by your AWS account in the current AWS Region.

postgres-10-example

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens (1 to 15 for SQL Server). First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

▼ Credentials Settings

Master username Info

Type a login ID for the master user of your DB instance.

postgres

1 to 16 alphanumeric characters. First character must be a letter

☐ Auto generate a password

Amazon RDS can generate a password for you, or you can specify your own password

Master password Info

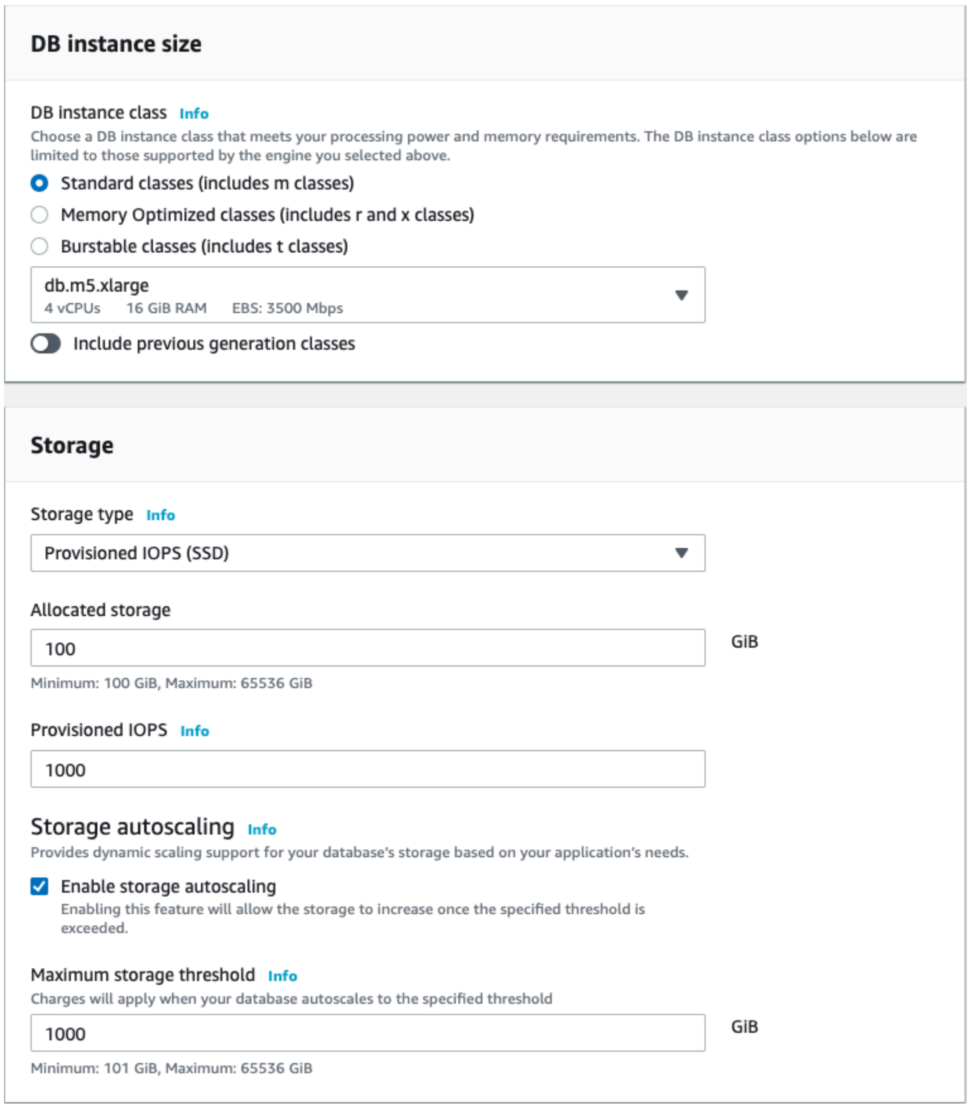
••••••••••••••

Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), "(double quote) and @ (at sign).

Confirm password Info

••••••••••••••

d. In the **DB instance class** section, select the appropriate class. For production, m3.2xlarge is recommended.



DB instance size

DB instance class Info

Choose a DB instance class that meets your processing power and memory requirements. The DB instance class options below are limited to those supported by the engine you selected above.

- [✓ Standard classes (includes m classes)]
- Memory Optimized classes (includes r and x classes)
- Burstable classes (includes t classes)

db.m5.xlarge
4 vCPUs 16 GiB RAM EBS: 3500 Mbps

Include previous generation classes

Storage

Storage type Info

Provisioned IOPS (SSD)

Allocated storage

100 GiB

Minimum: 100 GiB, Maximum: 65536 GiB

Provisioned IOPS Info

1000

Storage autoscaling Info

Provides dynamic scaling support for your database's storage based on your application's needs.

[✓] Enable storage autoscaling
Enabling this feature will allow the storage to increase once the specified threshold is exceeded.

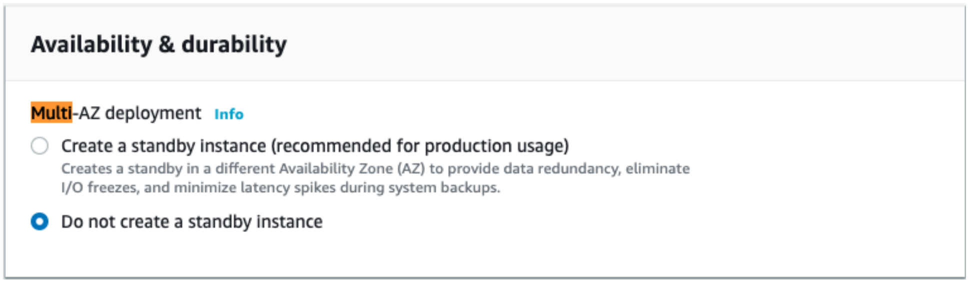
Maximum storage threshold Info

Charges will apply when your database autoscales to the specified threshold

1000 GiB

Minimum: 101 GiB, Maximum: 65536 GiB

e. In the **Mutli-AZ deployment** field, select **Create a standby instance** if you have an HA environment.



Availability & durability

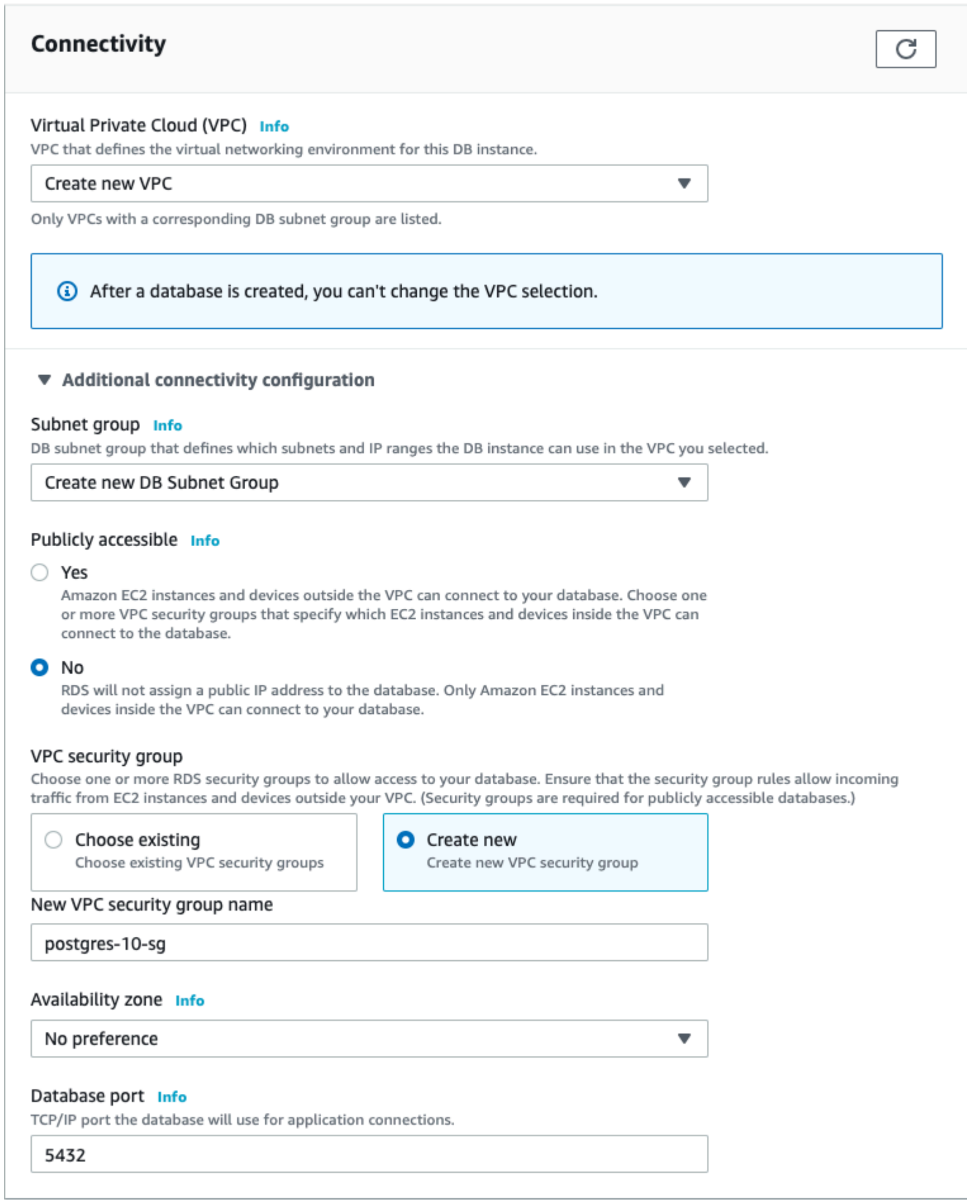
Multi-AZ deployment Info

- Create a standby instance (recommended for production usage)
  Creates a standby in a different Availability Zone (AZ) to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.
- [✓ Do not create a standby instance]

f. In the Connectivity section, select the appropriate options per your organizational needs. The settings should reflect the overall security configuration of the ThingWorx deployment environment (not specific to the database).

Note

The **VPC** and **VPC Security Group(s)** should be created prior to installing the RDS database.



Connectivity

Virtual Private Cloud (VPC) Info
VPC that defines the virtual networking environment for this DB instance.
Create new VPC
Only VPCs with a corresponding DB subnet group are listed.

After a database is created, you can't change the VPC selection.

▼ Additional connectivity configuration

Subnet group Info
DB subnet group that defines which subnets and IP ranges the DB instance can use in the VPC you selected.
Create new DB Subnet Group

Publicly accessible Info
Yes
Amazon EC2 instances and devices outside the VPC can connect to your database. Choose one or more VPC security groups that specify which EC2 instances and devices inside the VPC can connect to the database.
[✓ No]
RDS will not assign a public IP address to the database. Only Amazon EC2 instances and devices inside the VPC can connect to your database.

VPC security group
Choose one or more RDS security groups to allow access to your database. Ensure that the security group rules allow incoming traffic from EC2 instances and devices outside your VPC. (Security groups are required for publicly accessible databases.)
Choose existing — Choose existing VPC security groups
[✓ Create new] — Create new VPC security group

New VPC security group name
postgres-10-sg

Availability zone Info
No preference

Database port Info
TCP/IP port the database will use for application connections.
5432

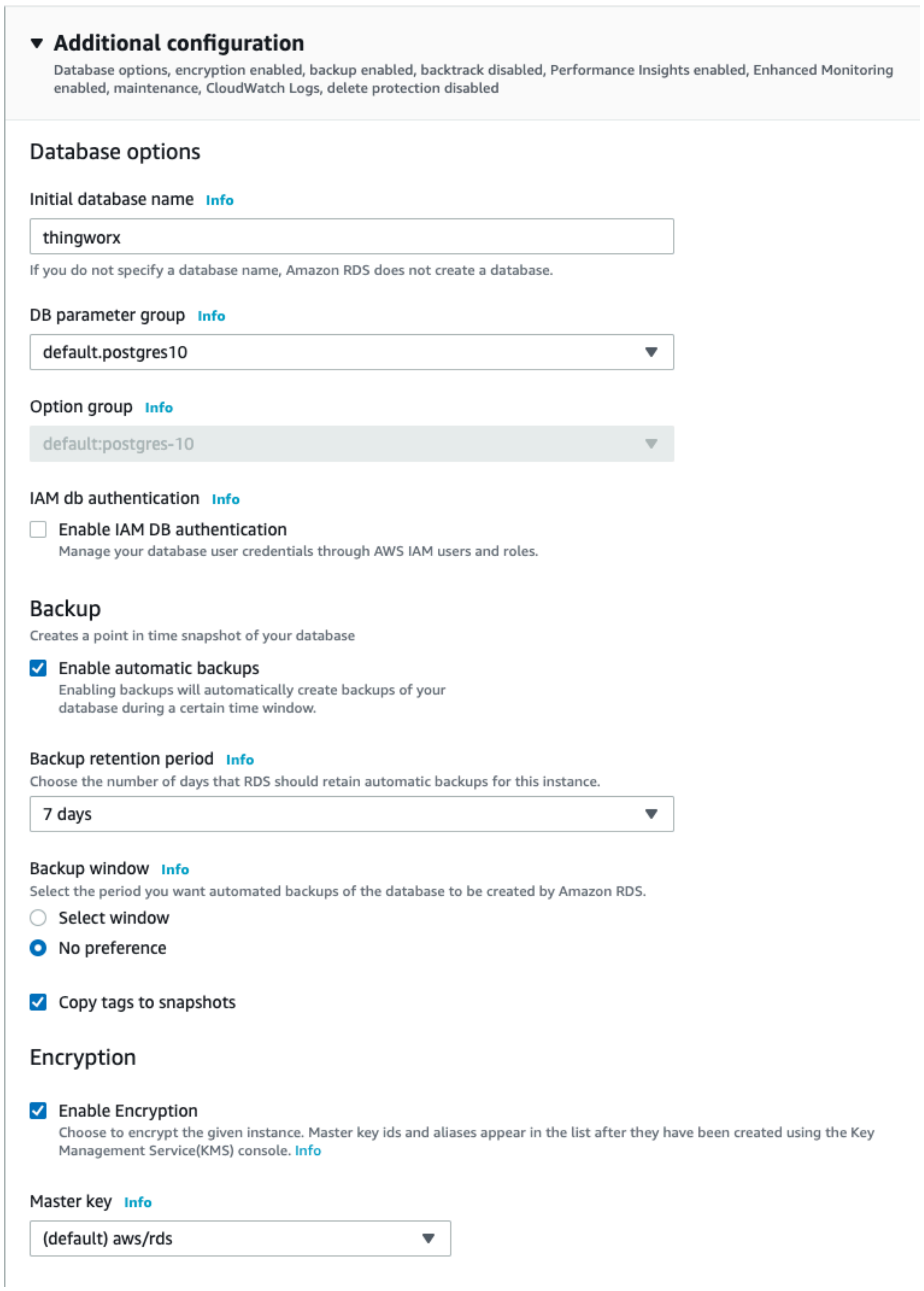
g. In the **Database Options** section, type `thingworx` as the **Database Name**

Note

`thingworx` is the default name that is used in the schema creation scripts.

h. In the **DB Parameter Group** field, select the name of the parameter group created previously.

i. If necessary, select **Enable Encryption** .



▼ Additional configuration

Database options, encryption enabled, backup enabled, backtrack disabled, Performance Insights enabled, Enhanced Monitoring enabled, maintenance, CloudWatch Logs, delete protection disabled

Database options

Initial database name Info

thingworx

If you do not specify a database name, Amazon RDS does not create a database.

DB parameter group Info

default.postgres10

Option group Info

default:postgres-10

IAM db authentication Info

☐ Enable IAM DB authentication

Manage your database user credentials through AWS IAM users and roles.

Backup

Creates a point in time snapshot of your database

☑ Enable automatic backups

Enabling backups will automatically create backups of your database during a certain time window.

Backup retention period Info

Choose the number of days that RDS should retain automatic backups for this instance.

7 days

Backup window Info

Select the period you want automated backups of the database to be created by Amazon RDS.

○ Select window

● No preference

☑ Copy tags to snapshots

Encryption

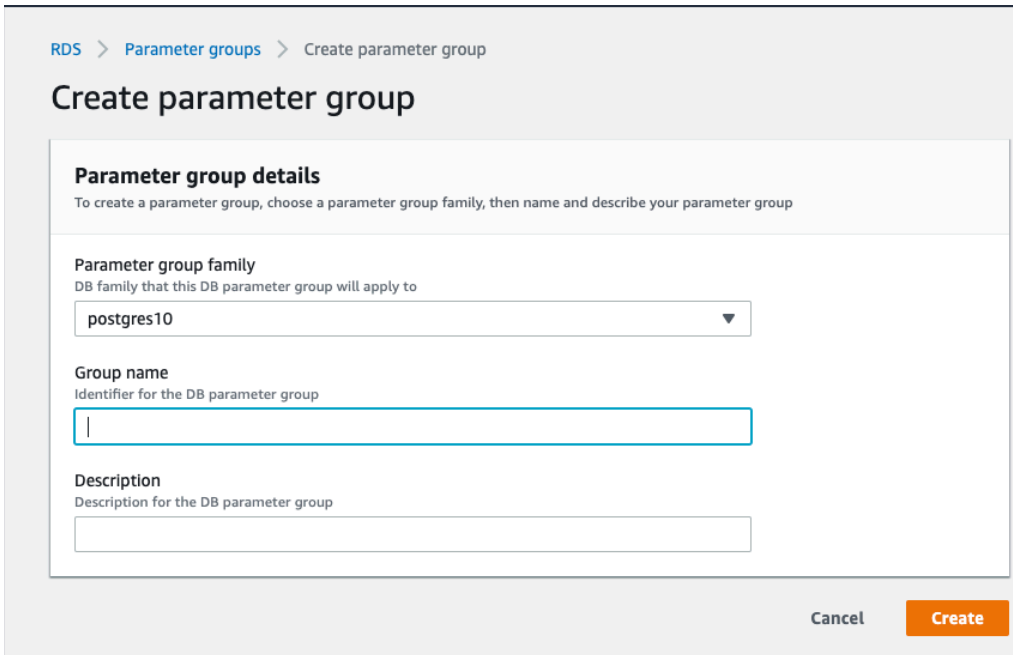
☑ Enable Encryption

Choose to encrypt the given instance. Master key ids and aliases appear in the list after they have been created using the Key Management Service(KMS) console. Info

Master key Info

(default) aws/rds

4. On the RDS Dashboard, click **Parameter groups->Create parameter group**.
5. In the **Parameter group family** field, create a **Group name** and a **Description** for PostgreSQL database configuration.


RDS > Parameter groups > Create parameter group

Create parameter group

Parameter group details

To create a parameter group, choose a parameter group family, then name and describe your parameter group

Parameter group family
DB family that this DB parameter group will apply to
postgres10

Group name
Identifier for the DB parameter group

Description
Description for the DB parameter group

6. On the RDS Dashboard, click **Security Groups**.
7. Create a DB security group to control the DB access later.
8. In the default **DB Security Group**, select the security group that the ThingWorx server will be using to allow access from the ThingWorx server to the database server. This is not the same security group that was created in the previous step. This security group must be created in the EC2 section of AWS with the appropriate inbound/outbound rules to allow the PostgreSQL port to connect to the security group. This security group should also be assigned to the ThingWorx server instance.

Step 2: Configure and Execute the Model/Data Provider Schema Script

To set up the PostgreSQL model/data provider schema, the `thingworxPostgresSchemaSetup` script must be configured and executed. This will set up the public schema under your database on the Amazon RDS PostgreSQL instance.

1. Obtain and open the `thingworxPostgresSchemaSetup` script from the ThingWorx software download package.
2. If necessary, configure the script. Reference the options in the table below.

Note

Ubuntu/RHEL only: The script can be run with the default parameters as:

```
$ sudo sh thingworxPostgresSchemaSetup.sh
```

3. Execute the script.

thingworxPostgresSchemaSetup Script Options

Option	Parameter	Default	Description	Example
-h or -H	server	rds-host	Hostname or IP of RDS PostgreSQL instance.	`-h rds-host`
-p or -P	port	5432	Port number of PostgreSQL.	`-p 5432`
-d or -D	database	thingworx	Database name to use.	`-d thingworx`
-s or -S	schema	public	Schema name to use.	`-s mySchema`

thingworxPostgresSchemaSetup Script Options (continued)

Option	Parameter	Default	Description	Example
-u or -U	username	twadmin	Username to update the database schema	`-u twadmin`
-o or -O	option	all	There are three options: • all: Sets up the model and data provider schemas into the specified database. • model: Sets up the model provider schema into the specified database. • data: Sets up the data provider schema into the specified database.	`-o data`

Step 3: Configure platform-settings.json

1. Create the folder `ThingworxPlatform` at the root of the drive where Tomcat was installed or as a system variable or set a system variable to the location of the folder. Note the following:
   - To specify the location where ThingWorx stores its settings, you can set the THINGWORX_PLATFORM_SETTINGS environment variable to the desired location. Ensure that the folder referenced by THINGWORX_ PLATFORM_SETTINGS exists and is writable by the Tomcat user. This

environment variable should be configured as part of the system environment variables. Ubuntu example: `THINGWORX_PLATFORM_SETTINGS=/data/ThingworxPlatform`

- The ThingWorx server will fail to start if it does not have read and write access to this folder.

```
(UBUNTU command)
$ sudo mkdir /ThingworxPlatform
```

2. Place the `platform-settings.json` file into the `ThingworxPlatform` folder. This file is available in the software download.

```
(UBUNTU/RHEL command)
$ sudo cp platform-settings.json /ThingworxPlatform/
```

3. Open `platform-settings.json` and configure as necessary. Refer to the configuration options in platform-settings.json Configuration Details on page 121.

Note

If your PostgreSQL server is not the same as your ThingWorx server, and you are having issues with your ThingWorx installation, review your Tomcat logs and `platform-settings.json` file. The default installation assumes both servers are on the same machine.

Step 4: Install and Configure PostgreSQL DB Host Servers

The DB host server is the Amazon RDS instance that was created above.

1. Edit the parameter group created earlier to suit your environment. Reference the table below. The values listed in the Configuration column reflect the example deployment in the reference architecture, but can be modified for your environment. For many of the settings in the table below, links are provided to help you determine the configuration values for your environment. RDS specific information can be found at http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Appendix.PostgreSQL.CommonDBATasks.html

Setting	Configuration	Description
shared_buffers	1024MB	Optional performance tuning. Sets the amount of memory the database server uses for shared memory buffers. It is recommended to set this at 1/4 of the memory available on the machine. Refer to https://www.postgresql.org/docs/current/runtime-config-resource.html#RUNTIME-CONFIG-RESOURCE-MEMORY.
work_mem	32MB	Optional performance tuning. Specifies the amount of memory to be used by internal sort operations and hash tables before writing to temporary disk files. Refer to http://www.postgresql.org/docs/current/static/runtime-config-resource.html#GUC-WORK-MEM
maintenance_work_mem	512MB	Optional performance tuning. Specifies the maximum amount of memory to be used by maintenance operations. Refer to http://www.postgresql.org/docs/current/static/runtime-config-resource.html#GUC-MAINTENANCE-WORK-MEM
effective_cache_size		Should be set to an estimate of how much memory is available for disk caching by the OS and within the database. It

Setting	Configuration	Description
		is recommended to set this to half the memory available on the machine.
checkpoint_segments		Depends on the size of the PostgreSQL box. Should set to 32/64/128/256, depending upon machine size.
checkpoint_completion_target		If the **checkpoint_segments** is changed from the default value of 3, change this to 0.9.
ssl_renegotiation_limit		If PostgreSQL is deployed on Ubuntu, set this value to 0 (never) or increase the default (512MB) to something larger, e.g. 2GB to avoid ssl renegotiation from happening too often between master and synchronous slave.

Install ThingWorx

Go to Install ThingWorx on page 16.

6

Installation Appendices

Apache Tomcat Java Option Settings

Mandatory Settings

Setting	Description
`-server`	Explicitly tells the JVM to run in server mode. This is true by default when using 64-bit JDK, but it is best practice to declare it.
`-d64`	Explicitly tells the JVM to run in 64-bit mode. The current JVM automatically detects this, but it is best practice to declare it.
`-XX:+UseG1GC`	Tells the JVM to use the Garbage First Garbage Collector.
`-Dfile.encoding=UTF-8`	Tells the JVM to use UTF-8 as the default character set so that non-Western alphabets are displayed correctly.
`-Djava.library.path`	Specifies the path to the native library.

Mandatory Settings (continued)

Setting	Description
`-Xms3072m` (for a system with 4GB of memory)	Tells the JVM to allocate a minimum of 3072MB of memory to the Tomcat process. This should be set to 75% of the available system memory. **Note** The amount of memory needs to be tuned depending on the actual environment.
`-Xmx3072m` (for a system with 4GB of memory)	Tells the JVM to limit the maximum memory to the Tomcat process. This should be set to 75% of the available system memory. **Note** The amount of memory needs to be tuned depending on the actual environment. 5GB of memory is a good starting point for 100,000 things. **Note** The reason that the min and max amounts of memory are made equal is to reduce JVM having to re-evaluate required memory and resizing the allocation at runtime. While this is recommended for hosted and/or public-facing environments, for development and test environments, using –`Xms512m` would suffice. Also, verify that there is enough memory left to allow the operating system to function.

Optional Settings to Enable JMX Monitoring for VisualVM or JConsole

Setting	Description
`-Dcom.sun.management.jmxremote`	Notifies the JVM that you plan to remote monitor it via JMX
`-Dcom.sun.management.jmxremote.port=22222`	The port the JVM should open up for monitoring.
`-Dcom.sun.management.jmxremote.ssl=false`	No SSL usage.

Optional Settings to Enable JMX Monitoring for VisualVM or JConsole (continued)

Setting	Description
`-Dcom.sun.management.jmxremote.authenticate=false`	No authentication required.
`-Djava.rmi.server.hostname=<host or IP>`	The hostname or IP that the underlying RMI client connection will use.

Encrypting Passwords

See the topic, Security Management Tool, in the ThingWorx Help Center for more information.

Encrypting License and Database Passwords (8.4.0+)

The KeyStore provider makes use of a secure token stored encrypted in a file to work with the KeyStore. All data written to the KeyStore will be stored securely using the password. The first time the provider is started it will generate a random password value and a KeyStore file if they do not already exist.

Note

The KeyStore password and KeyStore file should be restricted to only the application user. The application user must have read/write permissions to the files.

Note

The examples below are Windows-based. Change commands as necessary if you are using a Linux-based OS.

To pre-create a KeyStore file and store initial data in it, you must use the Security Management Tool.

1. Obtain the Security Management Tool zip file from the PTC Support site.
2. Extract the contents of the zip file to a directory.
3. Create a configuration file with the following parameters and place it in the `bin` folder of the unzipped files.

Note

In this example, the file is named `keystore.conf`, the version of the tool is `1.0.0.36`, and it is located at `C://security-common-cli-1.0.0.36/bin.`

Note

The *default-encryption-key-length* must match the application configuration. In ThingWorx, it is the *InternalAesCryptographicKeyLength* parameter located in platform-settings.json on page 121. The default is 128, but you can use 256-bit encryption if you are using Java 1.8.0_162 or higher. If necessary, you can also use with older Java versions by updating the java policy for the key size limit.

```
{
    security {
        secret-provider = "com.thingworx.security.provider.keystore.
KeyStoreProvider"
        default-encryption-key-length = 128

        keystore {
            password-file-path = "/ThingworxPlatform"
            password-file-name = "keystore-password"
            path = "/ThingworxStorage"
            name = "keystore"
        }
    }
}
```

4. Launch a command prompt and go to the location of `security-common-cli-1.0.0.36\bin`.
5. Run the following to create a password file and KeyStore at the configured location:
   - license password:
     ```
     C:\security-common-cli-1.0.0.36\bin> security-common-cli.bat <path to
     keystore>\keystore.conf
     set encrypt.licensing.password "add-password-here"
     ```
   - database password:
     ```
     C:\security-common-cli-1.0.0.36\bin> security-common-cli.bat <path to
     keystore>\keystore.conf
     set encrypt.db.password "add-password-here"
     ```
   - license proxy password:

```
C:\security-common-cli-1.0.0.36\bin> security-common-cli.bat <path to
keystore>\keystore.conf
set encrypt.proxy.password "add-password-here"
```

- RabbitMQ password (if you have installed ThingWorx Flow):

  ```
  C:\security-common-cli-1.0.3.48\bin> security-common-cli.bat <path to
  keystore>\keystore.conf
  set encrypt.queue.password "add-password-here"
  ```

6. Open `ThingworxPlatform\platform-settings.json` and make the following updates:

   - license password: Under `LicensingConnectionSettings`, change the `password` value to `encrypt.licensing.password`. For example, `"password": "encrypt.licensing.password"`
   - database password: Under the `PersistenceProviderPackageConfigs ConnectionInformation` for the persistence provider you are using, change the `password` value to `encrypt.db.password`. For example, `"password": "encrypt.db.password"`
   - license proxy password: Under `LicensingConnectionSettings`, change the password value to `encrypt.proxy.password`. For example, `"password": "encrypt.proxy.password"`
   - RabbitMQ password (if you have installed ThingWorx Flow): In the `platform-settings.json` file, under `OrchestrationSettings`, change the `QueuePassword` value to `encrypt.queue.password`. For example, `"QueuePassword": "encrypt.queue.password"`

   This password signals the ThingWorx platform to look up the encrypted password in the keystore when it is encountered.

Encrypting License and Database Passwords (8.3.x and below)

Refer to older versions of the Installation Guide for this process.

platform-settings.json Configuration Details

The platform-settings.json file is available for administrators to adjust settings for fine-tuning and is available in the software download.

Note

The sample below contains all options. Only one persistence provider is required.

```
{
    "PlatformSettingsConfig": {
        "BasicSettings": {
            "BackupStorage": "/ThingworxBackupStorage",
            "DatabaseLogRetentionPolicy": 7,
            "EnableBackup": true,
            "EnableHA": false,
            "EnableSystemLogging": false,
            "EnableSSO": false,
            "FileRepositoryRoot": "/ThingworxStorage",
            "HTTPRequestHeaderMaxLength": 2000,
            "HTTPRequestParameterMaxLength": 2000,
            "InternalAesCryptographicKeyLength": 128,
            "Storage": "/ThingworxStorage",
            "ScriptTimeout": 30
},
        "SolutionCentralSettings": {
      "SolutionCentralHost": "<Solution Central host name>",
      "KeyStorePath": "<Path for your keystore>",
      "KeyStorePass": "<Password for your keystore>"
        },
        "AdministratorUserSettings": {
            "InitialPassword": "changeme"
        },
        "ContentTypeSettings": {
      "supportedMediaEntityContentTypes" : ["image/svg+xml","image/png","image/
gif","image/bmp","image/jpeg","application/pdf","image/vnd.microsoft.icon"]
},
        "OrchestrationSettings": {
             "EnableOrchestration": true,
             "QueueHost": "localhost",
             "QueuePort": 5672,
             "QueueName": "256mb",
             "QueueUsername": "flowuser",
             "QueuePassword": "encrypt.queue.password",
            "QueueVHost": "orchestration"
     },
       "ExtensionPackageImportPolicy": {
            "importEnabled": false,
            "allowJarResources": false,
            "allowJavascriptResources": false,
            "allowCSSResources": false,
            "allowJSONResources": false,
            "allowWebAppResources": false,
            "allowEntities": false,
            "allowExtensibleEntities": false
       },
       "HASettings": {
           "CoordinatorConnectionTimeout": 15000,
           "CoordinatorHosts": "127.0.0.1:2181",
           "CoordinatorMaxRetries": 3,
           "CoordinatorRetryTimeout": 1000,
           "CoordinatorSessionTimeout": 90000,
           "CoordinatorZNode": "/HALeadershipCoordinator",
           "LoadBalancerBase64EncodedCredentials": "QWRtaW5pc3RyYXRvcjphZG1pbg=="
       },
       "LicensingConnectionSettings": {
           "username": "<username>",
           "password": "<password>",
           "timeout":"60",
           "useProxy": false,
           "proxyHost": "<proxyHost>",
           "proxyPort" : "<proxy port>",
           "proxyScheme": "<http or https>",
           "proxyUseNTLM": true,
           "proxyUsername": "<user>",
           "proxyPassword": "<user password>",
           "proxyWorkstation": "<dummyWorkstation>",
           "proxyDomain": "<dummyDomain>"
       }
   },
   "PersistenceProviderPackageConfigs": {
       "NeoPersistenceProviderPackage": {
           "StreamProcessorSettings": {
               "maximumBlockSize": 2500,
               "maximumQueueSize": 250000,
               "maximumWaitTime": 10000,
               "scanRate": 5,
               "sizeThreshold": 1000
           },
           "ValueStreamProcessorSettings": {
            "maximumBlockSize": 2500,
            "maximumQueueSize": 500000,
            "maximumWaitTime": 10000,
            "scanRate": 5,
            "sizeThreshold": 1000
        },
        "PersistentPropertyProcessorSettings": {
            "maximumBlockSize": 2500,
            "maximumWaitTime": 1000,
            "maximumQueueSize": 100000,
            "numberOfProcessingThreads": 20,
            "scanRate": 25,
            "sizeThreshold": 1000
        }
    },
    "H2PersistenceProviderPackage": {
        "ConnectionInformation": {
            "acquireIncrement": 5,
            "acquireRetryAttempts": 30,
            "acquireRetryDelay": 1000,
            "checkoutTimeout": 2000,
            "idleConnectionTestPeriod": 6,
            "initialPoolSize": 10,
            "maxConnectionAge": 0,
            "maxIdleTime": 0,
            "maxIdleTimeExcessConnections": 36000,
            "maxPoolSize": 100,
            "maxStatements": 0,
            "maxStatementsPerConnection": 50,
            "minPoolSize": 10,
            "numHelperThreads": 6,
            "password": "password",
            "tableLockTimeout": 10000,
            "testConnectionOnCheckout": false,
            "unreturnedConnectionTimeout": 0,
            "username": "twadmin"
        },
        "StreamProcessorSettings": {
            "maximumBlockSize": 2500,
            "maximumQueueSize": 250000,
            "maximumWaitTime": 10000,
            "numberOfProcessingThreads": 5,
            "scanRate": 5,
            "sizeThreshold": 1000
        },
        "ValueStreamProcessorSettings": {
                "maximumBlockSize": 2500,
                "maximumWaitTime": 10000,
                "maximumQueueSize": 500000,
                "numberOfProcessingThreads": 5,
                "scanRate": 5,
                "sizeThreshold": 1000
            },
            "PersistentPropertyProcessorSettings": {
                "maximumBlockSize": 2500,
                "maximumWaitTime": 1000,
                "maximumQueueSize": 100000,
                "numberOfProcessingThreads": 20,
                "scanRate": 25,
                "sizeThreshold": 1000
            }
        },
        "PostgresPersistenceProviderPackage": {
            "ConnectionInformation": {
                "acquireIncrement": 5,
                "acquireRetryAttempts": 3,
                "acquireRetryDelay": 10000,
                "checkoutTimeout": 1000000,
                "driverClass": "org.postgresql.Driver",
                "fetchSize": 5000,
                "idleConnectionTestPeriod": 60,
                "initialPoolSize": 5,
                "jdbcUrl": "jdbc:postgresql://localhost:5432/thingworx",
                "maxConnectionAge": 0,
                "maxIdleTime": 0,
                "maxIdleTimeExcessConnections": 300,
                "maxPoolSize": 100,
                "maxStatements": 100,
                "minPoolSize": 5,
                "numHelperThreads": 8,
                "password": "password",
                "testConnectionOnCheckout": false,
                "unreturnedConnectionTimeout": 0,
                "username": "twadmin"
            },
            "StreamProcessorSettings": {
                "maximumBlockSize": 2500,
                "maximumQueueSize": 250000,
                "maximumWaitTime": 10000,
                "numberOfProcessingThreads": 5,
                "scanRate": 5,
                "sizeThreshold": 1000
            },
            "ValueStreamProcessorSettings": {
                "maximumBlockSize": 2500,
                "maximumQueueSize": 500000,
                "maximumWaitTime": 10000,
                "numberOfProcessingThreads": 5,
                "scanRate": 5,
                "sizeThreshold": 1000
            },
            "PersistentPropertyProcessorSettings": {
                "maximumBlockSize": 2500,
                "maximumWaitTime": 1000,
                "maximumQueueSize": 100000,
                "numberOfProcessingThreads": 20,
                "scanRate": 25,
                "sizeThreshold": 1000
            }
        },
        "MssqlPersistenceProviderPackage": {
            "ConnectionInformation": {
                "acquireIncrement": 5,
                "acquireRetryAttempts": 3,
                "acquireRetryDelay": 10000,
                "checkoutTimeout": 1000000,
                "driverClass": "com.microsoft.sqlserver.jdbc.SQLServerDriver",
                "fetchSize": 5000,
                "idleConnectionTestPeriod": 60,
                "initialPoolSize": 5,
                "jdbcUrl": "jdbc:sqlserver://localhost:1433;databaseName=
thingworx;applicationName=Thingworx;",
                "maxConnectionAge": 0,
                "maxIdleTime": 0,
                "maxIdleTimeExcessConnections": 300,
                "maxPoolSize": 100,
                "maxStatements": 100,
                "minPoolSize": 5,
                "numHelperThreads": 8,
                "password": "Password@123",
                "testConnectionOnCheckout": false,
                "unreturnedConnectionTimeout": 0,
                "username": "msadmin"
            },
            "StreamProcessorSettings": {
                "maximumBlockSize": 2500,
                "maximumQueueSize": 250000,
                "maximumWaitTime": 10000,
                "numberOfProcessingThreads": 5,
                "scanRate": 5,
                "sizeThreshold": 1000
            },
            "ValueStreamProcessorSettings": {
                "maximumBlockSize": 2500,
                "maximumWaitTime": 10000,
                "maximumQueueSize": 500000,
                "numberOfProcessingThreads": 5,
                "scanRate": 5,
                "sizeThreshold": 1000
            },
            "PersistentPropertyProcessorSettings": {
                "maximumBlockSize": 2500,
                "maximumWaitTime": 1000,
                "maximumQueueSize": 100000,
                "numberOfProcessingThreads": 20,
                "scanRate": 25,
                "sizeThreshold": 1000
            }
        }
    }
  }
```

platform-settings.json Options

For all databases listed below, the following guidelines should be followed for value stream processor settings and stream processor settings :

```
StreamProcessorSettings.numberOfProcessingThreads +
ValueStreamProcessorSettings.numberOfProcessingThreads < 50% of DB connection pool
And
ValueStreamProcessorSettings.numberOfProcessingThreads >=
StreamProcessorSettings.numberOfProcessingThreads
```

Basic Settings

Setting	Default	Description
BackupStorage	/ThingworxBackupStorage	The directory name where all backups are written to.
DatabaseLogRetentionPolicy	7	The number of days that database logs are retained.
EnableBackup	true	Determines whether backups are retained.
EnableHA	false	Determines whether

Basic Settings (continued)

Setting	Default	Description
		ThingWorx can be configured for a highly available landscape.
EnableSystemLogging	false	Determines whether system logging is enabled. **Note** Do not turn this on unless instructed by ThingWorx Support.
EnableSSO	false	Set to true to enable SSO for ThingWorx Platform. When SSO is enabled, all authentication is redirected to the central authorization server that is configured in the `sso-settings.json` file. Edge websocket authentication is not affected.
FileRepositoryRoot	/ThingworxStorage	The directory where the root file repository is created. The default location is sufficient for standalone deployments. For ThingWorx HA deployments, the repository should be located on a shared file system where all ThingWorx servers have access.
HTTPRequestHeader-MaxLength	2000	The maximum allowable length for HTTP Request Headers values.
HTTPRequestParame-terMaxLength	2000	The maximum allowable length for HTTP Request Parameter values.

Basic Settings (continued)

Setting	Default	Description
InternalAesCryptographicKeyLength	128	Key length used when generating a symmetric AES key. Supported values are 128, 192, and 256. **Note** Encryption and decryption will fail if the key length is higher than 128 and the Java policies are not configured to use that key size.
Storage	/ThingworxStorage	The directory where all storage directories are created/located (excluding Backup Storage).
ScriptTimeout	30	The maximum number of seconds that a script may execute before the ThingWorx platform terminates the script. **Note** Scripts on the platform are terminated automatically if the script executes for longer than the number of seconds configured for this timeout. **Note** Please consider the sensitivity and/or importance of the information handled by scripts when configuring their timeout duration.

Basic Settings (continued)

Setting	Default	Description
		Although it is important to terminate scripts after a given period of time for security reasons, doing so prematurely can lead to a loss of data. Due to the flexibility of the ThingWorx platform, there are use cases that could potentially require timeout periods shorter or longer than the default.

Solution Central Settings

Setting	Default	Description
SolutionCentralHost	sc.thingworx.com	Solution Central host name.
SolutionCentralPort	443	Solution Central port details. **Note** Do not set **SolutionCentralPort** if it has a default value.
KeyStorePath	/ThingworxPlatform/sc-keystore	Path for your keystore.
KeyStorePass	None	Password for your keystore. **Note** For encrypting your password, set **KeyStorePass** to `encrypt.sc.password`.

ThingWorx Flow Settings

These properties are applicable only if ThingWorx Flow is installed on ThingWorx Foundation. These values must be defined, both in ThingWorx Flow and RabbitMQ configuration files and must not be edited unless changed across both applications.

Setting	Default	Description
EnableOrchestration	`true`	Indicates if ThingWorx Flow is enabled in ThingWorx Foundation.
QueueHost	`localhost`	RabbitMQ host name
QueuePort	`5672`	RabbitMQ port
QueueName	`256mb`	RabbitMQ queue name
QueueUsername	`flowuser`	RabbitMQ queue username
QueuePassword	`encrypt.queue.password`	RabbitMQ queue password **Note** The password is automatically encrypted while installing ThingWorx Flow on the same machine as ThingWorx Foundation. If ThingWorx Flow is installed on a different machine than ThingWorx Foundation, you need to encrypt this password on page 118.
QueueVHost	`orchestration`	RabbitMQ VHost name

Extension Package Import Policy

Extension import is disabled by default for all users. Use the following settings to configure extension import functionality. Reference Importing Extensions for more information.

Setting	Description	Default	Examples
importEnabled	The top level control that represents the ability to import (=true) or not import (=false) extensions.	false	• `"importEnabled": false` - Extensions cannot be imported, even if other **ExtensionPackageImportPolicy** settings are set to true. • `"importEnabled": true` - Passes the extension import to the next set of **allow<Content>Resources** settings (see rows below). **Note** If the **allow<Content>Resources** settings are false, then an empty extension (no entities, extensible entities, or resources) can be imported. Since this is likely not a useful configuration, if **importEnabled** is set to true, at least one other

Setting	Description	Default	Examples
			allow<Content>Resources setting should also be set to true.
allowJarResources	Allows extensions with Jar Resources to be imported.	false	• `"allowJarResources": true` - allows extensions that declare jar files in their manifest as jar resources to be imported. • `"allowJarResources": false` -will not allow extensions that declare jar files in their manifest as jar resources to be imported.
allowJavascriptResources	Allows extensions with JavaScript Resources to be imported.	false	• `"allowJavascriptResources": true` -allows extensions that declare JavaScript UI file resources of JS type in their manifest as JavaScript resources to be imported. • `"allowJavascriptResources": false` - will not allow

Setting	Description	Default	Examples
			extensions that declare JavaScript UI file resources of JS type in their manifest as JavaScript resources to be imported.
allowCSSResources	Allows extensions with CSS Resources to be imported.	false	• `"allow-CSSResources": true` -allows extensions that contain CSS UI file resources to be imported. • `"allowCSSResources": false` -will not allow extensions that contain CSS UI file resources to be imported.
allowJSONResources	Allows extensions with JSON Resources (for example, localization files) to be imported.	false	• `"allow-JSONResources": true` - allows extensions that contain JSON UI file resources to be imported. • `"allowJSONResources": false` - will not allow extensions that contain JSON UI file resources to be

Setting	Description	Default	Examples
			imported.
allowWebAppResources	Allows extensions with Web Resources to be imported.	false	• `"allowWebAppResources": true` - allows extensions that contain WebApp UI file resources to be imported. • `"allowWebAppResources": false` - will not allow extensions that contain WebApp UI file resources to be imported.

Setting	Description	Default	Examples
allowEntities	Allows extensions with non-extensible entities to be imported. Examples of non-extensible entities include: • Application Key • Authenticator • Dashboard • Data Analysis Definition • Data Shape • GenericContentEntity and derived child classes like State Definition, Style Definition, Style Theme • Group • Localization Table • Log • Mashup • Media Entity • Menu • ModeledServiceProviderEntity and child classes like Notification Content • Network • Notification Definition • Organization	false	• `"allowEntities": true` - allows extensions that declare non-extensible entities in their manifest to be imported. • `"allowEntities": false` - will not allow extensions that declare non-extensible entities in their manifest to be imported.

Setting	Description	Default	Examples
	• Persistence Provider • PersistenceProviderPackage and derived child classes • Project • Thing Shape • Thing Template • User • Vocabulary and derived child classes like DataTagVocabulary, ModelTagVocabulary		
allowExtensibleEntities	Allows extensions with non-extensible entities to be imported. Examples of extensible entities include: • DirectoryService and derived child classes • ExtensionPackage • Resource and derived child classes that contain custom functions/ services used as resources similar to OOTB	false	• `"allowExtensibleEntities": true` - allows extensions that declare extensible entities in their manifest to be imported. • `"allowExtensibleEntities": false` - will not allow extensions that declare extensible entities in their manifest to be imported.

Setting	Description	Default	Examples
	Resources such as InfoTableFunctions, EntityServices, and EncryptionServices. **Note** The OOTB Subsystems that are not part of extensions are not affected. • ScriptFunctionLibrary and derived child classes. • Subsystem and derived child classes. **Note** The OOTB Subsystems that are not part of extensions are not affected. • Thing Package • Widget		

HA Settings

Settings specific to a PostgreSQL HA landscape configuration. All settings are ignored if the **EnableHA** setting above is set to false. **CoordinatorHosts** and **LoadBalancerBase64EncodedCredentials** must be modified to suit your environment.

Setting	Default	Description
CoordinatorConnection-Timeout	15000	How long to wait (in milliseconds) for a connection to be established with Apache ZooKeeper service used to coordinate ThingWorx leadership.
CoordinatorHosts	127.0.0.1:2181	A comma-delimited list of the Apache ZooKeeper servers used to coordinate ThingWorx leader election. String pattern is IP:port. (e.g. "127.0.0.1:2181, 127.0.0.2:2181").
CoordinatorMaxRetries	3	The maximum allowable number of retries that will be made to establish a connection with the Apache ZooKeeper service used to coordinate ThingWorx leadership.
CoordinatorRetryTime-out	1000	How long to wait (in milliseconds) for each retry attempt.
CoordinatorSessionTi-meout	90000	How long ThingWorx waits (in milliseconds) without receiving a "heartbeat" from the Apache ZooKeeper service used to coordinate ThingWorx leadership.

Setting	Default	Description
CoordinatorZNode	/HALeadershipCoordinator	When one Apache ZooKeeper service is shared by multiple ThingWorx HA deployments, this setting must provide a unique value for each ThingWorx HA deployment. This setting's value can be arbitrary but must follow the format `/<anyTextHere>`. For example, ThingWorx instances TWX1 and TWX2 are in HA system A, and ThingWorx instances TWX3 and TWX4 are in HA system B. CoordinatorZNode is set to `/HAsystemA` for TWX1 and TWX2, and it is set to `/HAsystemB` for TWX3 and TWX4.
LoadBalancerBase64EncodedCredentials	QWRtaW5pc3RyYXRvcjphZG1pbg==	The Base64-encoded credentials for the HA Load Balancer, in the format of `<user>:<unique password>` **Note** Do not use a ThingWorx user. **Note** You can use any utility that Base64 encodes the matching `<user>:<unique password>` string used in your load balancer

Setting	Default	Description
		setup.

Administrator User Settings

Setting	Default	Description
InitialPassword	n/a	The initial Administrator password that is required to log into ThingWorx for the first time. The minimum length can be configured in the User Management Subsystem (minimum 10 characters, default is 14 characters). See Passwords for more information.

Content Type Settings

Setting	Default	Description
supportedMediaEntity-ContentTypes	`"image/svg+xml","image/png","image/gif","image/bmp","image/jpeg","application/pdf","image/vnd.microsoft.icon"`	Comma-delimited list of valid MIME content types that can be dynamically linked to Media entities. Additional types can be added. **Note** If the content type that is coming from a different server not a supported media entity type, then the content is downloaded as a file on the client machine instead of streamed with the media entity.

Licensing Connection Settings

Setting	Default	Description
username	n/a	PTC Support site username
password	n/a	PTC Support site

Licensing Connection Settings (continued)

Setting	Default	Description
		password
timeout (in seconds)	60	After the timeout period, the following error is logged in the Application Log: **License Server could not process request**
useProxy	false	Enables proxy settings for licensing. If true, proxy settings are used to connect to the licensing server.
proxyHost		The name of the proxy host.
proxyPort		The port number of the proxy host.
proxyScheme	http	`http` or `https`.
proxyUsername		The username for authentication if the proxy server connection requires authentication.
proxyPassword		The password for authentication if the proxy server connection requires authentication. **Note** See Encrypting Passwords on page 118 for information on encrypting this value.
proxyUseNTLM	false	Option to use the NTLM protocol.

Licensing Connection Settings (continued)

Setting	Default	Description
proxyWorkstation		The name of the user's computer on the network, if NTLM authentication is required.
proxyDomain		The name of the user's domain, if NTLM authentication is required.

NeoPersistenceProviderPackage

Contains Neo4j-specific Persistence Provider settings. If Neo4j is not the Persistence Provider, this entire section should be ignored.

Setting	Default	Description
StreamProcessorSettings		
maximumBlockSize	2500	The maximum number of stream writes to process in one block.
maximumQueueSize	250000	The maximum number of stream entries to queue (will be rejected after that).
maximumWaitTime	10000	The maximum wait time (in milliseconds) before flushing stream buffer.
scanRate	5	The rate (in milliseconds) at which to check the buffer status.
sizeThreshold	1000	The maximum number of items to accumulate before flushing stream buffer.
ValueStreamProcessorSettings		
maximumBlockSize	2500	The maximum number of stream writes to process in one block.
maximumQueueSize	500000	The maximum number of stream entries to queue (will be rejected after that).

(continued)

Setting	Default	Description
maximumWaitTime	10000	The maximum wait time (in milliseconds) before flushing the stream buffer.
scanRate	5	The rate (in milliseconds) at which to check the buffer status.
sizeThreshold	1000	The maximum number of items to accumulate before flushing stream buffer.
PersistentPropertyProcessorSettings		
maximumBlockSize	2500	The maximum number of property writes to process in one block.
maximumWaitTime	1000	The maximum wait time (in milliseconds) before flushing the property buffer.
maximumQueueSize	100000	The maximum number of property entries to queue (will be rejected after that).
numberOfProces-singThreads	20	The number of threads to use when processing properties.
scanRate	25	The rate (in milliseconds) at which to check the buffer status.
sizeThreshold	1000	The maximum number of items to accumulate before flushing the property buffer.

H2PersistenceProviderPackage

Setting	Default	Description
Connection Information		
acquireIncrement	5	Determines how many

H2PersistenceProviderPackage (continued)

Setting	Default	Description
		connections at a time the ThingWorx will try to acquire when the pool is exhausted.
acquireRetryAttempts	30	Defines how many times ThingWorx will try to acquire a new connection from the database before giving up.
acquireRetryDelay	1000	The time (in milliseconds) ThingWorx will wait between acquire attempts.
checkoutTimeout	1000000	The number of milliseconds a client calling **getConnection()** will wait for a connection to be checked-in or acquired when the pool is exhausted.
idleConnectionTestPer-iod	6	Time period (in seconds) where connections will be tested so that idle connections won't be closed from outside processes such as firewalls, etc. If this is a number greater than 0, ThingWorx will test all idle, pooled but unchecked-out connections, every x

H2PersistenceProviderPackage (continued)

Setting	Default	Description
		number of seconds. **Note** If you are experiencing "No connection to model provider" errors, review this setting. Compare to firewall defaults. Lowering the default will alleviate disconnection issues.
initialPoolSize	10	Initial number of database connections created and maintained within a pool upon startup. Should be between **minPoolSize** and **maxPoolSize**.
maxConnectionAge	0	Seconds, effectively a time to live. A connection older than **maxConnectionAge** will be destroyed and purged from the pool.
maxIdleTime	0	Seconds a connection can remain pooled but unused before being discarded. Zero means idle connections never expire.
maxIdleTimeExcess-Connections	36000	The number of seconds that connections in excess of **minPoolSize** are permitted to remain in idle in the pool before being culled. Intended for applications that wish to aggressively minimize the number of open connections, shrinking the pool back towards

H2PersistenceProviderPackage (continued)

Setting	Default	Description
		minPoolSize if, following a spike, the load level diminishes and connections acquired are no longer needed. If **maxIdleTime** is set, **maxIdleTimeExcess-Connections** should be smaller to have any effect. Setting this to zero means no enforcement and excess connections are not idled out.
maxPoolSize	100	Maximum number of connections a pool will maintain at any given time.
maxStatements	0	The size of the ThingWorx global PreparedStatement cache.
maxStatementsPerCon-nection	50	The size of the ThingWorx global PreparedStatement cache for each connection.
minPoolSize	5	Minimum number of connections a pool will maintain at any given time.
numHelperThreads	6	The number of helper threads to spawn. Slow JDBC operations are generally performed by helper threads that don't hold contended locks. Spreading these operations over multiple threads can significantly improve performance by allowing multiple

H2PersistenceProviderPackage (continued)

Setting	Default	Description
		operations to be performed simultaneously.
password	n/a	Database password.
username	twadmin	Database username.
tableLockTimeout	10000	The number of milliseconds a client will wait for a database table to be unlocked.
testConnectionOn-Checkout	false	If true, an operation will be performed at every connection checkout to verify that the connection is valid.
unreturnedConnection-Timeout	0	The number of seconds to wait for a response from an unresponsive connection before discarding it. If set, if an application checks out but then fails to check-in a connection within the specified period of time, the pool will discard the connection. This permits applications with occasional connection leaks to survive, rather than eventually exhausting the connection pool. Zero means no timeout, and applications are expected to close their own connections.
StreamProcessorSettings		
maximumBlockSize	2500	The maximum number of stream writes to process in one block.
maximumQueueSize	250000	The maximum number of

H2PersistenceProviderPackage (continued)

Setting	Default	Description
		stream entries to queue (will be rejected after that).
maximumWaitTime	10000	The maximum wait time (in milliseconds) before flushing stream buffer.
numberOfProces-singThreads	5	The number of threads to use when processing properties.
scanRate	5	The rate (in milliseconds) at which to check the buffer status.
sizeThreshold	1000	The maximum number of items to accumulate before flushing stream buffer.
ValueStreamProcessorSettings		
maximumBlockSize	2500	The maximum number of stream writes to process in one block.
maximumQueueSize	250000	The maximum number of stream entries to queue (will be rejected after that).
maximumWaitTime	10000	The maximum wait time (in milliseconds) before flushing the stream buffer.
numberOfProces-singThreads	5	The number of threads to use when processing properties.
scanRate	5	The rate (in milliseconds) at which to check the buffer status.
sizeThreshold	1000	The maximum number of items to accumulate before flushing stream buffer.
PersistentPropertyProcessorSettings		

H2PersistenceProviderPackage (continued)

Setting	Default	Description
maximumBlockSize	2500	The maximum number of property writes to process in one block.
maximumWaitTime	1000	The maximum wait time (in milliseconds) before flushing the property buffer.
maximumQueueSize	100000	The maximum number of property entries to queue (will be rejected after that).
numberOfProces-singThreads	20	The number of threads to use when processing properties.
scanRate	25	The rate (in milliseconds) at which to check the buffer status.
sizeThreshold	1000	The maximum number of items to accumulate before flushing the property buffer.

PostgresPersistenceProviderPackage

Setting	Default	Description
ConnectionInformation		
acquireIncrement	5	Determines how many connections at a time the platform will try to acquire when the pool is exhausted.
acquireRetryAttempts	3	Defines how many times ThingWorx will try to acquire a new connection from the database before giving up.
acquireRetryDelay	10000	The time (in milliseconds) ThingWorx will wait between acquire attempts.

PostgresPersistenceProviderPackage (continued)

Setting	Default	Description
checkoutTimeout	10000000	The number of milliseconds a client calling **getConnection()** will wait for a connection to be checked-in or acquired when the pool is exhausted.
driverClass	org.postgresql.Driver	The fully-qualified class name of the JDBC driverClass that is expected to provide Connections.
fetchSize	5000	The count of rows to be fetched in batches instead of caching all rows on the client side.
idleConnectionTestPer-iod	60	If this is a number greater than 0, ThingWorx will test all idle, pooled but unchecked-out connections, every x number of seconds.
initialPoolSize	5	Initial number of database connections created and maintained within a pool upon startup. Should be between **minPoolSize** and **maxPoolSize**.
jdbcUrl	jdbc:postgresql://localhost:5432/thingworx	The jdbc url used to

PostgresPersistenceProviderPackage (continued)

Setting	Default	Description
		connect to PostgreSQL. **Note** If the default schema name is changed (from public), you must add `<databasename>?currentSchema=<name of schema>`. For example, if the schema name is `mySchema`, it would be: `jdbc:postgresql://`*`<DBServer>`*`:`*`<DBPort>`*`/`*`<databasename>`*`?currentSchema=mySchema` **Note** If you are configuring an HA solution, this should reflect the server IP that the pgPool process is running on. Change the port to the port that pgPool is serving.
maxConnectionAge	0	Seconds, effectively a time to live. A Connection older than **maxConnectionAge** will be destroyed and purged from the pool.
maxIdleTime	0	Seconds a connection can remain pooled but unused before being discarded. Zero means idle connections never expire.
maxIdleTimeExcess-Connections	300	The number of seconds that connections in excess of **minPoolSize** are

PostgresPersistenceProviderPackage (continued)

Setting	Default	Description
		permitted to remain in idle in the pool before being culled. Intended for applications that wish to aggressively minimize the number of open connections, shrinking the pool back towards **minPoolSize** if, following a spike, the load level diminishes and connections acquired are no longer needed. If **maxIdleTime** is set, **maxIdleTimeExcess-Connections** should be smaller to have any effect. Setting this to zero means no enforcement and excess connections are not idled out.
maxPoolSize	100	Maximum number of connections a pool will maintain at any given time.
maxStatements	100	The size of ThingWorx's global PreparedStatement cache.
minPoolSize	5	Minimum number of Connections a pool will maintain at any given time.
numHelperThreads	8	The number of helper threads to spawn. Slow JDBC operations are generally performed by helper threads that don't hold contended locks. Spreading these operations over multiple threads can significantly improve performance by allowing multiple

PostgresPersistenceProviderPackage (continued)

Setting	Default	Description
		operations to be performed simultaneously.
password	*<unique password>*	The password used to log into the database.
testConnectionOn-Checkout	false	If true, an operation will be performed at every connection checkout to verify that the connection is valid.
unreturnedConnection-Timeout	0	The number of seconds to wait for a response from an unresponsive connection before discarding it. If set, if an application checks out but then fails to check-in a connection within the specified period of time, the pool will discard the connection. This permits applications with occasional connection leaks to survive, rather than eventually exhausting the Connection pool. Zero means no timeout, and applications are expected to close their own connections.
username	twadmin	The user that has the privilege to modify tables. This is the user created on the database for the ThingWorx server.

PostgresPersistenceProviderPackage (continued)

Setting	Default	Description
		Note To change the PostgreSQL password: Change this user's password, and also change the unencrypted password setting in the `platform-settings.json` file or the encrypted value in the `/ThingworxStorage/keystore.jks` keystore.
Stream Processor Settings		
maximumBlockSize	2500	The maximum number of stream writes to process in one block.
maximumQueueSize	250000	The maximum number of stream entries to queue (will be rejected after that).
maximumWaitTime	10000	Number of milliseconds the system waits before flushing the stream buffer.
numberOfProcessingThreads	5	The number of processing threads.
scanRate	5	The buffer status is checked at the specified rate value in milliseconds.
sizeThreshold	1000	Maximum number of items to accumulate before flushing the stream buffer.
Value Stream Processor Settings		
maximumBlockSize	2500	Maximum number of value stream writes to process in one block.
maximumQueueSize	500000	Maximum number of value stream entries to

PostgresPersistenceProviderPackage (continued)

Setting	Default	Description
		queue (will be rejected after that).
maximumWaitTime	10000	Number of milliseconds the system waits before flushing the value stream buffer.
numberofProcessingTh-reads	5	The number of processing threads.
scanRate	5	The rate (in milliseconds) before flushing the stream buffer.
sizeThreshold	1000	Maximum number of items to accumulate before flushing the value stream buffer.
PersistentPropertyProcessorSettings		
maximumBlockSize	2500	The maximum number of property writes to process in one block.
maximumWaitTime	1000	The maximum wait time (in milliseconds) before flushing the property buffer.
maximumQueueSize	100000	The maximum number of property entries to queue (will be rejected after that).
numberOfProces-singThreads	20	The number of threads to use when processing properties.
scanRate	25	The rate (in milliseconds) at which to check the buffer status.
sizeThreshold	1000	The maximum number of items to accumulate before flushing the property buffer.

MssqlPersistenceProviderPackage

Setting	Default	Description
ConnectionInformation		
acquireIncrement	5	Determines how many connections at a time ThingWorx will try to acquire when the pool is exhausted.
acquireRetryAttempts	3	Defines how many times ThingWorx will try to acquire a new connection from the database before giving up.
acquireRetryDelay	10000	The time (in milliseconds) ThingWorx will wait between acquire attempts.
checkoutTimeout	1000000	The number of milliseconds a client calling **getConnection()** will wait for a connection to be checked-in or acquired when the pool is exhausted.
driverClass	com.microsoft.sqlserver.jdbc.SQLServerDriver	The fully-qualified class name of the JDBC driverClass that is expected to provide connections.
fetchSize	5000	The count of rows to be fetched in batches instead of caching all rows on the client side.
idleConnectionTestPeriod	60	Time period (in seconds) where connections will be tested so that idle connections won't be closed from outside processes such as firewalls, etc. If this is a number greater than 0,

MssqlPersistenceProviderPackage (continued)

Setting	Default	Description
		ThingWorx will test all idle, pooled but unchecked-out connections, every x number of seconds. **Note** If you are experiencing "No connection to model provider" errors, review this setting. Compare to firewall defaults. Lowering the default will alleviate disconnection issues.
initialPoolSize	5	Initial number of database connections created and maintained within a pool upon startup. Should be between **minPoolSize** and **maxPoolSize**.
jdbcUrl	jdbc:sqlserver://localhost:1433;databaseName=thingworx;applicationName=Thingworx;	The jdbc url used to connect to MSSQL.
maxConnectionAge	0	Seconds, effectively a time to live. A connection older than **maxConnectionAge** will be destroyed and purged from the pool.
maxIdleTime	0	Seconds a connection can remain pooled but unused before being discarded. Zero means idle connections never expire.
	300	The number of seconds

MssqlPersistenceProviderPackage (continued)

Setting	Default	Description
maxIdleTimeExcess-Connections		that connections in excess of **minPoolSize** are permitted to remain in idle in the pool before being culled. Intended for applications that wish to aggressively minimize the number of open connections, shrinking the pool back towards **minPoolSize** if, following a spike, the load level diminishes and Connections acquired are no longer needed. If **maxIdleTime** is set, **maxIdleTimeExcess-Connections** should be smaller to have any effect. Setting this to zero means no enforcement and excess connections are not idled out.
maxPoolSize	100	Maximum number of Connections a pool will maintain at any given time.
maxStatements	100	The size of the ThingWorx global PreparedStatement cache.
minPoolSize	5	Minimum number of Connections a pool will maintain at any given time.
numHelperThreads	8	The number of helper threads to spawn. Slow JDBC operations are generally performed by helper threads that don't hold contended locks.

MssqlPersistenceProviderPackage (continued)

Setting	Default	Description
		Spreading these operations over multiple threads can significantly improve performance by allowing multiple operations to be performed simultaneously.
password	*<unique password>*	The password to log into the database.
testConnectionOn-Checkout	false	If true, an operation will be performed at every connection checkout to verify that the connection is valid.
unreturnedConnection-Timeout	0	The number of seconds to wait for a response from an unresponsive connection before discarding it. If set, if an application checks out but then fails to check-in a connection within the specified period of time, the pool will discard the connection. This permits applications with occasional connection leaks to survive, rather than eventually exhausting the Connection pool. Zero means no timeout, and applications are expected to close their own connections.
username	msadmin	This is the userid that owns the TWSCHEMA schema and is used for

MssqlPersistenceProviderPackage (continued)

Setting	Default	Description
		authentication to MSSQL in the JDBC connection string.
Stream Processor Settings		
maximumBlockSize	2500	The maximum number of stream writes to process in one block.
maximumQueueSize	250000	The maximum number of stream entries to queue (will be rejected after that).
maximumWaitTime	10000	Number of milliseconds the system waits before flushing the stream buffer.
numberOfProces-singThreads	5	The number of processing threads.
scanRate	5	The buffer status is checked at the specified rate value in milliseconds.
sizeThreshold	1000	Maximum number of items to accumulate before flushing the stream buffer.
Value Stream Processor Settings		
maximumBlockSize	2500	Maximum number of value stream writes to process in one block.
maximumWaitTime	10000	Number of milliseconds the system waits before flushing the value stream buffer.
maximumQueueSize	500000	Maximum number of value stream entries to queue (will be rejected after that).
numberofProcessingTh-reads	5	The number of processing threads.

MssqlPersistenceProviderPackage (continued)

Setting	Default	Description
scanRate	5	The rate (in milliseconds) before flushing the stream buffer.
sizeThreshold	1000	Maximum number of items to accumulate before flushing the value stream buffer.
PersistentPropertyProcessorSettings		
maximumBlockSize	2500	The maximum number of property writes to process in one block.
maximumWaitTime	1000	The maximum wait time (in milliseconds) before flushing the property buffer.
maximumQueueSize	100000	The maximum number of property entries to queue (will be rejected after that).
numberOfProces-singThreads	20	The number of threads to use when processing properties.
scanRate	25	The rate (in milliseconds) at which to check the buffer status.
sizeThreshold	1000	The maximum number of items to accumulate before flushing the property buffer.

Installation Troubleshooting

Issue	Possible Resolution(s)
How do I enable Cross Origin Resource Sharing (CORS) in ThingWorx?	Enabling CORS allows requests to be made from a domain/website to an instance of ThingWorx that is deployed on a different server. This can be done by updating the Apache Tomcat web.xml file. Detailed process steps are available at https://www.ptc.com/en/support/article?n=CS229450.
After installing Tomcat and deploying the `Thingworx.war` file, Composer doesn't load with a **404 error Application not found**	• Verify the proper port on Tomcat is being used when accessing Composer • Check for proxy server/redirection • Verify that the `Thingworx.war` file and corresponding folder in `<Tomcat directory>/webapps` have the correct case (**Thingworx**, not **thingworx** or **ThingWorx**) **Note** If the folder or WAR file were deployed with the wrong case, shut down the Tomcat server, remove the "thingworx" folder from webapps, rename the thingworx.war file to the correct case and restart Tomcat. • Verify that the URL accessed is correct `http://<server>:<port>/Thingworx` (not `http://<server>:<port>/ThingWorx)` • If a 404 page not found error is encountered in a RHEL environment after ThingWorx installation, verify the following steps as well:

Issue	Possible Resolution(s)
	○ Verify that the JDK is present in the `/usr/lib/jvm/` folder. If the JDK is not present, then follow the steps to install Java in Installing Oracle Java and Apache Tomcat (RHEL) on page ○ Verify that the JAVA_HOME environment variable has the JDK path. For example, `JAVA_HOME = /usr/lib/jvm/ jdk1.8.144`
Problem deploying `thingworx.war`.	Verify that the `ThingworxStorage/ extensions/web-inf` folder contains the licensing libraries (DLL files).
The following error is received when deploying ThingWorx: `org.apache.catalina.core.ApplicationContext.log HTMLManager: FAIL - Deploy Upload Failed, Exception: org.apache.tomcat.util.http.fileupload.FileUploadBase$SizeLimitExceededException: the request was rejected because its size (90883556) exceeds the configured maximum (52437800) java.lang.IllegalStateException: org.apache.tomcat.util.http.fileupload.FileUploadBase$SizeLimitExceededException: the request was rejected because its size (90883556) exceeds the configured maximum (52437800) at org.apache.catalina.connector.Request.parseParts(Request.java:2871`	The max file size in the Tomcat `web.xml` file must be increased (default is 50MB). This file is located at : `<path to Tomcat>\Apache Software Foundation\Tomcat 8.5\webapps\manager\WEB-INF` 1. Open the `web.xml`. 2. Change the max-file-size and max-request-size to 104857600. 3. Save and close the file. 4. Restart Tomcat.
The following error message is received when importing a PTC licensed extension: `is licensed but cannot find feature in license.bin file`	Visit the Manage Licenses section on the PTC Support site to confirm the correct license file that matches your entitlement. If you need further assistance with your licenses, please contact the License Management team.
The following error message is	Remove `-Djava.library.path`

Issue	Possible Resolution(s)

received when attempting to undeploy ThingWorx:
`FAIL - Unable to delete [<path to Tomcat>\webapps\Thingworx].`
`The continued presence of this file may cause problems.`
`Due to FlxCore64.dll`
`(<path to Tomcat>\webapps\Thingworx\WEB-INF\extensions\FlxCore64.dll)`
	from Tomcat's Java configuration before undeployment.

An error message similar to the following is seen in the `ConfigurationLog.log`:
`2017-03-10 05:56:07.097-0500 [L: ERROR] [O: ] [I: ] [U: SuperUser] [S: ] [T: localhost-startStop-1]`
` **********LICENSING ERROR ANALYSIS`
`2017-03-10 05:56:07.097-0500 [L: ERROR] [O: ] [I: ] [U: SuperUser] [S: ] [T: localhost-startStop-1]`
`/Library/flexs is listed as a java.library.path but it does not exist.`
`/Library/blah is listed as a java.library.path but it does not exist.`
`/Library/zzz is listed as a java.library.path but it does not exist.`
`No flx dll files found.`
`Is the java.library.path set?`
`2017-03-10 05:56:07.097-0500 [L: ERROR] [O: ] [I: ] [U: SuperUser] [S: ] [T: localhost-startStop-1]`
` **********END LICENSING ERROR ANALYIS`
	The log message verifies if there is an issue with the license file.

Issue	Possible Resolution(s)

An error message similar to the following is thrown while the platform is starting:
`2017-06-12 11:33:59.204+0530 [L: ERROR]`
`[O: c.t.s.s.l.LicensingSubsystem]`
`[I: ] [U: SuperUser]`
`[S: ] [T: localhost-startStop-1]`
`[message: The size of provided data is`
`incorrect.]`
`2017-06-12 11:33:59.205+0530 [L: ERROR]`
`[O: c.t.s.s.l.LicensingSubsystem] [I: ]`
`[U: SuperUser] [S: ] [T: localhost-`
`startStop-1]`
`======================================`
`2017-06-12 11:33:59.205+0530 [L: ERROR]`
`[O: c.t.s.s.l.LicensingSubsystem] [I: ]`
`[U: SuperUser] [S: ] [T: localhost-`
`startStop-1]`
`Invalid License file:`
`/ThingworxPlatform\license.bin`
`2017-06-12 11:33:59.205+0530 [L: ERROR]`
`[O: c.t.s.s.l.LicensingSubsystem] [I: ]`
` [U: SuperUser] [S: ] [T: localhost-`
`startStop-1]`
`======================================`
`2017-06-12 11:33:59.205+0530 [L: WARN]`
`[O: c.t.s.ThingWorxServer] [I: ]`
`[U: SuperUser] [S: ] [T: localhost-`
`startStop-1] Shutting down the Platform.`
	The license file may have been opened/ edited/saved in a browser. Download the license file again, rename it to `license_capability_response.bin`, and place in `ThingworxPlatform` folder without editing or saving it.