



Installing ThingWorx 8

V1.0

Copyright © 2018 PTC Inc. and/or Its Subsidiary Companies. All Rights Reserved.

User and training guides and related documentation from PTC Inc. and its subsidiary companies (collectively "PTC") are subject to the copyright laws of the United States and other countries and are provided under a license agreement that restricts copying, disclosure, and use of such documentation. PTC hereby grants to the licensed software user the right to make copies in printed form of this documentation if provided on software media, but only for internal/personal use and in accordance with the license agreement under which the applicable software is licensed. Any copy made shall include the PTC copyright notice and any other proprietary notice provided by PTC. Training materials may not be copied without the express written consent of PTC. This documentation may not be disclosed, transferred, modified, or reduced to any form, including electronic media, or transmitted or made publicly available by any means without the prior written consent of PTC and no authorization is granted to make copies for such purposes. Information described herein is furnished for general information only, is subject to change without notice, and should not be construed as a warranty or commitment by PTC. PTC assumes no responsibility or liability for any errors or inaccuracies that may appear in this document.

The software described in this document is provided under written license agreement, contains valuable trade secrets and proprietary information, and is protected by the copyright laws of the United States and other countries. It may not be copied or distributed in any form or medium, disclosed to third parties, or used in any manner not provided for in the software licenses agreement except with written prior approval from PTC.

UNAUTHORIZED USE OF SOFTWARE OR ITS DOCUMENTATION CAN RESULT IN CIVIL DAMAGES AND CRIMINAL PROSECUTION.

PTC regards software piracy as the crime it is, and we view offenders accordingly. We do not tolerate the piracy of PTC software products, and we pursue (both civilly and criminally) those who do so using all legal means available, including public and private surveillance resources. As part of these efforts, PTC uses data monitoring and scouring technologies to obtain and transmit data on users of illegal copies of our software. This data collection is not performed on users of legally licensed software from PTC and its authorized distributors. If you are using an illegal copy of our software and do not consent to the collection and transmission of such data (including to the United States), cease using the illegal version, and contact PTC to obtain a legally licensed copy.

Important Copyright, Trademark, Patent, and Licensing Information: See the About Box, or copyright notice, of your PTC software.

UNITED STATES GOVERNMENT RIGHTS

PTC software products and software documentation are "commercial items" as that term is defined at 48 C.F.R. 2.101. Pursuant to Federal Acquisition Regulation (FAR) 12.212 (a)-(b) (Computer Software) (MAY 2014) for civilian agencies or the Defense Federal Acquisition Regulation Supplement (DFARS) at 227.7202-1(a) (Policy) and 227.7202-3 (a) (Rights in commercial computer software or commercial computer software documentation) (FEB 2014) for the Department of Defense, PTC software products and software documentation are provided to the U.S. Government under the PTC commercial license agreement. Use, duplication or disclosure by the U.S. Government is subject solely to the terms and conditions set forth in the applicable PTC software license agreement.

PTC Inc., 140 Kendrick Street, Needham, MA 02494 USA

Contents

ThingWorx Installation Overview	4
Windows Installation.....	6
H2	7
PostgreSQL	18
Ubuntu Installation	38
H2	39
PostgreSQL	51
RHEL Installation	75
H2	76
PostgreSQL	87
Amazon RDS Installation	110
Installation Appendices.....	121
Apache Tomcat Java Option Settings	122
platform-settings.json Configuration Details	124
Installation Troubleshooting.....	153

1

ThingWorx Installation Overview

Note

These installation steps were tested on ThingWorx 8.3.0, and file names used in the process reflect this, but the general steps can be used for any version of ThingWorx 8. Installation steps are available in the [Help Center](#) and PDF versions for other versions are located on the [PTC Support Portal](#).

Installation Prerequisites

Prerequisite third-party software includes Apache Tomcat and Oracle Java. PostgreSQL is also required if you are not using H2, MSSQL Server, or SAP HANA for your database. Reference the [ThingWorx Deployment Architecture Guide](#) for more information about database and deployment options.

Installation Options

ThingWorx is currently supported on

- [Windows on page 6](#)
- [Ubuntu on page 38](#)
- [RHEL on page 75](#)
- [Amazon RDS on page 110](#)

Database Options

There are several database options to consider before installing ThingWorx. H2 is an embedded database option, while PostgreSQL, MSSQL, and SAP HANA are external databases that require additional steps to configure. For more information on database options, see the [ThingWorx Deployment Architecture Guide](#) and the [ThingWorx Sizing Guide](#).

Note

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

- SAP HANA: [Getting Started with SAP HANA and ThingWorx Guide](#)
- Microsoft SQL Server: [Getting Started with MS SQL Server and ThingWorx Guide](#)

For additional information on database options, see the [Persistence Providers](#) topic.

Upgrading

If you are upgrading to a newer version, refer to the [Upgrading ThingWorx](#) guide.

System Requirements

For detailed software and hardware requirements, refer to the [ThingWorx System Requirements](#) document.

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 Administrator's Guide](#).

Metrics Reporting

By default, ThingWorx metrics data (such as usage, performance, and diagnostics) is sent to a PTC server. The configuration settings for metrics reporting are included in the [Platform Subsystem](#) and must be changed to opt out.

2

Windows Installation

H2.....	7
PostgreSQL.....	18

- [H2 on page 7](#)
- [PostgreSQL on page 32](#)

 **Note**

See [ThingWorx Installation Overview on page 4](#) for other options.

H2

Install Java and Apache Tomcat (Windows)

1. Download and install the required version of the Java JDK from the [Oracle website](#).

 **Note**

Refer to the [System Requirements](#) document for version requirements.

2. Visit the [Tomcat website](#) to download the **32-bit/64-bit Windows Service Installer (pgp, sha1, sha512)**.

 **Note**

Refer to the [System Requirements](#) document for version requirements.

 **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.

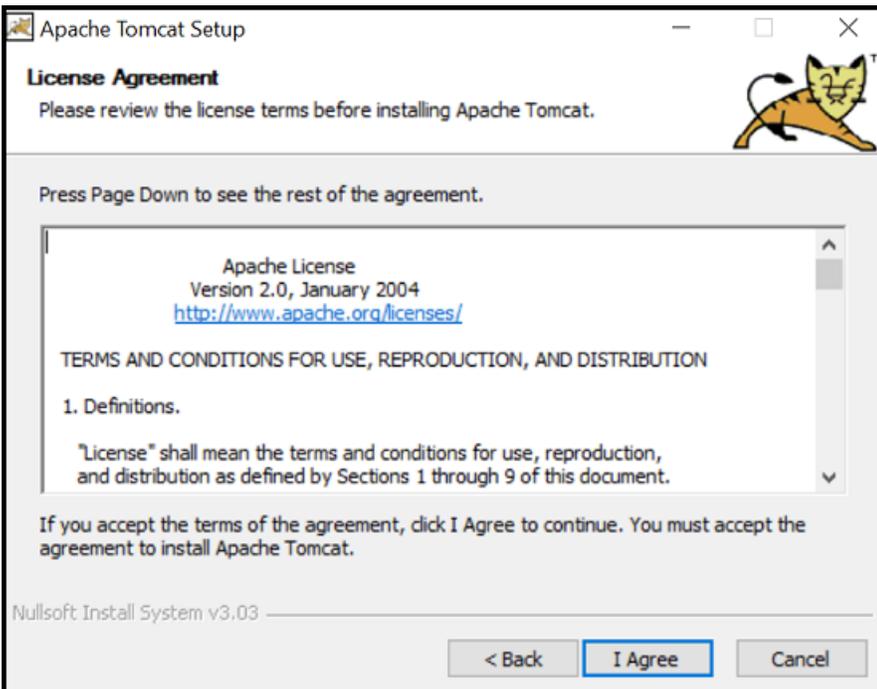
Core:

- [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 \(pgp, sha1, sha512\)](#)

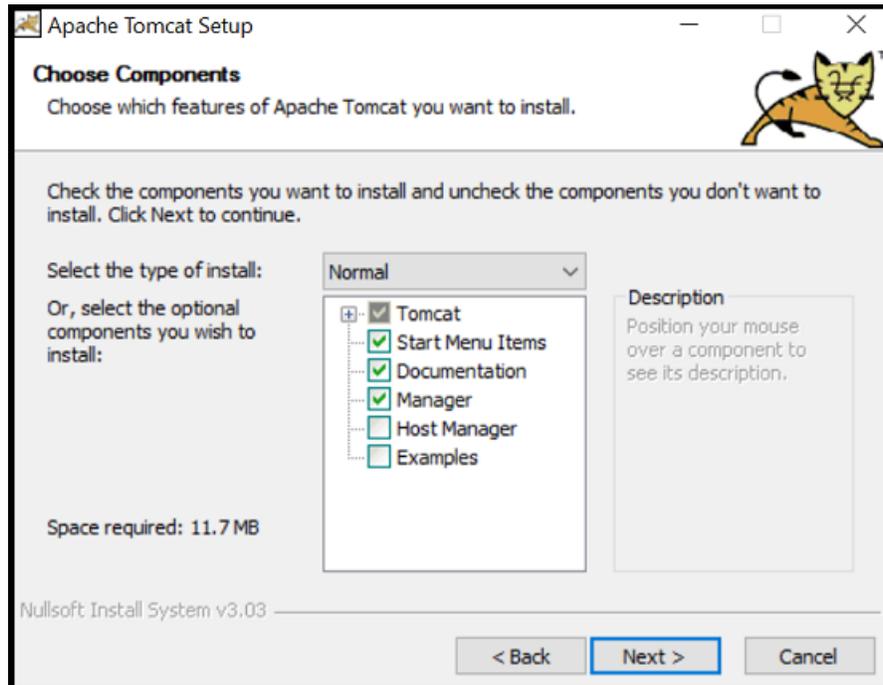
3. The Apache Tomcat Setup Wizard launches. Click **Next**.



4. Click **I Agree**.



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

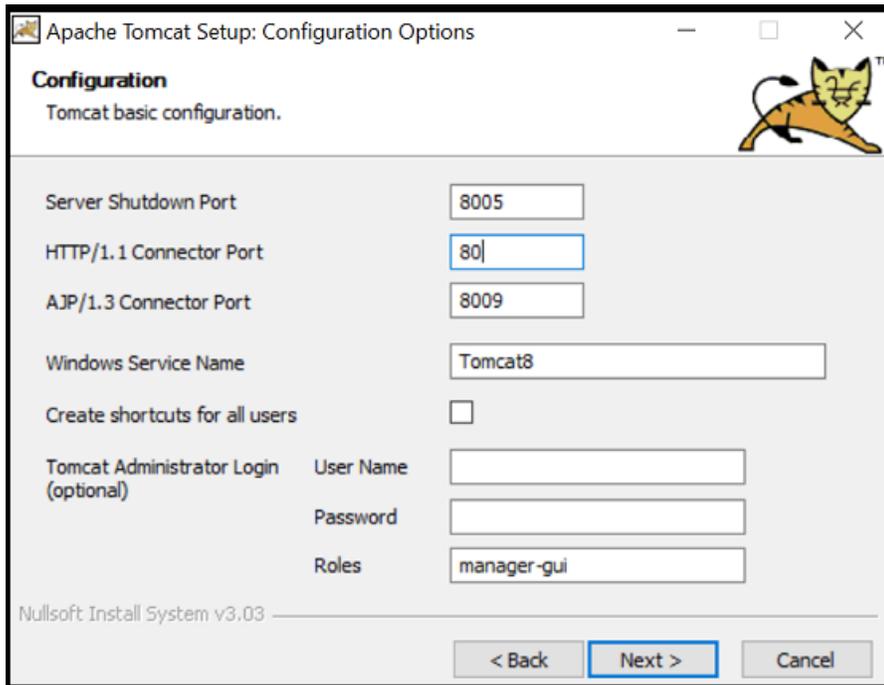


6. In the **HTTP/1.1 Connector Port** field, type 80 (or other available port).
7. In the **Tomcat Administrator Login** fields, enter a Tomcat user name and a unique, secure password for Tomcat administration.

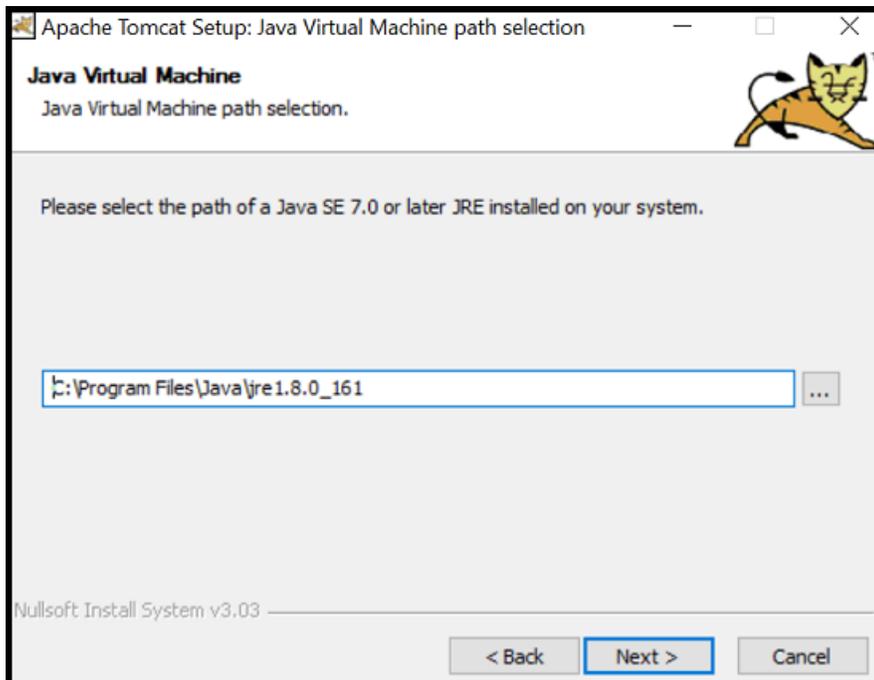
 **Note**

Although setting a Tomcat Administrator Login is shown as optional, it is required for use with ThingWorx.

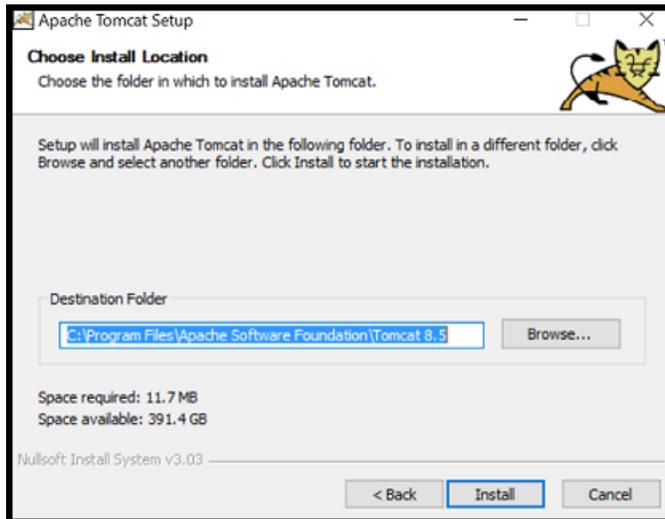
8. Click **Next**.



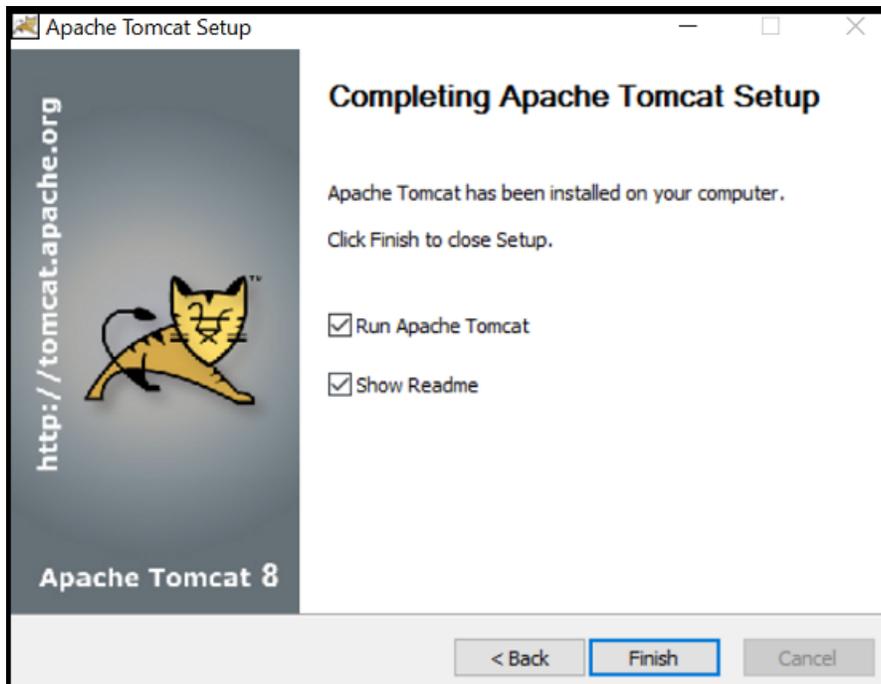
9. Click **Next**.



10. Click **Install**.



11. Click **Finish**.



12. Launch Tomcat. Click **Configure Tomcat**. In the Configure Tomcat window, click the **Java** tab.

13. 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

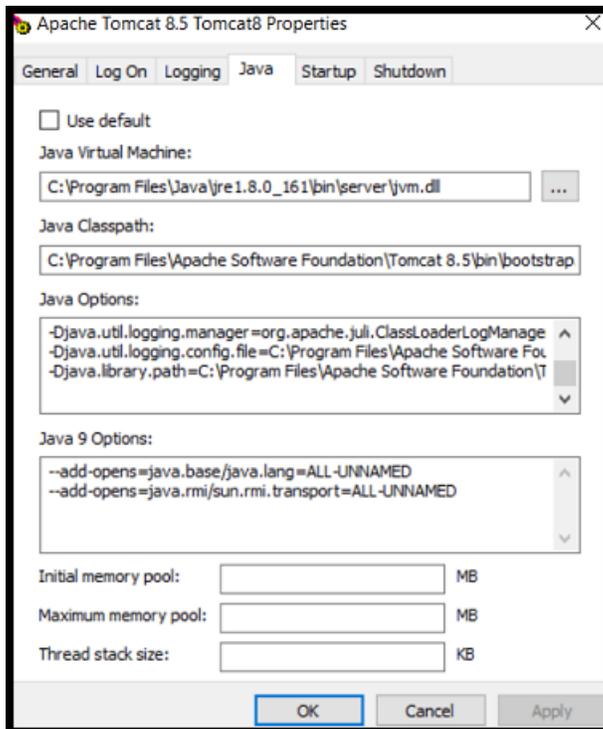
Djava.library.path example:

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

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 122](#).

14. Clear any values in the **Initial memory pool** and **Maximum memory pool** fields.



15. Click **OK**
16. (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"/>
```

 **Note**

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
```

17. For H2 ONLY: Go to [Install ThingWorx on page 13](#).
18. For POSTGRESQL ONLY: Go to [Install and Configure PostgreSQL on page 24](#).

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. For more information, reference [this article](#).

2. If you have not already done so, obtain the Thingworx.war file.

 **Note**

ThingWorx downloads are available in [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 10 characters long. Reference [platform-settings.json Configuration Details on](#)

[page 124](#) for more information on placement. See [Passwords on page](#) for additional information on setting passwords.

 **Note**

Do not copy and paste the sample below, as it may cause bad formatting in your `platform-settings.json`. Instead, click the link below 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`. Password must be a minimum of 10 characters, check the following:

- The password setting exists in `platform-settings.json`
 - The password is valid (10 or more characters)
 - The `platform-settings.json` file is formatted correctly - bad formatting could lead to errors
-

5. Configure licensing:

- Open the `platform-settings.json` file and add the following to the `PlatformSettingsConfig` section (reference [platform-settings](#)).

[json Configuration Options on page 124](#) 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).

6. Encrypt the license server password.

 **Note**

This step is optional, but it is the recommended best practice to encrypt the password.

- a. Create a working directory where you will perform this process, and copy the `Thingworx.war` file to that location.
- b. Unzip the `Thingworx.war`.
- c. Open a command prompt, `cd` to your working directory, and set your `CLASSPATH` by doing the following:
 - Go to **Control Panel > System Properties > Environment Variables**. Create a new environment variable named **PG_PW_UTIL**:

 **Note**

The file versions are based on ThingWorx 8.3.0, and may need to be changed if you are using a different version. Replace **xx** with the build number you are using.

Set the **PG_PW_UTIL** variable to:

```
PG_PW_UTIL
"<location where zip file is extracted>\WEB-INF\lib\
thingworx-platform-common-8.3.0-bxx.jar;
<location where zip file is extracted>\WEB-INF\lib\slf4j-
api-1.7.25.jar;
<location where zip file is extracted>\WEB-INF\lib
\logback-core-1.2.3.jar;
<location where zip file is extracted>\WEB-INF\lib
\logback-classic-1.2.3.jar;
<location where zip file is extracted>\WEB-INF\lib\
thingworx-common-8.3.0-bxx.jar"
```

- d. Append the `%PG_PW_UTIL%` variable to the `CLASSPATH` variable. For example:

```
CLASSPATH =<don't touch existing classpath>; %PG_PW_UTIL%
```
- e. In your command shell, enter `'java -version'`. It should respond with a Java version.
- f. Stop Tomcat.
- g. Open `/ThingworxPlatform/platform-settings.json` and change the `LicensingConnectionSettings password` value to `'encrypt.licensing.password'`. For example, `"password":`

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

- h. To create a key store with the licensing password encrypted inside, run the following command. In the second argument, enter your unique license server password:

```
java com.thingworx.security.keystore.ThingworxKeyStore  
encrypt.licensing.password <unique license_password>
```

 **Note**

By default, the password is stored in /ThingworxPlatform. The keystore is stored in /ThingworxStorage. If you are planning to configure custom folder locations, run the following command:

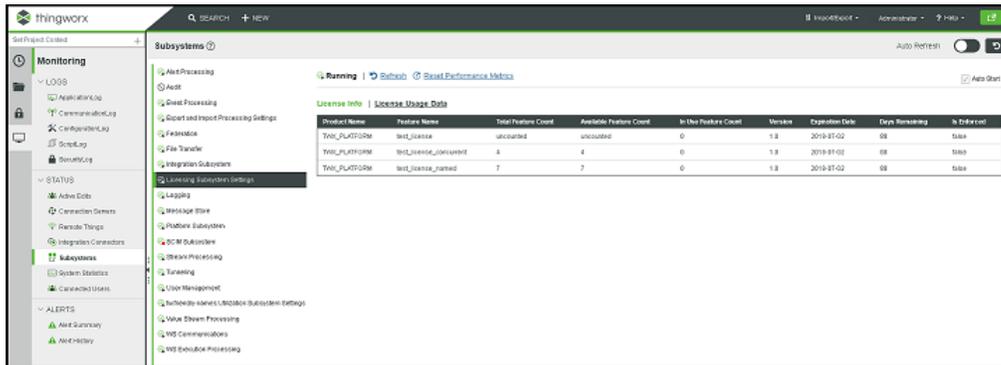
```
java com.thingworx.security.keystore.ThingworxKeyStore  
encrypt.licensing.password <unique license_password>  
<Password location> <Keystore location>
```

7. Start Tomcat.
8. To launch ThingWorx, go to `http://<servername>:<port>/Thingworx` in a Web browser.
9. 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.
10. Select **Done**.
11. (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.



PostgreSQL

Install Java and Apache Tomcat (Windows)

1. Download and install the required version of the Java JDK from the [Oracle website](#).

Note

Refer to the [System Requirements](#) document for version requirements.

2. Visit the [Tomcat website](#) to download the **32-bit/64-bit Windows Service Installer (pgp, sha1, sha512)**.

Note

Refer to the [System Requirements](#) document for version requirements.

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.

```

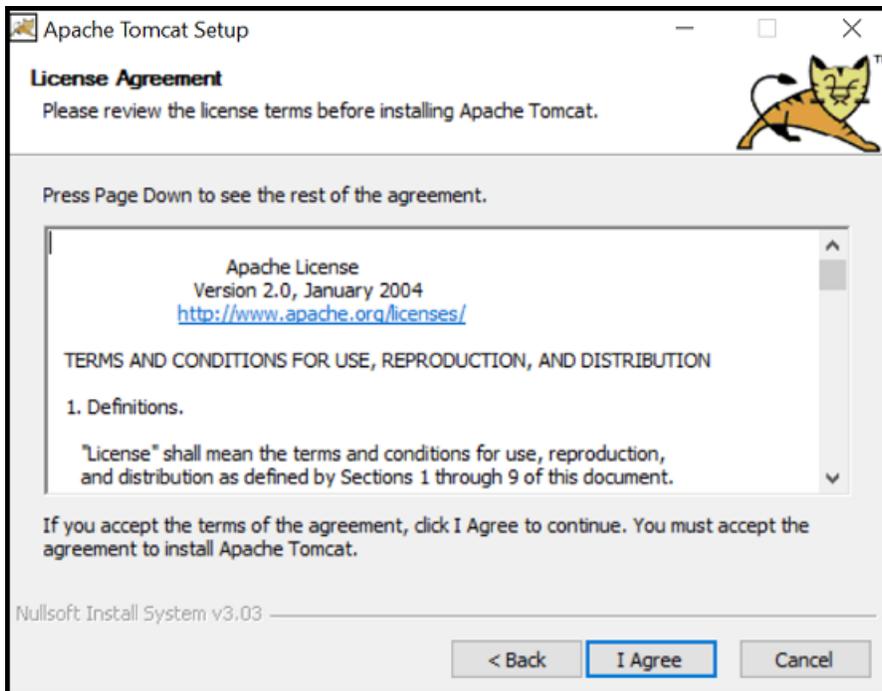
Core:
  o zip \(pgp, sha1, sha512\)
  o tar.gz \(pgp, sha1, sha512\)
  o 32-bit Windows zip \(pgp, sha1, sha512\)
  o 64-bit Windows zip \(pgp, sha1, sha512\)
  o 32-bit/64-bit Windows Service Installer \(pgp, sha1, sha512\)

```

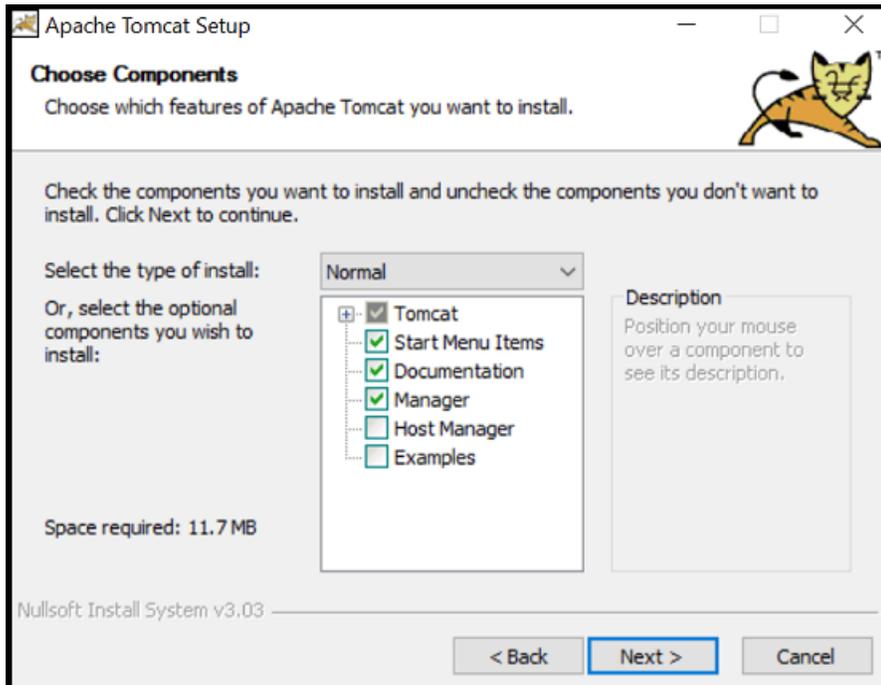
3. The Apache Tomcat Setup Wizard launches. Click **Next**.



4. Click **I Agree**.



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

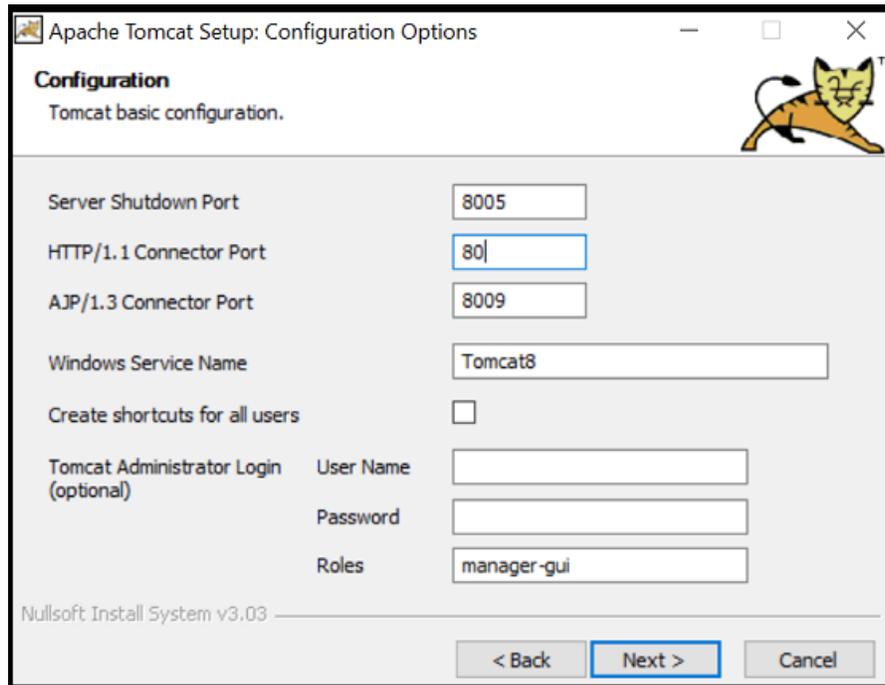


6. In the **HTTP/1.1 Connector Port** field, type 80 (or other available port).
7. In the **Tomcat Administrator Login** fields, enter a Tomcat user name and a unique, secure password for Tomcat administration.

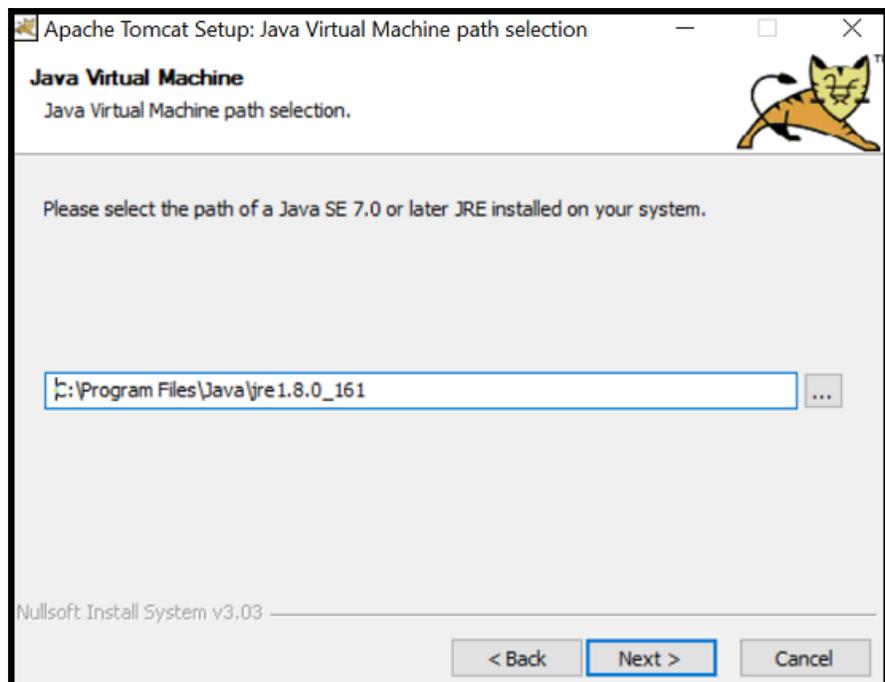
 **Note**

Although setting a Tomcat Administrator Login is shown as optional, it is required for use with ThingWorx.

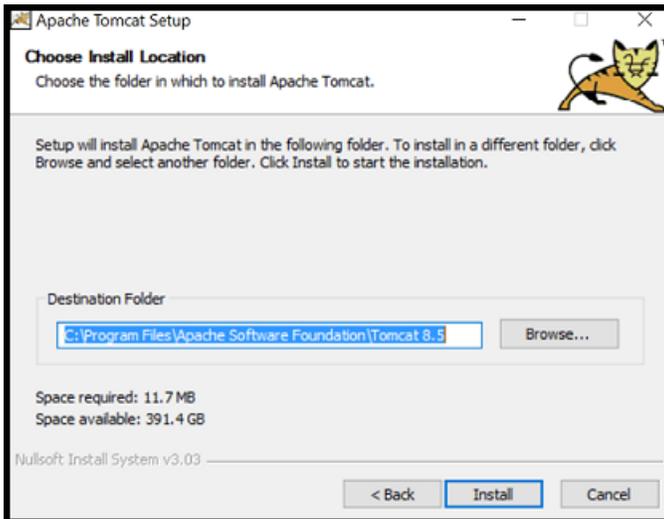
8. Click **Next**.



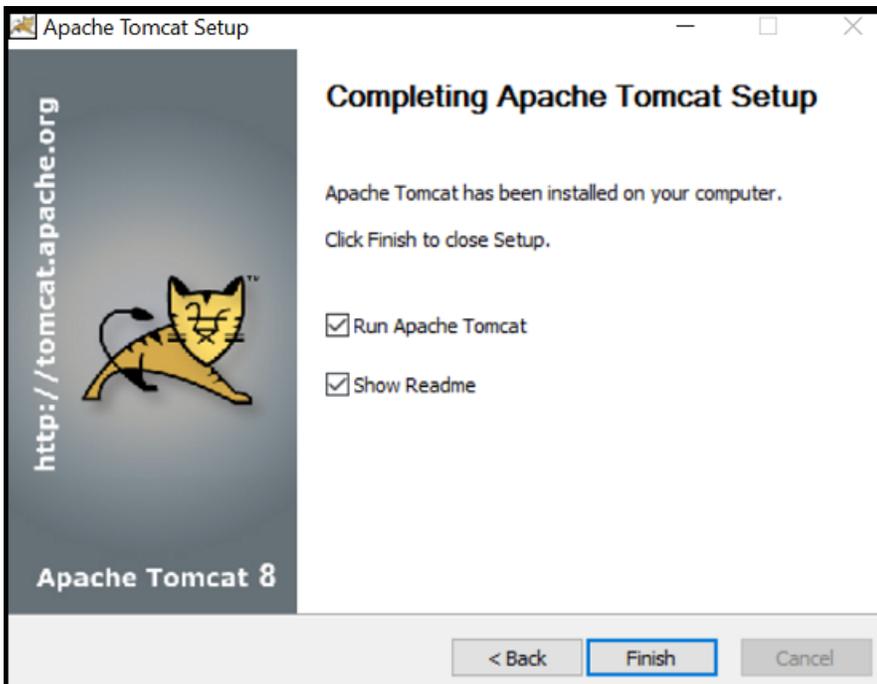
9. Click **Next**.



10. Click **Install**.



11. Click **Finish**.



12. Launch Tomcat. Click **Configure Tomcat**. In the Configure Tomcat window, click the **Java** tab.

13. 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**

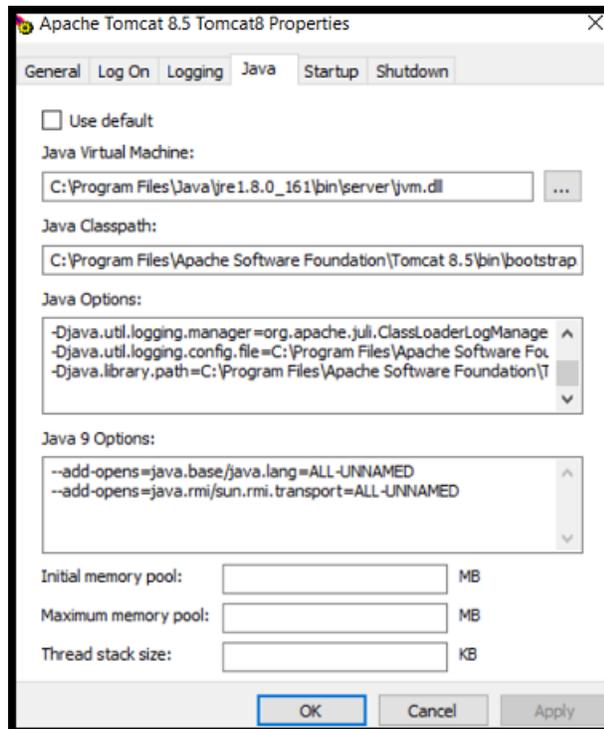
Djava.library.path example:

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

 **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 122](#).

14. Clear any values in the **Initial memory pool** and **Maximum memory pool** fields.



15. Click **OK**
16. (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"/>
```

 **Note**

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
```

17. For H2 ONLY: Go to [Install ThingWorx](#) on page 13.
18. For POSTGRES SQL ONLY: Go to [Install and Configure PostgreSQL](#) on page 24.

Install and Configure PostgreSQL (Windows)

The instructions provided below are intended for the PostgreSQL administrator (not the DB host servers).

 **Note**

If you are including the HA layer to your implementation, refer to the [ThingWorx High Availability Administrator's Guide](#).

Install PostgreSQL and Create a New User Role

1. Download and install the appropriate version of PostgreSQL from the following site: <http://www.postgresql.org/download/>

 **Note**

Refer to the [System Requirements](#) document for information on supported versions.

2. Open PostgreSQL using pgAdmin III.

 **Note**

pgAdmin III 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.

3. Create a new user role:
 - a. Right click **PostgreSQL9.x (<IP or host name of the database>:<Port number of PostgreSQL>)**. Example: PostgreSQL9.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.

 **Note**

You will need to re-enter this password in later steps.

4. Click **OK**.

 **Note**

Remember the user role name created in this step for later use.

Configure PostgreSQL Database Located on a Separate Server than ThingWorx (Optional)

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 Windows, the default data folder is `C:\Program Files\PostgreSQL\9.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:	<code>hostallall0.0.0.0/0md5</code>
If you want to allow only a specific IPv4 address to connect (Replace <code><ipAddress></code> with the IP address of the machine making the connection):	<code>hostallall<ipAddress>/32md5</code>
If you want to allow all IPv6 addresses to connect:	<code>hostallall:::0/0md5</code>
If you want to allow only a specific IPv6 address to connect (Replace <code><ipv6Address></code> with the appropriate address):	<code>hostallall<ipv6Address>/128md5</code>

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 \(9.4\)](#).

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**

If you create the folder using the `-d<databasename>` command, you do not have to use the PostgreSQL user.

 **Note**

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.

 **Note**

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.
5. Execute the script.

thingworxPostgresDBSetup Script Options

Option	Parameter	Default	Description	Example
t or -T	server	localhost	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	tablespace	thingworx	Name of the PostgreSQL tablespace	-h localhost

thingworxPostgresDBSetup Script Options (continued)

Option	Parameter	Default	Description	Example
-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-username	twadmin	User name that has permissions to write to the database.	-u twadmin

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 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.
3. Execute the script.

thingworxPostgresSchemaSetup Script Options

Option	Parameter	Default	Description	Example
-h or -H	server	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	-s mySchema

thingworxPostgresSchemaSetup Script Options (continued)

Option	Parameter	Default	Description	Example
			to use.	
-u or -U	username	twadmin	Username to update the database schema	-u twadmin
-o or -O	option	all	There are three options: <ul style="list-style-type: none"> • 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

Configure platform-settings.json

1. Create a folder named ThingworxPlatform at the root of the drive where Tomcat was installed or as a system variable.

 **Note**

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.

 **Note**

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 124](#).

 **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 (Optional)

If you want to provide added security encryption for the PostgreSQL database settings in the `platform-settings.json` file, you can perform the following steps.

Note

You must have Java installed and on your path. You must have PostgreSQL installed and recall the password.

1. Create a working directory where you will perform this process, such as `C:\<password_setup_location>` and copy the `Thingworx.war` to that location.

Note

ThingWorx downloads are available in [PTC Software Downloads](#).

2. Unzip the `Thingworx.war`.
3. Open a command prompt, `cd` to your working directory, and set your `CLASSPATH` by doing the following:
 - Go to **Control Panel > System Properties > Environment Variables** and create a new environment variable named **PG_PW_UTIL**:
`C:\<password_setup_location>\WEB-INF\lib\slf4j-api-1.7.25.jar;`
`C:\<password_setup_location>\WEB-INF\lib\logback-core-1.2.3.jar;`
`C:\<password_setup_location>\WEB-INF\lib\logback-classic-1.2.3.jar;`
`C:\<password_setup_location>\WEB-INF\lib\thingworx-common-<release-version>.jar`
 - Append the **PG_PW_UTIL** variable to the `CLASSPATH`:
`CLASSPATH`
`<don't touch existing classpath>; %PG_PW_UTIL%`
 - In your command shell, enter `'java -version'`. It should respond with a Java version.
4. Open `/ThingworxPlatform/platform-settings.json` and change the password value to `'encrypt.db.password'`. For example:
"password": "encrypt.db.password"

Note

Since the PostgreSQL admin password should not be included in the `platform-settings.json`, adding the `encrypt.db.password` string for the password signals the ThingWorx platform to look up the encrypted password in the keystore when it is encountered.

5. To create a key store with the PostgreSQL password encrypted inside, run the following command. In the second argument, enter your unique PostgreSQL password:

```
java com.thingworx.security.keystore.ThingworxKeyStore encrypt.db.password <unique postgres_password>
```

Note

By default, the password is stored in `/ThingworxPlatform`. The keystore is stored in `/ThingworxStorage`. If you are planning to configure custom folder locations, run the following command:

```
java com.thingworx.security.keystore.ThingworxKeyStore encrypt.db.password <unique postgres_password> <Password location> <Keystore location>
```

6. After you have created the encrypted password, remove the updates to the CLASSPATH.

Installing the PostgreSQL Client Package and PostgreSQL User (optional)

In order to issue PostgreSQL commands from the client machine to the PostgreSQL server, do so from a PostgreSQL user. The `postgresql-client-9.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 32](#).

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. For more information, reference [this article](#).

2. If you have not already done so, obtain the Thingworx.war file.

 **Note**

ThingWorx downloads are available in [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 10 characters long. Reference [platform-settings.json Configuration Details on page 124](#) for more information on placement. See [Passwords on page](#) for additional information on setting passwords.

 **Note**

Do not copy and paste the sample below, as it may cause bad formatting in your platform-settings.json. Instead, click the link below 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. Password must be a minimum of 10 characters, check the following:

- The password setting exists in platform-settings.json
- The password is valid (10 or more characters)
- The platform-settings.json file is formatted correctly - bad formatting could lead to errors

5. Configure licensing:

- Open the platform-settings.json file and add the following to the PlatformSettingsConfig section (reference [platform-settings.json Configuration Options on page 124](#) 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](#) for disconnected sites (no connection to PTC Support portal).

6. Encrypt the license server password.

 **Note**

This step is optional, but it is the recommended best practice to encrypt the password.

- a. Create a working directory where you will perform this process, and copy the `Thingworx.war` file to that location.
- b. Unzip the `Thingworx.war`.
- c. Open a command prompt, `cd` to your working directory, and set your `CLASSPATH` by doing the following:
 - Go to **Control Panel > System Properties > Environment Variables**. Create a new environment variable named **PG_PW_UTIL**:

 **Note**

The file versions are based on ThingWorx 8.3.0, and may need to be changed if you are using a different version. Replace **xx** with the build number you are using.

Set the **PG_PW_UTIL** variable to:

```
PG_PW_UTIL
"<location where zip file is extracted>\WEB-INF\lib
\thingworx-platform-common-8.3.0-bxx.jar;
```

```
<location where zip file is extracted>\WEB-INF\lib\slf4j-api-1.7.25.jar;
<location where zip file is extracted>\WEB-INF\lib\logback-core-1.2.3.jar;
<location where zip file is extracted>\WEB-INF\lib\logback-classic-1.2.3.jar;
<location where zip file is extracted>\WEB-INF\lib\thingworx-common-8.3.0-bxx.jar"
```

- d. Append the `%PG_PW_UTIL%` variable to the CLASSPATH variable. For example:
`CLASSPATH =<don't touch existing classpath>; %PG_PW_UTIL%`
- e. In your command shell, enter `'java -version'`. It should respond with a Java version.
- f. Stop Tomcat.
- g. Open `/ThingworxPlatform/platform-settings.json` and change the `LicensingConnectionSettings password` value to `'encrypt.licensing.password'`. For example, `"password": "encrypt.licensing.password",`. This password signals the ThingWorx platform to look up the encrypted licensing password in the keystore when it is encountered.
- h. To create a key store with the licensing password encrypted inside, run the following command. In the second argument, enter your unique license server password:

- ```
java com.thingworx.security.keystore.ThingworxKeyStore
encrypt.licensing.password <unique license_password>
```

---

### Note

By default, the password is stored in `/ThingworxPlatform`. The keystore is stored in `/ThingworxStorage`. If you are planning to configure custom folder locations, run the following command:

- ```
java com.thingworx.security.keystore.ThingworxKeyStore
encrypt.licensing.password <unique license_password>
<Password location> <Keystore location>
```

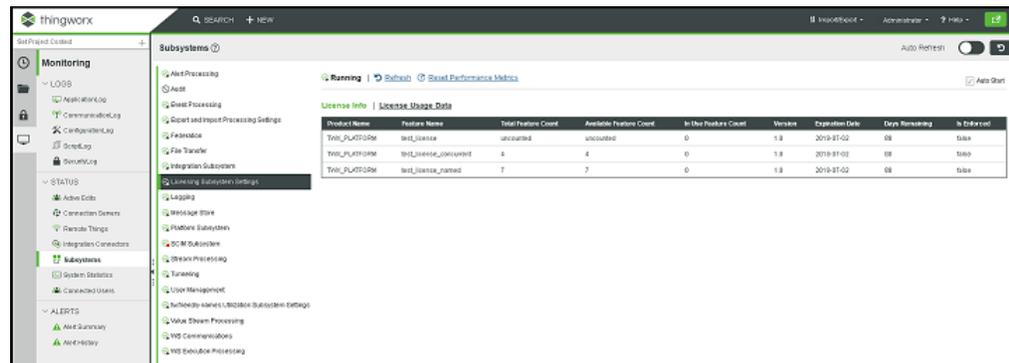
-
7. Start Tomcat.
 8. To launch ThingWorx, go to `http://<servername>:<port>/Thingworx` in a Web browser.
 9. Change the default password:

- 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 (minimum 10) 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.
10. Select **Done**.
 11. (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.



3

Ubuntu Installation

H2.....	39
PostgreSQL.....	51

- [H2 on page 39](#)
- [PostgreSQL on page 51](#)

 **Note**

See [ThingWorx Installation Overview on page 4](#) for other options.

H2

Install Java and Apache Tomcat (Ubuntu)

Note

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

1. Update Ubuntu packages:

```
$ sudo apt-get update
```
2. 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?](#)
-

3. Edit AUTHBIND properties to allow Tomcat to bind to ports below 1024:

```
$ sudo apt-get install authbind
```
4. Download the appropriate Java JDK tar file from [Oracle's website](#).

Note

Refer to the [System Requirements](#) document for version requirements.

5. Extract tar file:

```
$ tar -xf jdk-8uxxx-linux-x64.tar.gz
```
6. 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/
```

7. 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
```

8. Change access permissions:

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

9. Change owner:

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

10. 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.

11. Verify Java version:

```
$ java -version
```

 **Note**

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)
```

12. Download Apache Tomcat:

 **Note**

This 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.

13. Extract tar file:

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

14. 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
```

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

```
$ export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_131
$ export CATALINA_HOME=/usr/share/tomcat8.5/8.5.xx
```

16. Change directory to `$CATALINA_HOME`:

```
$ cd $CATALINA_HOME
```

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 `conf/`:

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

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

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

20. Create self-signed certificate:

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

21. 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
```

-
22. Uncomment the `Manager` element in `/$CATALINA_HOME/conf/context.xml` to prevent sessions from persisting across restarts:
- ```
<Manager pathname="" />
```

23. Comment out the following non-SSL Connector:

```
sudo vi $CATALINA_HOME/conf/server.xml

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

---

 **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
```

---

24. Modify the shutdown string and protocol used by the SSL Connector in `server.xml` by pasting in the following information below the code that was commented out in the previous step. Enter your `<keystore password>` that was previously set:

```
sudo vi $CATALINA_HOME/conf/server.xml
<Connector port="443" protocol="org.apache.coyote.http11.
Http11NioProtocol"
maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
keystoreFile="$ {user.home} /8.5.xx/conf/.keystore" keystorePass=
"<keystore password> " clientAuth="false" sslProtocol="TLS" />
```

25. 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"/>
```

26. Determine uid of tomcat8.5 user:

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

27. 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>
```

---

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

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

29. 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>
```

30. 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 "$@"
```

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

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

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

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

33. 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
```

34. 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
```

---

35. Set up Tomcat as a service to start on boot. First, build JSVC:

---

 **Note**

This may already be installed on your system. If so, skip and go to the next step.

---

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

36. 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 ../..
```

37. Create the Tomcat service file:

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

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

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

39. Paste the following in the Tomcat service file:

```
[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 \
-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

```

**40. 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"

```

**41. (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"/>

```

---

 **Note**

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

```

---

**42. H2 only: Go to [Install ThingWorx on page 46](#).**

**43. PostgreSQL only: Go to [Install and Configure PostgreSQL on page 58](#).**

---

## 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`:

```
$ sudo chown tomcat8.5:tomcat8.5 /ThingworxStorage /ThingworxBackupStorage /ThingworxPlatform
$ sudo chmod 775 /ThingworxStorage /ThingworxBackupStorage /ThingworxPlatform
```

---

### Note

Without these permissions, the server will fail to start. For more information, reference [this article](#).

---

3. If you have not already done so, obtain the `Thingworx.war` file.

---

### Note

ThingWorx downloads are available in [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 10 characters long. Reference [platform-settings.json Configuration Details](#) on

---

[page 124](#) for more information on placement. See [Passwords on page](#) for additional information on setting passwords.

---

 **Note**

Do not copy and paste the sample below, as it may cause bad formatting in your `platform-settings.json`. Instead, click the link below 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`. Password must be a minimum of 10 characters, check the following:

- The password setting exists in `platform-settings.json`
  - The password is valid (10 or more characters)
  - The `platform-settings.json` file is formatted correctly - bad formatting could lead to errors
- 

7. Configure licensing:

- Open the `platform-settings.json` file and add the following to the `PlatformSettingsConfig` section (reference [platform-settings](#)).

---

[json Configuration Options on page 124](#) 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).

- 
8. Encrypt the license server password.

---

 **Note**

This step is optional, but it is the recommended best practice to encrypt the password.

- a. Create a working directory where you will perform this process, such as `<password_setup_location>`, and copy the `Thingworx.war` file to that location.
- b. Unzip the `Thingworx.war`.

- 
- c. Open a command prompt, cd to your working directory, and set your CLASSPATH by doing the following:

---

 **Note**

The file versions are based on ThingWorx 8.3, and may need to be changed if you are using a different version. Replace **xx** with the build number you are using.

---

- ```
export CLASSPATH= /<password_setup_location>/WEB-INF/lib/
thingworx-platform-common-8.3.0-bxx.jar:
/<password_setup_location>/WEB-INF/lib/slf4j-api-1.7.25.
jar:
/<password_setup_location>/WEB-INF/lib/logback-core-
1.2.3.jar:
/<password_setup_location>/WEB-INF/lib/logback-classic-
1.2.3.jar:
/<password_setup_location>/WEB-INF/lib/thingworx-common-
8.3.0-bxx.jar
```
- d. In your command shell, enter 'java -version'. It should respond with a Java version.
- e. Stop Tomcat.

```
$ sudo service tomcat8.5 stop
```
- f. Open /ThingworxPlatform/platform-settings.json and change the LicensingConnectionSettings password value to 'encrypt.licensing.password'. For example, "password": "encrypt.licensing.password",. This password signals the ThingWorx platform to look up the encrypted licensing password in the keystore when it is encountered.
- g. To create a key store with the licensing password encrypted inside, run the following command. In the second argument, enter your unique license server password:

- ```
sudo java -classpath $CLASSPATH com.thingworx.security.
```

---

```
keystore.ThingworxKeyStore encrypt.licensing.password
<unique license_password>
```

---

 **Note**

By default, the password is stored in `/ThingworxPlatform`. The keystore is stored in `/ThingworxStorage`. If you are planning to configure custom folder locations, run the following command:

- ```
sudo java -classpath $CLASSPATH com.thingworx.security.  
keystore.ThingworxKeyStore encrypt.licensing.password  
<unique license_password> <Password location> <Keystore  
location>
```

9. **Start Tomcat.**

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

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

 **Note**

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

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

11. 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**.

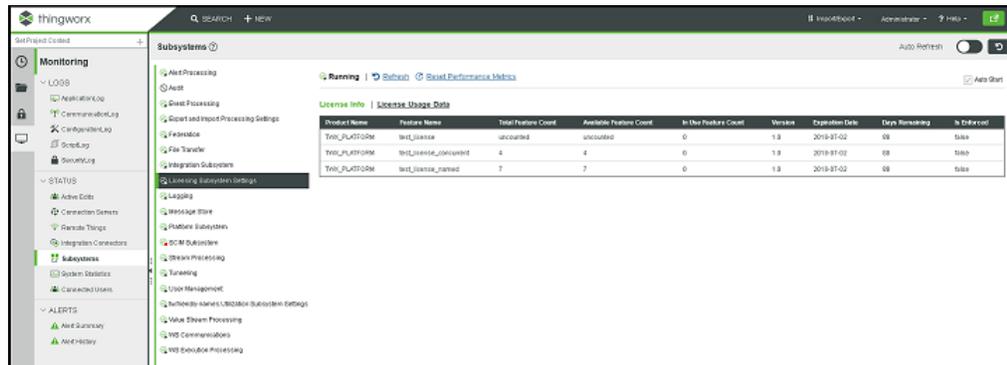
 **Note**

The password, which should not be easily guessed or a known, common password, is recommended to be at least 14 characters in length (minimum 10) 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.



PostgreSQL

Install Java and Apache Tomcat (Ubuntu)

Note

This version of ThingWorx has been tested with Ubuntu 14.04. Other versions may not be supported and may not work.

1. Update Ubuntu packages:
`$ sudo apt-get update`
2. 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?](#)

-
3. Edit AUTHBIND properties to allow Tomcat to bind to ports below 1024:

```
$ sudo apt-get install authbind
```
 4. Download the Java JDK tar file from Oracle's website, or run the following

 **Note**

The steps in this process have been tested with Java 8 update 131. Other versions are not supported and may not work.

```
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
```

5. Extract tar file:

```
$ tar -xf jdk-8u131-linux-x64.tar.gz
```
6. 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_131/ /usr/lib/jvm/
```

7. Add alternatives to the system:

```
$ sudo update-alternatives --install "/usr/bin/java" "java"
"/usr/lib/jvm/jdk1.8.0_131/bin/java" 1
$ sudo update-alternatives --install "/usr/bin/keytool"
"keytool" "/usr/lib/jvm/jdk1.8.0_131/bin/keytool" 1
```
8. Change access permissions:

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

```
$ sudo chown -R root:root /usr/lib/jvm/jdk1.8.0_131/
```
10. 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_131/bin/` can be installed using the previous set of steps.

11. Verify Java version:

```
$ java -version
```

Note

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

```
java version "1.8.0_131"  
Java(TM) SE Runtime Environment (build 1.8.0_131-b11)  
Java HotSpot(TM) 64-Bit Server VM (build 24.75-b04, mixed mode)
```

12. Download Apache Tomcat:

Note

This 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.

13. Extract tar file:

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

14. 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
```

15. Define environment variables in /etc/environment:

```
$ export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_131  
$ export CATALINA_HOME=/usr/share/tomcat8.5/8.5.xx
```

16. Change directory to \$CATALINA_HOME:

```
$ cd $CATALINA_HOME
```

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 `conf/`:

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

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

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

20. Create self-signed certificate:

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

21. 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
```

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

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

23. Comment out the following non-SSL Connector:

```
sudo vi $CATALINA_HOME/conf/server.xml
```

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

 **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
```

-
24. Modify the shutdown string and protocol used by the SSL Connector in `server.xml` by pasting in the following information below the code that was commented out in the previous step. Enter your `<keystore password>` that was previously set:

```
sudo vi $CATALINA_HOME/conf/server.xml
<Connector port="443" protocol="org.apache.coyote.http11.
Http11NioProtocol"
maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
keystoreFile="{user.home}/8.5.xx/conf/.keystore" keystorePass=
"<keystore password> " clientAuth="false" sslProtocol="TLS" />
```

25. 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"/>
```

26. Determine uid of tomcat8.5 user:

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

27. 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>
```

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

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

29. 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>
```

30. 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 "$@"
```

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

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

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

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

33. 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

```

34. 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

```

35. Set up Tomcat as a service to start on boot. First, build JSVC:

 **Note**

This may already be installed on your system. If so, skip and go to the next step.

```

$ sudo apt-get install gcc

```

36. 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 ../..

```

37. Create the Tomcat service file:

```

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

```

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

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

39. Paste the following in the Tomcat service file:

```
[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_131
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 \
        -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
```

40. 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"
```

41. (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"/>
```

 **Note**

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
```

42. H2 only: Go to [Install ThingWorx on page 46](#).

43. PostgreSQL only: Go to [Install and Configure PostgreSQL on page 58](#).

Install and Configure PostgreSQL (Ubuntu)

The instructions provided below are intended for the PostgreSQL administrator (not the DB host servers).

 **Note**

If you are including the HA layer to your implementation, refer to the [ThingWorx High Availability Administrator's Guide](#).

Install PostgreSQL and Create a New User Role

1. 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**

Refer to the [ThingWorx System Requirements](#) guide for supported versions of PostgreSQL.

 **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-9.x -y
```

2. Install PgAdmin III, the PostgreSQL admin tool:

```
$ sudo apt-get install pgadmin3 -y
```

 **Note**

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

3. 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>'"
```

4. 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.

5. Configure pgAdmin III:

```
$ sudo pgadmin3
```

- In the pgAdmin III GUI, click on **file->Open postgresql.conf**
- Open `/etc/postgresql/9.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/9.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 III

6. Restart the PostgreSQL service:

```
$ sudo service postgresql restart
```

7. Set up PgAdmin III to connect to the database:

```
$ sudo pgadmin3
```

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

```
Name: PostgreSQL 9.x
Host: localhost
Port: 5432
Service: <blank>
Maintenance DB: postgres
Username: postgres
Password: <unique PostgreSQL password as set previously >
Store password: Checked
Group: Servers
```

9. Click **OK**.

10. 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 **PostgreSQL9.x (<IP or host name of the database>:<Port number of PostgreSQL>)**. Example: PostgreSQL9.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).

 **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.

- e. Click **OK**.

Configure PostgreSQL Database Located on a Separate Server than ThingWorx (Optional)

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/9.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:	<code>hostallall0.0.0.0/0md5</code>
If you want to allow only a specific IPv4 address to connect (Replace	<code>hostallall<ipAddress>/32md5</code>

<ipAddress> with the IP address of the machine making the connection):	
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 \(9.4\)](#).

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 a folder named `ThingworxPostgresStorage` on the drive that the `ThingworxStorage` folder is located (in the root directory by default).

Note

If you create the folder using the `-d<databasename>` command, you do not have to use the PostgreSQL user.

 **Note**

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.

 **Note**

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 8.3.x download from the PTC site. If necessary, change the file name to the version you are using.

```
$ sudo unzip MED-61111-CD-083_ThingWorx-Platform-Postgres-8-3-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
```
5. Execute the script.

thingworxPostgresDBSetup Script Options

Option	Parameter	Default	Description	Example
t or -T	server	localhost	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	tablespace	thingworx	Name of the PostgreSQL tablespace	-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-username	twadmin	User name that has permissions to write to the database.	-u twadmin

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.bat` 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
```

3. Execute the script.

 **Note**

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

thingworxPostgresSchemaSetup Script Options

Option	Parameter	Default	Description	Example
-h or -H	server	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: <ul style="list-style-type: none">• 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

Configure platform-settings.json

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

```
$ sudo mkdir /ThingworxPlatform
```

 **Note**

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`

 **Note**

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 124](#).

 **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 (Optional)

If you want to provide added security encryption for the PostgreSQL database settings in the `platform-settings.json` file, you can perform the following steps.

Note

This encryption process is optional.

Note

You must have Java installed and on your path. You must have PostgreSQL installed and recall the password.

1. Create a working directory where you will perform this process, such as `~/<password_setup location>`, and copy the `Thingworx.war` to that location.

Note

ThingWorx downloads are available in [PTC Software Downloads](#).

2. Unzip the `Thingworx.war`.
3. Open a command prompt, `cd` to your working directory, and set your `CLASSPATH` by doing the following:

```
CLASSPATH= /  
<password_setup location>/WEB-INF/lib/logback-core-1.2.3.jar:/  
<password_setup location>/WEB-INF/lib/logback-classic-1.2.3.  
jar:/<password_setup location>/WEB-INF/lib/thingworx-common-  
8.3.0-bxx.jar
```
4. Open `/ThingworxPlatform/platform-settings.json` and change the password value to `'encrypt.db.password'`. For example:

```
"password": "encrypt.db.password"
```

Note

Since the PostgreSQL admin password should not be included in the `platform-settings.json`, adding the `encrypt.db.password` string for the password signals the ThingWorx platform to look up the encrypted password in the keystore when it is encountered.

-
5. To create a key store with the PostgreSQL password encrypted inside, run the following command. In the second argument, enter your unique PostgreSQL password:

```
$ sudo java -classpath $CLASSPATH
com.thingworx.security.keystore.ThingworxKeyStore encrypt.db.
password <unique postgres_password>
```

 **Note**

By default, the password is stored in `/ThingworxPlatform`. The keystore is stored in `/ThingworxStorage`. If you are planning to configure custom folder locations, run the following command:

```
$ sudo java -classpath $CLASSPATH
com.thingworx.security.keystore.ThingworxKeyStore
encrypt.db.password <unique postgres_password> <Password location>
<Keystore location>
```

6. After you have created the encrypted password, remove the updates to the CLASSPATH.

Installing the PostgreSQL Client Package and PostgreSQL User (optional)

In order to issue PostgreSQL commands from the client machine to the PostgreSQL server, do so from a PostgreSQL user. The `postgresql-client-9.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 69](#).

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`:

```
$ sudo chown tomcat8.5:tomcat8.5 /ThingworxStorage
/ThingworxBackupStorage /ThingworxPlatform
```

```
$ sudo chmod 775 /ThingworxStorage /ThingworxBackupStorage
/ThingworxPlatform
```

 **Note**

Without these permissions, the server will fail to start. For more information, reference [this article](#).

3. If you have not already done so, obtain the `Thingworx.war` file.

 **Note**

ThingWorx downloads are available in [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 10 characters long. Reference [platform-settings.json Configuration Details on page 124](#) for more information on placement. See [Passwords on page](#) for additional information on setting passwords.

 **Note**

Do not copy and paste the sample below, as it may cause bad formatting in your `platform-settings.json`. Instead, click the link below 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. Password must be a minimum of 10 characters, check the following:

- The password setting exists in platform-settings.json
- The password is valid (10 or more characters)
- The platform-settings.json file is formatted correctly - bad formatting could lead to errors

7. Configure licensing:

- Open the platform-settings.json file and add the following to the PlatformSettingsConfig section (reference [platform-settings.json Configuration Options on page 124](#) 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).

8. Encrypt the license server password.

 **Note**

This step is optional, but it is the recommended best practice to encrypt the password.

- a. Create a working directory where you will perform this process, such as `<password_setup_location>`, and copy the `Thingworx.war` file to that location.
- b. Unzip the `Thingworx.war`.
- c. Open a command prompt, `cd` to your working directory, and set your `CLASSPATH` by doing the following:

 **Note**

The file versions are based on ThingWorx 8.3, and may need to be changed if you are using a different version. Replace `xx` with the build number you are using.

- ```
export CLASSPATH= /<password_setup_location>/WEB-INF/lib/
thingworx-platform-common-8.3.0-bxx.jar:
/<password_setup_location>/WEB-INF/lib/slf4j-api-1.7.25.
jar:
```

---

```
/<password_setup_location>/WEB-INF/lib/logback-core-
1.2.3.jar:
/<password_setup_location>/WEB-INF/lib/logback-classic-
1.2.3.jar:
/<password_setup_location>/WEB-INF/lib/thingworx-common-
8.3.0-bxx.jar
```

- d. In your command shell, enter 'java -version'. It should respond with a Java version.
- e. Stop Tomcat.  

```
$ sudo service tomcat8.5 stop
```
- f. Open /ThingworxPlatform/platform-settings.json and change the LicensingConnectionSettings password value to 'encrypt.licensing.password'. For example, "password": "encrypt.licensing.password",. This password signals the ThingWorx platform to look up the encrypted licensing password in the keystore when it is encountered.
- g. To create a key store with the licensing password encrypted inside, run the following command. In the second argument, enter your unique license server password:

- ```
sudo java -classpath $CLASSPATH com.thingworx.security.
keystore.ThingworxKeyStore encrypt.licensing.password
<unique license_password>
```

Note

By default, the password is stored in /ThingworxPlatform. The keystore is stored in /ThingworxStorage. If you are planning to configure custom folder locations, run the following command:

- ```
sudo java -classpath $CLASSPATH com.thingworx.security.
keystore.ThingworxKeyStore encrypt.licensing.password
<unique license_password> <Password location> <Keystore
location>
```

---

## 9. Start Tomcat.

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

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

---

**Note**

Verify that a license file (successful\_license\_capability\_response.bin) is created in the ThingworxPlatform folder.

---

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**.

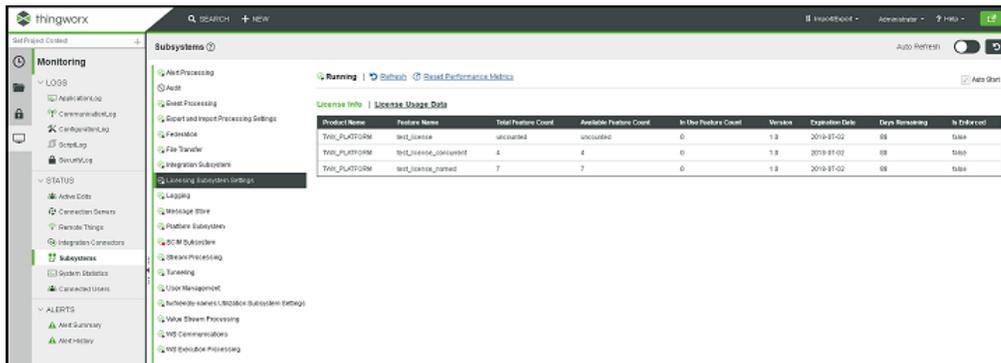
---

**Note**

The password, which should not be easily guessed or a known, common password, is recommended to be at least 14 characters in length (minimum 10) and should include a mix of uppercase and lowercase letters, numbers, and special characters.

---

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.



# 4

## RHEL Installation

|                 |    |
|-----------------|----|
| H2.....         | 76 |
| PostgreSQL..... | 87 |

- [H2 on page 76](#)
- [PostgreSQL on page 87](#)

---

 **Note**

See [ThingWorx Installation Overview on page 4](#) for other options.

---

---

## H2

### Install Java and Apache Tomcat (RHEL)

---

 **Note**

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

---

1. Download the Java (JDK) RPM file from [Oracle's website](#).

---

 **Note**

Refer to the [System Requirements](#) document for version requirements.

---

2. Run the Java installer:

```
$ sudo rpm -i jdk-8uxxx-linux-x64.rpm
```

3. Create the directory and move the JDK:

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

4. 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
```

5. 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
```

---

6. Change Owner:

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

7. 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
```

8. Verify Java version:

---

 **Note**

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 xx.xx-bxx, mixed mode)
```

9. Install Tomcat. Download the Tomcat installer:

---

 **Note**

This 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.

---

---

10. Extract the contents:

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

11. 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
```

12. 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
```

13. Change directory to `/usr/share/tomcat8.5/8.5.xx`:

```
$ cd /usr/share/tomcat8.5/8.5.xx
```

14. 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
```

15. 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/
```

16. 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
```

---

 **Note**

Permissions and ownership should be revisited for a production system to increase security on a operating system level.

---

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

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

18. Create self-signed certificate:

```
$ /usr/lib/jvm/jdk1.8.0_xxx/jre/bin/keytool -genkey -alias
tomcat8.5 -keyalg RSA
```

19. 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
```

20. 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.

21. Save the file.

22. Modify the shutdown string and protocol used by the SSL Connector in `server.xml`. Open `/usr/share/tomcat8.5/8.5.xx/conf/server.xml` in a text editor (as root) Uncomment the following section:

```
<Connector executor="tomcatThreadPool"
 port="80" protocol="HTTP/1.1"
 connectionTimeout="20000"
 redirectPort="8443" />

-->
```

23. Paste the following directly below the uncommented section:

```
<Connector port="443" protocol="org.apache.coyote.http11.
Http11NioProtocol"
maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
keystoreFile="${user.home}/8.5.xx/conf/.keystore" keystorePass=
"<PostgreSQL keystore password>" clientAuth="false"
sslProtocol="TLS" />
```

24. 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"/>
```

25. Save the file.

---

 **Note**

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

---

26. Set up Tomcat as a service to start on boot. First, build JSVC:

```
$ sudo yum install gcc
```

---

 **Note**

This may already be installed on your system.

---

```
$ 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 ../..

```

## 27. Create the Tomcat service file:

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

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

```

[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 \
 -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
```

29. 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"
```

30. Set Tomcat to run on system start up:

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

---

### Note

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
```

---

## 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 only: Go to [Install ThingWorx on page 82](#).
2. PostgreSQL only: Go to [Install and Configure PostgreSQL on page 93](#).

---

## 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`:

```
$ sudo chown tomcat8.5:tomcat8.5 /ThingworxStorage
/ThingworxBackupStorage /ThingworxPlatform
$ sudo chmod 775 /ThingworxStorage /ThingworxBackupStorage
/ThingworxPlatform
```

---

### Note

Without these permissions, the server will fail to start. For more information, reference [this article](#).

---

3. If you have not already done so, obtain the `Thingworx.war` file.

---

### Note

ThingWorx downloads are available in [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 10 characters long. Reference [platform-settings.json Configuration Details](#) on

---

[page 124](#) for more information on placement. See [Passwords on page](#) for additional information on setting passwords.

---

 **Note**

Do not copy and paste the sample below, as it may cause bad formatting in your `platform-settings.json`. Instead, click the link below 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`. Password must be a minimum of 10 characters, check the following:

- The password setting exists in `platform-settings.json`
  - The password is valid (10 or more characters)
  - The `platform-settings.json` file is formatted correctly - bad formatting could lead to errors
- 

7. Configure licensing:

- Open the `platform-settings.json` file and add the following to the `PlatformSettingsConfig` section (reference [platform-settings](#)).

---

[json Configuration Options on page 124](#) 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).

---

8. Encrypt the license server password.

---

 **Note**

This step is optional, but it is the recommended best practice to encrypt the password.

---

- a. Create a working directory where you will perform this process, such as `<password_setup_location>`, and copy the `Thingworx.war` file to that location.
- b. Unzip the `Thingworx.war`.
- c. Open a command prompt, `cd` to your working directory, and set your `CLASSPATH` by doing the following:

---

 **Note**

The file versions are based on ThingWorx 8.3, and may need to be changed if you are using a different version. Replace **xx** with the build number you are using.

---

- ```
export CLASSPATH= /<password_setup_location>/WEB-INF/lib/
thingworx-platform-common-8.3.0-bxx.jar:
/<password_setup_location>/WEB-INF/lib/slf4j-api-1.7.25.
jar:
/<password_setup_location>/WEB-INF/lib/logback-core-
1.2.3.jar:
/<password_setup_location>/WEB-INF/lib/logback-classic-
1.2.3.jar:
/<password_setup_location>/WEB-INF/lib/thingworx-common-
8.3.0-bxx.jar
```
- d. In your command shell, enter `'java -version'`. It should respond with a Java version.
- e. Stop Tomcat.

```
$ sudo service tomcat8.5 stop
```
- f. Open `/ThingworxPlatform/platform-settings.json` and change the `LicensingConnectionSettings password` value to `'encrypt.licensing.password'`. For example, `"password": "encrypt.licensing.password",`. This password signals the ThingWorx platform to look up the encrypted licensing password in the keystore when it is encountered.
- g. To create a key store with the licensing password encrypted inside, run the following command. In the second argument, enter your unique license server password:
 - ```
sudo java -classpath $CLASSPATH com.thingworx.security.
```

---

```
keystore.ThingworxKeyStore encrypt.licensing.password
<unique license_password>
```

---

 **Note**

By default, the password is stored in `/ThingworxPlatform`. The keystore is stored in `/ThingworxStorage`. If you are planning to configure custom folder locations, run the following command:

- ```
sudo java -classpath $CLASSPATH com.thingworx.security.  
keystore.ThingworxKeyStore encrypt.licensing.password  
<unique license_password> <Password location> <Keystore  
location>
```

9. **Start Tomcat.**

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

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

 **Note**

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

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**.

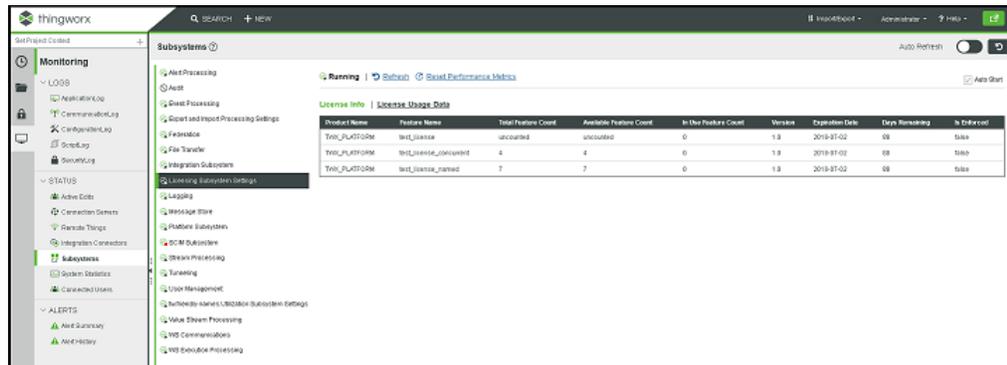
 **Note**

The password, which should not be easily guessed or a known, common password, is recommended to be at least 14 characters in length (minimum 10) 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.



PostgreSQL

Install Java and Apache Tomcat (RHEL)

Note

The steps in this process have been tested with Java 8 update 131. Other versions are not supported and may not work.

1. Download the Java (JDK) RPM 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/d54c1d3a095b44ff2b6607d096fa80163/jdk-8u131-linux-x64.rpm
```

2. Run the Java installer:

```
$ sudo rpm -i jdk-8u131-linux-x64.rpm
```

3. Create the directory and move the JDK:

```
$ sudo mkdir -p /usr/lib/jvm  
$ sudo mv /usr/java/jdk1.8.0_131/ /usr/lib/jvm/
```

4. Set the Java alternatives:

```
$ sudo alternatives --install /usr/bin/java java /usr/lib/jvm/  
jdk1.8.0_131/bin/java 1  
$ sudo alternatives --install /usr/bin/keytool keytool /usr/  
lib/jvm/jdk1.8.0_131/bin/keytool 1
```

5. 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
```

6. Change Owner:

```
$ sudo chown -R root:root /usr/lib/jvm/jdk1.8.0_131/
```

7. Configure master links:

```
$ sudo alternatives --config java
```

 **Note**

Select the option that contains `/usr/lib/jvm/jdk1.8.0_131/bin/java`

```
$ sudo rm /usr/java/latest
$ sudo ln -s /usr/lib/jvm/jdk1.8.0_131 /usr/java/latest
$ sudo ln -s /usr/lib/jvm/jdk1.8.0_131/bin/keytool /usr/bin/keytool
```

 **Note**

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

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

8. Verify Java version:

 **Note**

Your build version may differ.

```
$ java -version
java version "1.8.0_131"
Java(TM) SE Runtime Environment (build 1.8.0_131-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.45-b02, mixed mode)
```

9. Install Tomcat. Download the Tomcat installer:

 **Note**

This 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.

10. Extract the contents:

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

11. 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
```

12. Define environment variables in /etc/environment:

```
$ export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_131
$ export CATALINA_HOME=/usr/share/tomcat8.5/8.5.xx
```

13. Change directory to /usr/share/tomcat8.5/8.5.xx:

```
$ cd /usr/share/tomcat8.5/8.5.xx
```

14. 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
```

15. 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/
```

16. 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
```

 **Note**

Permissions and ownership should be revisited for a production system to increase security on a operating system level.

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

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

18. Create self-signed certificate:

```
$ /usr/lib/jvm/jdk1.8.0_131/jre/bin/keytool -genkey -alias
tomcat8.5 -keyalg RSA
```

19. 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
```

20. 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.

21. Save the file.

22. Modify the shutdown string and protocol used by the SSL Connector in `server.xml`. Open `/usr/share/tomcat8.5/8.5.xx/conf/server.xml` in a text editor (as root) Uncomment the following section:

```
<Connector executor="tomcatThreadPool"
    port="80" protocol="HTTP/1.1"
    connectionTimeout="20000"
    redirectPort="8443" />
-->
```

23. Paste the following directly below the uncommented section:

```
<Connector port="443" protocol="org.apache.coyote.http11.
Http11NioProtocol"
    maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
    keystoreFile="${user.home}/8.5.xx/conf/.keystore" keystorePass=
"<PostgreSQL keystore password>" clientAuth="false"
    sslProtocol="TLS" />
```

24. 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"/>
```

25. Save the file.

 **Note**

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

26. Set up Tomcat as a service to start on boot. First, build JSVC:

```
$ sudo yum install gcc
```

 **Note**

This may already be installed on your system.

```
$ 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 ../..
```

27. Create the Tomcat service file:

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

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

```
[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_131
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 \
-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 \
-catalina.out \
-outfile ${CATALINA_HOME}/logs/
$CATALINA_OPTS \
org.apache.catalina.startup.
Bootstrap

[Install]
WantedBy=multi-user.target

```

29. 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"
```

30. Set Tomcat to run on system start up:

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

 **Note**

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

```

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 only: Go to [Install ThingWorx on page 46](#).
2. PostgreSQL only: Go to [Install and Configure PostgreSQL on page 93](#).

Install and Configure PostgreSQL (RHEL)

The instructions provided below are intended for the PostgreSQL administrator (not the DB host servers).

Note

If you are including the HA layer to your implementation, refer to 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 III 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. Add the PostgreSQL repository to Yum and install.
2. Install PgAdmin III, the PostgreSQL admin tool:

```
$ sudo yum install pgadmin3
```

Note

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

3. Initialize and launch the database:

```
$ sudo /usr/pgsql-9.x/bin/postgresql9.x-setup initdb
```

Set the PostgreSQL service to start on boot:

```
$ sudo chkconfig postgresql-9.x on  
$ sudo service postgresql-9.x start
```

4. Set up the password for the PostgreSQL user:

```
$ sudo passwd postgres
```

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. Set up the PostgreSQL user in psql:

 **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 \(Optional\)](#) and skip the next two steps.

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

 **Note**

The *<unique PostgreSQL password>* value is what you entered above.

7. If using the command line, open the following files and edit as noted. Skip this step if using pgAdmin III.

- `/var/lib/pgsql/9.x/data/postgresql.conf/postgresql.conf`: **Uncomment** `listen addresses` and `port`. The default settings of `localhost` and `5432` are usually sufficient.
- `/var/lib/pgsql/9.x/data/pg_hba.conf`: **Set Method** to `md5`

8. Configure pgAdmin III. Skip this step if you are not using pgAdmin III.

```
$ sudo pgadmin3
```

- In the pgAdmin III GUI, click on **file->Open postgresql.conf**
- Open `/var/lib/pgsql/9.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/9.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 III

9. Restart the PostgreSQL service:

```
$ sudo service postgresql-9.x restart
```

10. Set up PgAdmin III to connect to the database:

```
$ sudo pgadmin3
```

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

Name: PostgreSQL 9.x
Host: localhost
Port: 5432
Service: <blank>
Maintenance DB: postgres
Username: postgres
Password: <unique PostgreSQL password as set previously >
Store password: Checked
Group: Servers

12. Click **OK**.

13. 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 PostgreSQL9.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).

 **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.

- e. Click **OK**.

Configure PostgreSQL Database Located on a Separate Server than ThingWorx (Optional)

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

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 \(9.4\)](#).

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 a folder named `ThingworxPostgresqlStorage` on the drive that the `ThingworxStorage` folder is located (in the root directory by default).

Note

If you create the folder using the `-d<databasename>` command, you do not have to use the PostgreSQL user.

Note

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.

Note

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 8.3.x download from the PTC site. If necessary, change the file name to the version you are using.

```
$ sudo unzip MED-61111-CD-083_ThingWorx-Platform-Postgres-8-3-x.zip
$ cd install
```

- 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
```
- Execute the script.

thingworxPostgresDBSetup Script Options

Option	Parameter	Default	Description	Example
t or -T	server	localhost	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	tablespace	thingworx	Name of the PostgreSQL tablespace	-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-username	twadmin	User name that has permissions to write to the database.	-u twadmin

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.bat` 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
```

3. Execute the script.

Note

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

thingworxPostgresSchemaSetup Script Options

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

thingworxPostgresSchemaSetup Script Options (continued)

Option	Parameter	Default	Description	Example
			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: <ul style="list-style-type: none"> • 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

Configure platform-settings.json

1. Create a folder named ThingworxPlatform at the root of the drive where Tomcat was installed or as a system variable.

 **Note**

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.

 **Note**

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 124](#).

 **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 (Optional)

If you want to provide added security encryption for the PostgreSQL database settings in the `platform-settings.json` file, you can perform the following steps.

 **Note**

This encryption process is optional.

 **Note**

You must have Java installed and on your path. You must have PostgreSQL installed and recall the password.

1. Create a working directory where you will perform this process and copy the `Thingworx.war` to that location.

 **Note**

ThingWorx downloads are available in [PTC Software Downloads](#).

2. Unzip the `Thingworx.war`.
3. Open a command prompt, `cd` to your working directory, and set your `CLASSPATH` by doing the following:

```
CLASSPATH= /  
<password_setup location>/WEB-INF/lib/logback-core-1.2.3.jar:/  
<password_setup location>/WEB-INF/lib/logback-classic-1.2.3.  
jar:<password_setup location>/WEB-INF/lib/thingworx-common-  
8.3.0-bxx.jar
```
4. Open `/ThingworxPlatform/platform-settings.json` and change the password value to `'encrypt.db.password'`. For example:

```
"password": "encrypt.db.password"
```

 **Note**

Since the PostgreSQL admin password should not be included in the `platform-settings.json`, adding the `encrypt.db.password` string for the password signals the ThingWorx platform to look up the encrypted password in the keystore when it is encountered.

5. To create a key store with the PostgreSQL password encrypted inside, run the following command. In the second argument, enter your unique PostgreSQL password:

```
$ sudo java -classpath $CLASSPATH com.thingworx.security.  
keystore.ThingworxKeyStore encrypt.db.password <unique  
postgres_password>
```

Note

By default, the password is stored in `/ThingworxPlatform`. The keystore is stored in `/ThingworxStorage`. If you are planning to configure custom folder locations, run the following command:

```
$ sudo java -classpath $CLASSPATH
com.thingworx.security.keystore.ThingworxKeyStore
encrypt.db.password <unique postgres_password> <Password location>
<Keystore location>
```

6. After you have created the encrypted password, remove the updates to the CLASSPATH.

Installing the PostgreSQL Client Package and PostgreSQL User (optional)

In order to issue PostgreSQL commands from the client machine to the PostgreSQL server, do so from a PostgreSQL user. The `postgresql-client-9.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 104](#).

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`:

```
$ sudo chown tomcat8.5:tomcat8.5 /ThingworxStorage
/ThingworxBackupStorage /ThingworxPlatform
$ sudo chmod 775 /ThingworxStorage /ThingworxBackupStorage
/ThingworxPlatform
```

 **Note**

Without these permissions, the server will fail to start. For more information, reference [this article](#).

3. If you have not already done so, obtain the `Thingworx.war` file.

 **Note**

ThingWorx downloads are available in [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 10 characters long. Reference [platform-settings.json Configuration Details on page 124](#) for more information on placement. See [Passwords on page](#) for additional information on setting passwords.

 **Note**

Do not copy and paste the sample below, as it may cause bad formatting in your `platform-settings.json`. Instead, click the link below 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. Password must be a minimum of 10 characters, check the following:

- The password setting exists in platform-settings.json
- The password is valid (10 or more characters)
- The platform-settings.json file is formatted correctly - bad formatting could lead to errors

7. Configure licensing:

- Open the platform-settings.json file and add the following to the PlatformSettingsConfig section (reference [platform-settings.json Configuration Options on page 124](#) 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).

8. Encrypt the license server password.

 **Note**

This step is optional, but it is the recommended best practice to encrypt the password.

- a. Create a working directory where you will perform this process, such as `<password_setup_location>`, and copy the `Thingworx.war` file to that location.
- b. Unzip the `Thingworx.war`.
- c. Open a command prompt, `cd` to your working directory, and set your `CLASSPATH` by doing the following:

 **Note**

The file versions are based on ThingWorx 8.3, and may need to be changed if you are using a different version. Replace **xx** with the build number you are using.

-

```
export CLASSPATH= /<password_setup_location>/WEB-INF/lib/
thingworx-platform-common-8.3.0-bxx.jar:
/<password_setup_location>/WEB-INF/lib/slf4j-api-1.7.25.
jar:
```

```
/<password_setup_location>/WEB-INF/lib/logback-core-
1.2.3.jar:
/<password_setup_location>/WEB-INF/lib/logback-classic-
1.2.3.jar:
/<password_setup_location>/WEB-INF/lib/thingworx-common-
8.3.0-bxx.jar
```

- d. In your command shell, enter 'java -version'. It should respond with a Java version.
- e. Stop Tomcat.

```
$ sudo service tomcat8.5 stop
```
- f. Open /ThingworxPlatform/platform-settings.json and change the LicensingConnectionSettings password value to 'encrypt.licensing.password'. For example, "password": "encrypt.licensing.password",. This password signals the ThingWorx platform to look up the encrypted licensing password in the keystore when it is encountered.
- g. To create a key store with the licensing password encrypted inside, run the following command. In the second argument, enter your unique license server password:

- ```
sudo java -classpath $CLASSPATH com.thingworx.security.
keystore.ThingworxKeyStore encrypt.licensing.password
<unique license_password>
```

---

### Note

By default, the password is stored in /ThingworxPlatform. The keystore is stored in /ThingworxStorage. If you are planning to configure custom folder locations, run the following command:

- ```
sudo java -classpath $CLASSPATH com.thingworx.security.
keystore.ThingworxKeyStore encrypt.licensing.password
<unique license_password> <Password location> <Keystore
location>
```

9. Start Tomcat.

(UBUNTU)

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

(RHEL)

```
$ sudo systemctl start tomcat
```

Note

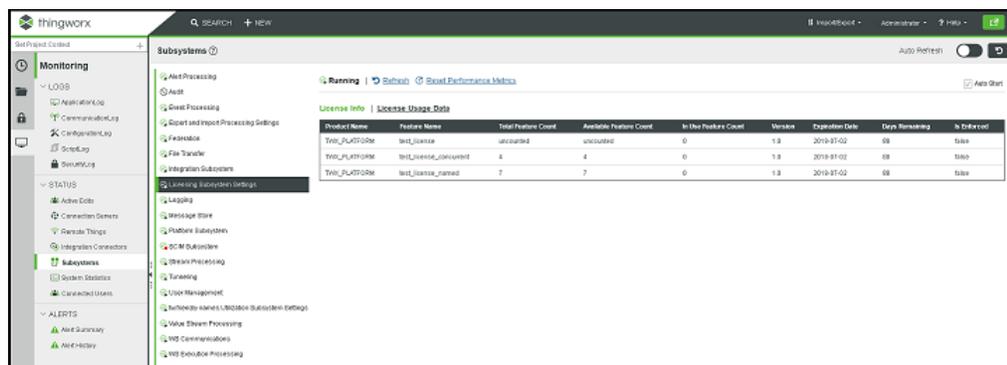
Verify that a license file (`successful_license_capability_response.bin`) is created in the `ThingworxPlatform` folder.

- To launch ThingWorx, go to `http://<servername>:<port>/Thingworx` in a Web browser.
- Change the default password:
 - In Composer, select **Administrator > Change Password**.
 - 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 (minimum 10) and should include a mix of uppercase and lowercase letters, numbers, and special characters.

- Delete the initial password from the `platform-settings.json` file.
- Select **Done**.
 - (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 Name	Feature Name	Total Feature Count	Available Feature Count	In Use Feature Count	Version	Duration Date	Days Remaining	Is Deleted
THW_PU_TPSDR	test_license	4	4	0	1.8	2019-01-02	99	false
THW_PU_TPSDR	test_license_named	7	7	0	1.8	2019-01-02	99	false

5

Amazon RDS Installation

Note

Before performing this section, perform the steps in the following sections (based on your OS):

- [Install Java and Apache Tomcat \(Windows\) on page 18](#)
 - [Install Java and Apache Tomcat \(Ubuntu\) on page 39](#)
 - [Install Java and Apache Tomcat \(RHEL\) on page 76](#)
-

1. Follow the steps outlined in the [Amazon RDS installation guide](#). The steps below provide supplemental guidance when you are ready to configure the **Specify DB Details** page in AWS.
2. On the **Specify DB Details** page, specify the following information:
 - a. In the **DB Engine Version** field, select the latest 9.4 PostgreSQL version available. (9.4.9 in this example).
 - b. In the **DB Instance Class** field, select the appropriate class.

Note

m3.2xlarge is recommended for production use.

- c. In the **Mutli-AZ Deployment** field, select **Yes** if you have an HA environment.
- d. Note the **DB Instance Identifier** and **Master Username** for later use.

Specify DB Details

Instance Specifications

DB Engine	postgres
License Model	postgresql-license
DB Engine Version	9.4.9
DB Instance Class	db.m3.xlarge — 4 vCPU, 15 GiB RAM
Multi-AZ Deployment	Yes
Storage Type	Provisioned IOPS (SSD)
Allocated Storage*	100 GB
Provisioned IOPS	1000

Settings

DB Instance Identifier*	TWXDBINSTANCE
Master Username*	pgadmin
Master Password*
Confirm Password*

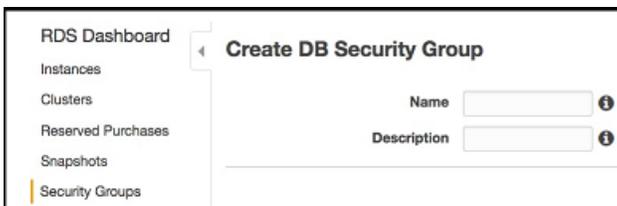
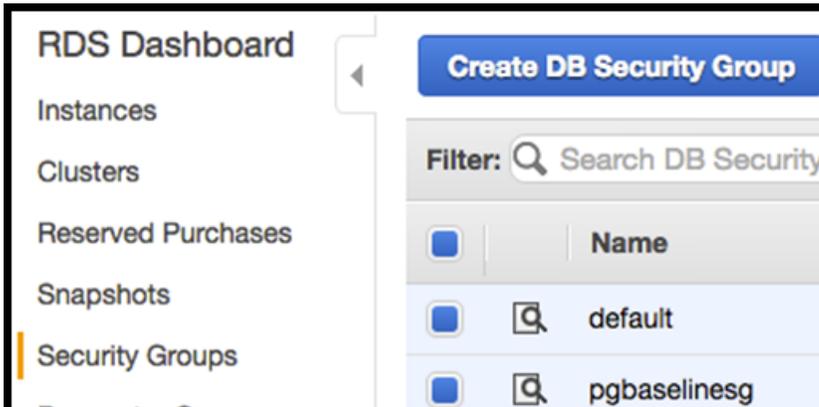
3. On the RDS Dashboard, click **Parameter Groups**>**Create DB Parameter Group**.
4. In the **Parameter Group Family** field, create a **Group Name** and **Description** for PostgreSQL database configuration later.

Create Parameter Group

To create a Parameter Group, select a Parameter Group Family, then name and description.

Parameter Group Family	postgres9.4	<i>i</i>
Group Name		<i>i</i>
Description		<i>i</i>

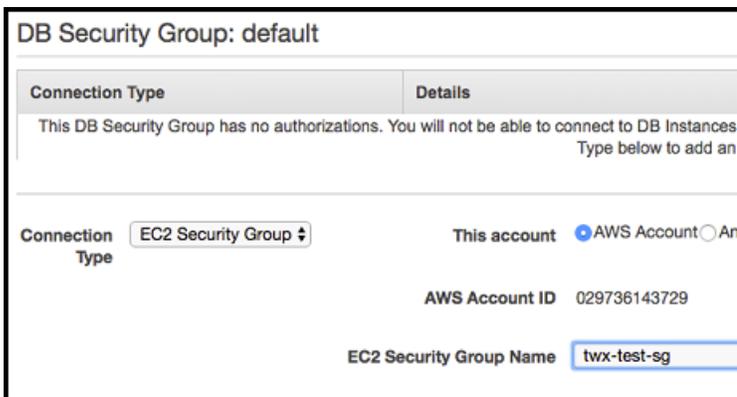
5. On the RDS Dashboard, click **Security Groups**.
6. Create a DB security group to control the DB access later.



7. 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.

 **Note**

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.



8. On the **Configure Advanced Settings** page, specify the following information:

-
- a. In the **Network & Security** section, 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.

- b. 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.

- c. In the **DB Parameter Group** field, select the name of the parameter group created previously.
- d. In the **Enable Encryption** field, select **Yes** if necessary.
- e. In the **Backup, Monitoring, and Maintenance** sections, select the appropriate options per your organizational needs.

Configure Advanced Settings

Network & Security ↻

VPC: vpc-SprintTest (vpc-3e75875b) ▾

Subnet Group: default-vpc-3e75875b ▾

Publicly Accessible: Yes ▾

Availability Zone: No Preference ▾

VPC Security Group(s): Create new Security Group
34363378_S_P1941870494_2016-09-
GEM_SECURITY_GROUP (VPC)
InfoSys-twx-7.2.x-latest-h2 (VPC)

Database Options

Database Name: thingworx

Database Port: 5432

DB Parameter Group: default.postgres9.4 ▾

Option Group: default.postgres-9-4 ▾

Copy Tags To Snapshots:

Enable Encryption: No ▾

Backup

Backup Retention Period: 7 ▾ days

Backup Window: No Preference ▾

Monitoring

Enable Enhanced Monitoring: Yes ▾

Monitoring Role: Default ▾

Granularity: 60 ▾ second(s)

I authorize RDS to create the IAM role rds-monitoring-role.

Maintenance

Auto Minor Version Upgrade: Yes ▾

Maintenance Window: No Preference ▾

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	<p>There are three options:</p> <ul style="list-style-type: none"> • 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

Configure platform-settings.json

1. Create a folder named ThingworxPlatform at the root of the drive where Tomcat was installed or as a system variable.

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

 **Note**

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`

 **Note**

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.

(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 124](#).

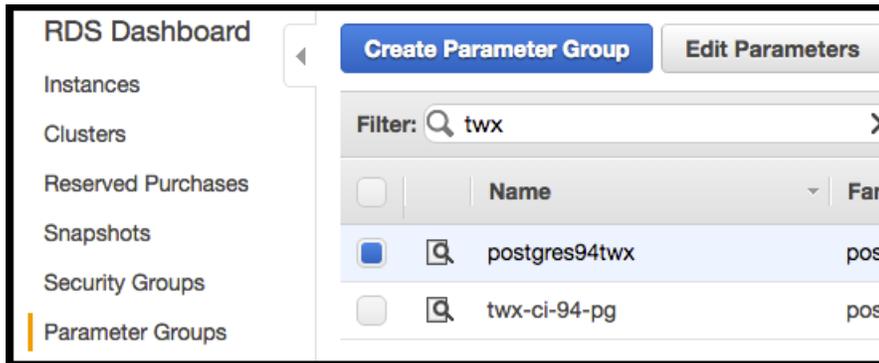
 **Note**

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.

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.



Note

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 http://www.postgresql.org/docs/current/static/runtime-config-resource.html#GUC-SHARED-BUFFERS
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/

Setting	Configuration	Description
		current/static/runtime-config-resource.html#GUC-WORK-MEM
<code>maintenance_work_mem</code>	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
<code>effective_cache_size</code>		Should be set to an estimate of how much memory is available for disk caching by the OS and within the database. It is recommended to set this to half the memory available on the machine.
<code>checkpoint_segments</code>		Depends on the size of the PostgreSQL box. Should set to 32/64/128/256, depending upon machine size.
<code>checkpoint_completion_target</code>		If the checkpoint_segments is changed from the default value of 3, change this to 0.9.
<code>ssl_renegotiation_limit</code>		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 13.

6

Installation Appendices

Apache Tomcat Java Option Settings	122
platform-settings.json Configuration Details	124
Installation Troubleshooting	153

Apache Tomcat Java Option Settings

Mandatory Settings

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

Mandatory Settings (continued)

Setting	Description
-Xms3072m (for a system with 4GB of memory)	<p>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.</p> <p> Note The amount of memory needs to be tuned depending on the actual environment.</p>
-Xmx3072m (for a system with 4GB of memory)	<p>Tells the JVM to limit the maximum memory to the Tomcat process. This should be set to 75% of the available system memory.</p> <p> 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.</p> <p> 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.</p>

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

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"
    },
    "AdministratorUserSettings": {
      "InitialPassword": "changeme"
    },
    "HASettings": {
      "CoordinatorConnectionTimeout": 15000,
      "CoordinatorHosts": "127.0.0.1:2181",
      "CoordinatorMaxRetries": 3,
      "CoordinatorRetryTimeout": 1000,
      "CoordinatorSessionTimeout": 90000,
      "CoordinatorZNode": "/HALeadershipCoordinator",
```

```

        "LoadBalancerBase64EncodedCredentials":
"QWRtaW5pc3RyYXRCvcjphZG1pbG=="
    },
    "LicensingConnectionSettings": {
        "username": "<username>",
        "password": "<password>",
        "timeout": "60"
    }
},
"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,
        "tableLockTimeout": 10000,
        "testConnectionOnCheckout": false,
        "unreturnedConnectionTimeout": 0
    },
    "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,

```


platform-settings.json Options

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 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 <code>sso-settings.json</code> 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

Basic Settings (continued)

Setting	Default	Description
		repository should be located on a shared file system where all ThingWorx servers have access.
HTTPRequestHeaderMaxLength	2000	The maximum allowable length for HTTP Request Headers values.
HTTPRequestParameterMaxLength	2000	The maximum allowable length for HTTP Request Parameter values.
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).

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
CoordinatorConnectionTimeout	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.
CoordinatorRetryTimeout	1000	How long to wait (in milliseconds) for each retry attempt.
CoordinatorSessionTimeout	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	<p>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 <code>/<anyTextHere></code>. 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 <code>/HASystemA</code> for TWX1 and TWX2, and it is set to <code>/HASystemB</code> for TWX3 and TWX4.</p>
LoadBalancerBase64EncodedCredentials	QWRtaW5p-c3RyYXRvcjphZG1pb-g==	<p>The Base64-encoded credentials for the HA Load Balancer, in the format of <code><user>:<unique password></code>.</p> <p> Note</p> <p>You can use any utility that Base64 encodes the matching <code><user>:<unique password></code> string used in your load balancer setup.</p>

Administrator User Settings

Setting	Default	Description
InitialPassword	n/a	The initial Administrator password that is required to log into ThingWorx for the first time. Must be at least 10 characters. See Passwords on page for more information.

Licensing Connection Settings

Setting	Default	Description
username	n/a	PTC Support site username
password	n/a	PTC Support site password
timeout (in seconds)	60	After the timeout period, the following error is logged in the Application Log: License Server could not process request

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.

(continued)

Setting	Default	Description
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).
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).
	20	The number of threads to

(continued)

Setting	Default	Description
numberOfProcessingThreads		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 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.
idleConnectionTestPeriod	6	Time period (in seconds) where connections will be tested so that idle connections won't be

H2PersistenceProviderPackage (continued)

Setting	Default	Description
		<p>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 number of seconds.</p> <p> Note</p> <p>If you are experiencing “No connection to model provider” errors, review this setting. Compare to firewall defaults. Lowering the default will alleviate disconnection issues.</p>
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

H2PersistenceProviderPackage (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, maxIdleTimeExcessConnections 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.
maxStatementsPerConnection	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

H2PersistenceProviderPackage (continued)

Setting	Default	Description
		generally performed by helper threads that don't hold contended locks. Spreading these operations over multiple threads can significantly improve performance by allowing multiple operations to be performed simultaneously.
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.

H2PersistenceProviderPackage (continued)

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.
numberOfProcessingThreads	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.
numberOfProcessingThreads	5	The number of threads to use when processing properties.
scanRate	5	The rate (in milliseconds) at which to check the buffer status.

H2PersistenceProviderPackage (continued)

Setting	Default	Description
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).
numberOfProcessingThreads	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

PostgresPersistenceProviderPackage (continued)

Setting	Default	Description
		giving up.
acquireRetryDelay	10000	The time (in milliseconds) ThingWorx will wait between acquire attempts.
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.
idleConnectionTestPeriod	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
		<p>connect to PostgreSQL.</p> <p> Note</p> <p>If the default schema name is changed (from public), you must add <code><databasename>?currentSchema=<name of schema></code>. For example, if the schema name is <code>mySchema</code>, it would be:</p> <pre>jdbc:postgres:// <DBServer>:<DB Port>/ <databasename>? currentSchema= mySchema</pre> <p> Note</p> <p>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.</p>
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	<i><unique password></i>	The password used to log into the database.
testConnectionOnCheckout	false	If true, an operation will be performed at every connection checkout to verify that the connection is valid.
unreturnedConnectionTimeout	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.
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

PostgresPersistenceProviderPackage (continued)

Setting	Default	Description
		(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 (cannot change for Neo4j).
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 queue (will be rejected after that).
maximumWaitTime	10000	Number of milliseconds the system waits before flushing the value stream buffer.
numberOfProcessingThreads	5	The number of processing threads (cannot change for Neo4j).
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

PostgresPersistenceProviderPackage (continued)

Setting	Default	Description
		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).
numberOfProcessingThreads	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

MssqlPersistenceProviderPackage (continued)

Setting	Default	Description
		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	<p>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 number of seconds.</p> <p> 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.</p>
initialPoolSize	5	Initial number of database connections created and maintained within a pool

MssqlPersistenceProviderPackage (continued)

Setting	Default	Description
		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.
maxIdleTimeExcess-Connections	300	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 minPoolSize if, following a spike, the load level diminishes and Connections acquired are no longer needed. If maxIdleTime is set, maxIdleTimeExcess-Connections should be

MssqlPersistenceProviderPackage (continued)

Setting	Default	Description
		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. Spreading these operations over multiple threads can significantly improve performance by allowing multiple operations to be performed simultaneously.
password	< <i>unique password</i> >	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

MssqlPersistenceProviderPackage (continued)

Setting	Default	Description
		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 TWSHEMA schema and is used for 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.
numberOfProcessingThreads	5	The number of processing threads (cannot change

MssqlPersistenceProviderPackage (continued)

Setting	Default	Description
		for Neo4j).
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).
numberOfProcessingThreads	5	The number of processing threads (cannot change for Neo4j).
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

MssqlPersistenceProviderPackage (continued)

Setting	Default	Description
		property entries to queue (will be rejected after that).
numberOfProcessingThreads	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)
<p>After installing Tomcat and deploying the Thingworx.war file, Composer doesn't load with a 404 error Application not found</p>	<ul style="list-style-type: none">• 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) <p> Note</p> <p>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.</p> <ul style="list-style-type: none">• Verify that the URL accessed is correct <code>http://<server>:<port>/Thingworx</code> (not <code>http://<server>:<port>/ThingWorx</code>)

Issue	Possible Resolution(s)
	<ul style="list-style-type: none"> • If a 404 page not found error is encountered in a RHEL environment after ThingWorx installation, verify the following steps as well: <ul style="list-style-type: none"> ○ Verify that the JDK is present in the <code>/usr/lib/jvm/</code> 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 <code>JAVA_HOME</code> environment variable has the JDK path. For example, <pre>JAVA_HOME = /usr/lib/jvm/jdk1.8.144</pre>
<p>Problem deploying thingworx.war.</p>	<p>Verify that the ThingworxStorage/extensions/web-inf folder contains the licensing libraries (DLL files).</p>
<p>The following error is received when deploying ThingWorx:</p> <pre>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)</pre>	<p>The max file size in the Tomcat web.xml file must be increased (default is 50MB). This file is located at :</p> <pre><path to Tomcat>\Apache Software Foundation\Tomcat 8.5\webapps\manager\WEB-INF</pre> <ol style="list-style-type: none"> 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.

Issue	Possible Resolution(s)
<pre>at org.apache.catalina.connector.Re quest.parseParts(Request.ja va:2871</pre>	
<p>The following error message is received when importing a PTC licensed extension:</p> <pre>is licensed but cannot find feature in license.bin file</pre>	<p>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.</p>
<p>The following error message is received when attempting to undeploy ThingWorx:</p> <pre>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)</pre>	<p>Remove <code>-Djava.library.path</code> from Tomcat's Java configuration before undeployment.</p>

Issue	Possible Resolution(s)
<p>An error message similar to the following is seen in the ConfigurationLog.log:</p> <pre> 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 ANALYSIS </pre>	<p>The log message verifies if there is an issue with the license file.</p>
<p>An error message similar to the following is thrown while the platform is starting:</p> <pre> 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] ===== </pre>	<p>The license file may have been opened/edited/saved in a browser. Download the license file again, rename it to <code>license_capability_response.bin</code>, and place in ThingworxPlatform folder without editing or saving it.</p>

Issue	Possible Resolution(s)
<pre> ===== 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. </pre>	