



Installing ThingWorx 8.2

Version 1.0

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

Document Revision History

Revision Date	Version	Description of Change
February 2, 2018	1.0	Initial version for 8.2



Installing ThingWorx

Document Revision History.....	1
Installing ThingWorx Core.....	4
Prerequisites	4
Upgrading.....	4
Database Options: PostgreSQL, Microsoft SQL Server, SAP HANA, or H2.....	4
Sizing Guide.....	4
High Availability Option	4
Installing ThingWorx for the First Time: H2 or PostgreSQL on Windows	5
Installing and Configuring PostgreSQL (Windows)	9
Installing PostgreSQL and Creating a New User Role in PostgreSQL (Windows).....	9
Configuring PostgreSQL Database Located on a Separate Server than Thingworx (Windows)	10
Configuring and Executing the PostgreSQL Database Script (Windows).....	10
Configuring and Executing the Model/Data Provider Schema Script (Windows).....	12
Configuring platform-settings.json (Windows).....	13
Encrypting the PostgreSQL Password (Windows).....	15
Installing ThingWorx (Windows).....	16
Installing ThingWorx for the First Time: PostgreSQL or H2 on Ubuntu	20
Installing Oracle Java and Apache Tomcat (Ubuntu)	20
Installing and Configuring PostgreSQL (Ubuntu)	28
Installing PostgreSQL and Creating a New User Role in PostgreSQL (Ubuntu)	28
Configuring PostgreSQL Database Located on a Separate Server from Thingworx (Ubuntu)	30
Enabling PostgreSQL to Listen for all Connections (Linux)	31
Configuring and Executing the PostgreSQL Database Script (Ubuntu).....	32
Configuring and Executing the Model/Data Provider Schema Script (Ubuntu)	33
Configuring platform-settings.json (Ubuntu).....	34
(OPTIONAL) Encrypting the PostgreSQL Password (Ubuntu)	34
Installing ThingWorx (Ubuntu).....	36
Installing ThingWorx for the First Time: PostgreSQL or H2 on Red Hat Enterprise Linux (RHEL).....	40
Configuring Ulimit Settings	40
Configuration File Example	40
Installing Oracle Java and Apache Tomcat (RHEL)	41
Installing and Configuring PostgreSQL (RHEL)	47
Installing PostgreSQL and Creating a New User Role in PostgreSQL (RHEL)	47

Configuring PostgreSQL Database Located on a Separate Server than ThingWorx (RHEL).....	51
Enabling PostgreSQL to Listen for all Connections (Linux)	52
On Linux installations of PostgreSQL, there is an additional configuration step required to configure the PostgreSQL server to listen for connections.	52
(OPTIONAL) Encrypting the PostgreSQL Password (RHEL)	52
Installing ThingWorx (RHEL).....	53
PostgreSQL Installation and Configuration with Amazon RDS	58
Installing PostgreSQL RDS Instance.....	58
Creating a New User Role in PostgreSQL (RDS)	60
Configuring and Executing the Model/Data Provider Schema Script (RDS)	61
Configuring platform-settings.json (RDS)	62
Installing and Configuring PostgreSQL DB Host Servers (RDS)	62
Installing ThingWorx (RDS)	63
Appendix A: Tomcat Java Option Settings	64
Appendix B: Sample platform-settings.json.....	66
Appendix C: platform-settings.json Options.....	69
Appendix D: Metrics Reporting.....	84
Appendix E: Installing PostgreSQL Client Package and PostgreSQL User	85
Appendix F: Licensing Troubleshooting	86

Installing ThingWorx

ThingWorx is currently supported on Windows, Ubuntu, Amazon EC2, and Red Hat Enterprise Linux.

Prerequisites

Prerequisite software includes Apache Tomcat and Oracle Java. PostgreSQL is also required if you are not using H2, MS SQL Server, or SAP HANA for your database. If you are installing ThingWorx for the first time, this document provides step-by-step installation instructions for your environment.

Upgrading

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

Database Options: PostgreSQL, Microsoft SQL Server, SAP HANA, or H2

With ThingWorx 8.1, you can use PostgreSQL (with an optional High Availability layer), SAP HANA, Microsoft SQL Server, or H2 for your data solution. If you are upgrading to 8.1, the following download package options are available when obtaining the thingworx.war from [PTC Software Downloads](#):

- H2: **Thingworx-Platform-H2-8.1.0**
- PostgreSQL/HA: **Thingworx-Platform-Postgres-8.1.0**
- Microsoft SQL Server: **Thingworx-Platform-Mssql-8.1.0**
- SAP HANA: **ThingWorx-Platform-Hana-8.1.0**

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 in the Help Center.

Sizing Guide

To gain a better understanding about which database best suits your requirements, reference the [ThingWorx Sizing Guide](#).

High Availability Option

With ThingWorx 7.0 and later, 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](#).

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

Installing ThingWorx for the First Time: H2 or PostgreSQL on Windows

NOTE: If you are using PostgreSQL, Microsoft SQL Server, or SAP HANA for your database, additional steps are required. If you are installing H2, steps are included below to skip all PostgreSQL sections.

Installing Oracle Java and Apache Tomcat (Windows)

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

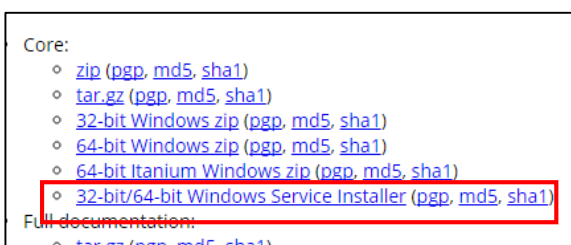
NOTE: Refer to the [System Requirements and Compatibility Matrix](#) document for version requirements.

2. Visit the [Tomcat website](#) to download the **32-bit/64-bit Windows Service Installer (pgp, md5, sha1)**.

NOTE: Refer to the [System Requirements and Compatibility Matrix](#) 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.

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



4. Click **I Agree**.

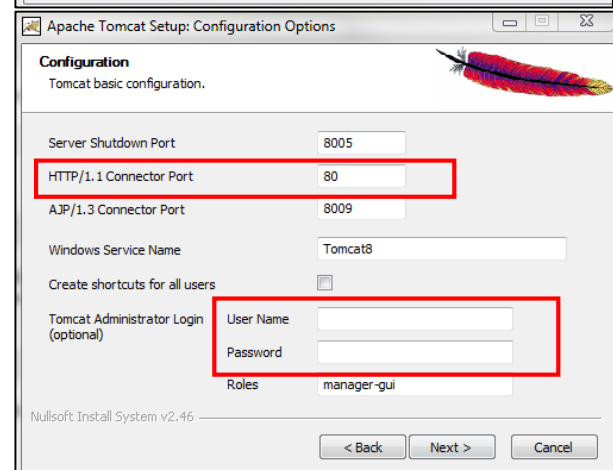
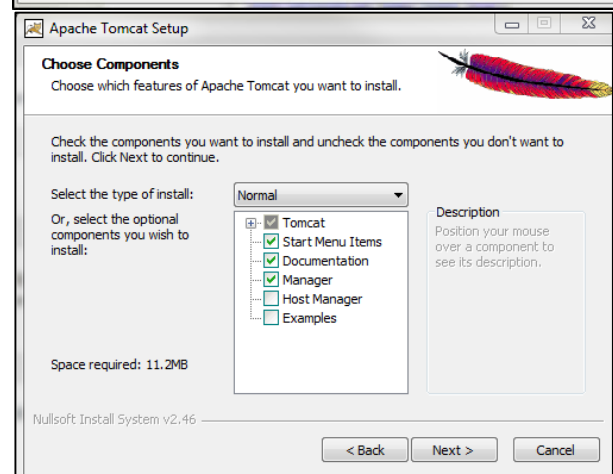
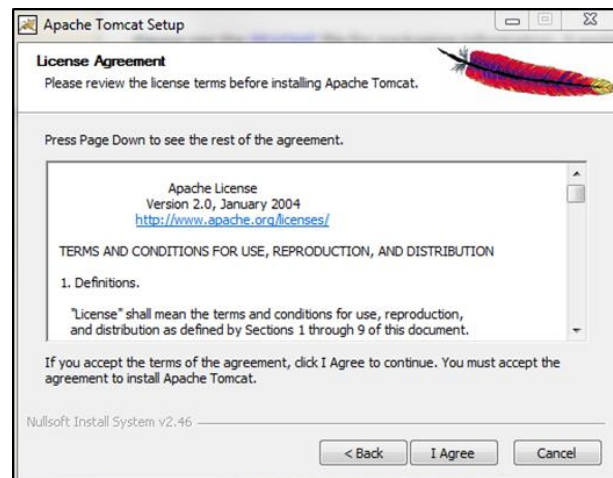
5. In the **Components** section, use the default settings.

6. Click **Next**.

7. In the **HTTP/1.1 Connector Port** field, type **80** (or other available port).

8. In the **Tomcat Administrator Login** fields, enter a *<Tomcat user name>* and a unique, secure password for Tomcat administration.

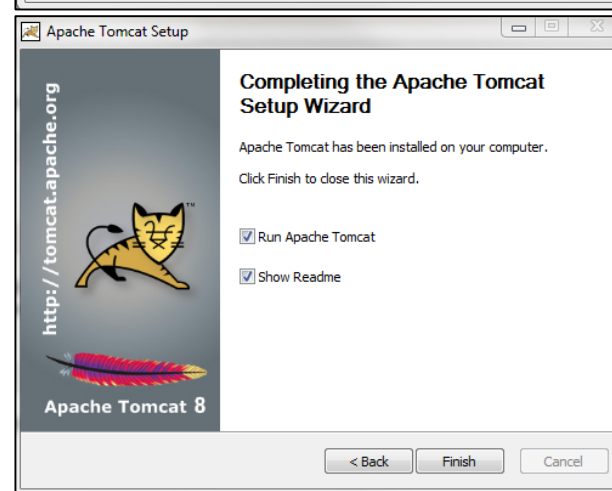
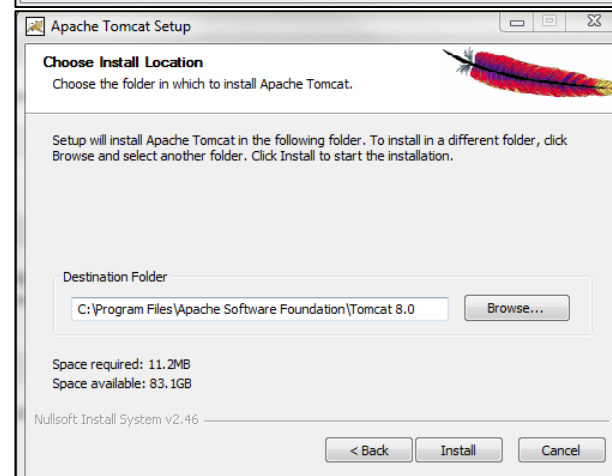
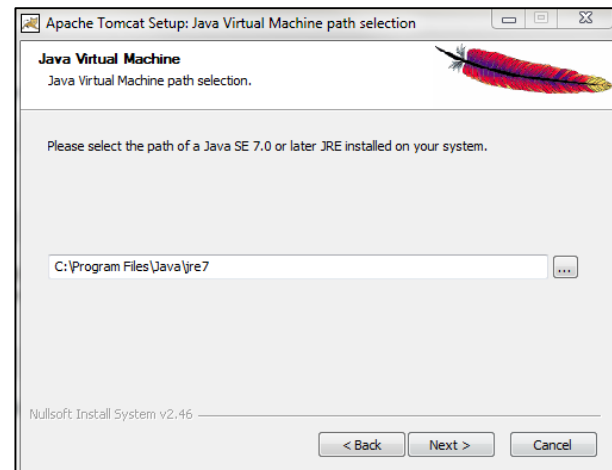
9. Click **Next**.



10. Enter the path to the proper 64-bit Java installation directory.
11. Click **Next**.

12. Click **Install**.

13. Click **Finish**.



14. Click **Start>Configure Tomcat**.
15. Open the **Java** tab.
16. 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

Djava.library.path example:

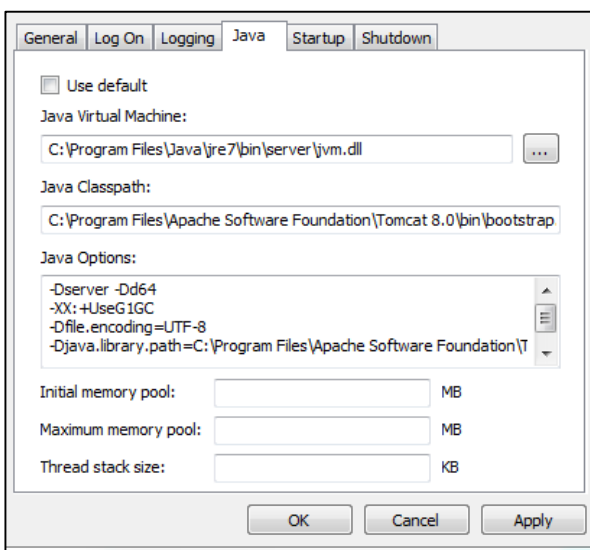
```
-Djava.library.path=C:\Program
Files\Apache Software
Foundation\Tomcat
8.0\webapps\Thingworx\WEB-
INF\extensions
```

NOTE: For more information on these options and for additional options for hosted and/or public-facing environments, refer to the [Appendix: Tomcat Java Option Settings](#).

17. Clear any values in the **Initial memory pool** and **Maximum memory pool** fields.
18. Click **OK**.

19. OPTIONAL STEP: If you want to increase the default cache settings that affect static file caching, add the following line within the <context></context> tags in the \$TOMCAT_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
```

20. If you are not installing PostgreSQL, skip to the [Installing ThingWorx](#) section.
21. If you are using a database other than PostgreSQL and H2, refer to the following:
 - a. SAP HANA: [Getting Started with SAP HANA and ThingWorx Guide](#)
 - b. Microsoft SQL Server: [Getting Started with MS SQL Server and ThingWorx Guide](#)

Installing and Configuring 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](#).

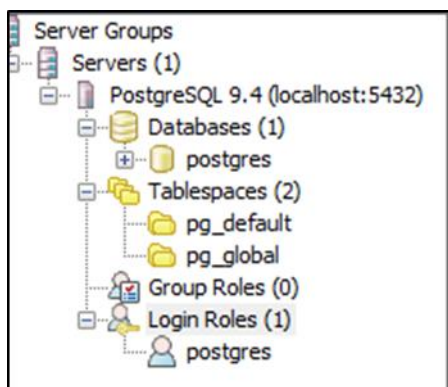
Installing PostgreSQL and Creating a New User Role in PostgreSQL (Windows)

1. Download and install the appropriate version of PostgreSQL from the following site:
<http://www.postgresql.org/download/>
- pgAdmin III Tool
 - 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, multithreaded query, and data editing tools and support for all PostgreSQL object types.

2. Open PostgreSQL using pgAdmin III.

3. Create a new user role:
 - a. Right click **PostgreSQL9.4** (localhost:5432).
 - b. Select **NewObject>New Login Role**. On the **Properties** tab, in the **Role name** field, enter the *<PostgreSQL user role name>* for PostgreSQL administration.
 - c. On the **Definition** tab, in the **Password** field, enter a unique and secure password for PostgreSQL administration (you will be prompted to enter it twice).

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



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

4. Click **OK**.

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

Configuring PostgreSQL Database Located on a Separate Server than Thingworx (Windows)

By default, the PostgreSQL server is installed in a locked-down state. On Windows, the server will only listen for connections from the local machine. In order to get ThingWorx to talk 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\9.4\data**. This location will be referred to as **<PGDATA>** in this guide.

Modify the **<PGDATA>/pg_hba.conf** file and add the following lines based on your desired configuration:

If you want to allow all IPv4 addresses to connect:	host all all 0.0.0.0/0 md5
If you want to allow only a specific IPv4 address to connect (Replace <ipAddress> with the IP address of the machine making the connection):	host all all <ipAddress>/32 md5
If you want to allow all IPv6 addresses to connect:	host all all ::0/0 md5
If you want to allow only a specific IPv6 address to connect (Replace <ipv6Address> with the appropriate address)	host all all <ipv6Address>/128 md5

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.

For additional information about configuring the **pg_hba.conf** file, see the [Official PostgreSQL Documentation \(9.4\)](#).

Configuring and Executing the PostgreSQL Database Script (Windows)

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

1. Add the **<postgres-installation>/bin** folder to your system path variable.
2. Create a directory named **ThingworxPostgresqlStorage** on the drive that Thingworx Storage is located (in the root directory by default).

NOTE: If you create with 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.

The folder must have appropriate ownership and access rights. The folder 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 **thingworxPostgresDBSetup.bat** from the ThingWorx software download package.
4. Configure the script. Reference the configuration options in the table below.

Various parameters such as **server**, **port**, **database**, **tablespace**, **tablespace location** and **thingworxusername** can be configured in the script, depending on your requirements. Execute this script with the **--help** option for usage information.

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

```
thingworxPostgresDBSetup -a postgres -u <PostgreSQL user role name> -l
C:\ThingworxPostgresqlStorage
```

where **<PostgreSQL user role name>** is the user role name that you entered in previous steps.

5. Execute the script. Once executed, this creates a new database and tablespace with a default PostgreSQL installation in the PostgreSQL instance installed on the localhost.

NOTE: You may need to run the command prompt as admin.

thingworxPostgresDBSetup.bat 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

Option	Parameter	Default	Description	Example
-h or -H	tablespace	thingworx	Name of the PostgreSQL tablespace.	-h localhost
-l or -L	tablespace_location	/ThingworxPostgresqlStorage	Required. Location in the file system where the files representing database objects are stored.	-l or -L
-a or -A	adminusername	postgres	Administrator Name	-a postgres
-u or -U	thingworxusername	twadmin	User name that has permissions to write to the database.	-u twadmin

Configuring and Executing the Model/Data Provider Schema Script (Windows)

To set up the PostgreSQL model/data provider schema, the **thingworxPostgresSchemaSetup.bat** 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.
2. Configure the script. Reference the configuration options in the table below.

Various parameters such as **server**, **port**, **database**, **username**, **schema**, and **option** can be configured in the script depending on the requirements. Execute this script with **--help** option for usage information.

3. Execute the script.
NOTE: You may be prompted to provide your password three times.

thingworxPostgresSchemaSetup.bat Script Options

Option	Parameter	Default	Description	Example
-h or -H	server	localhost For RDS: rds-host	IP or host name of the database NOTE: For RDS: Hostname or IP of RDS PostgreSQL instance	-h localhost For RDS: -h rds-host
-p or -P	port	5432	Port number of PostgreSQL	-p 5432

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

Configuring platform-settings.json (Windows)

1. To use the default ThingworxPlatform configuration directory, create a folder called **ThingworxPlatform** at the root of the drive where Tomcat was installed. Alternatively, if you want 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 Appendix C: platform-settings.json Options and Appendix B: platform-settings.json sample.

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.

Encrypting the PostgreSQL Password (Windows)

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

NOTE This encryption process is optional.

Prerequisites

- You must have Java installed and on your path.
 - You must have PostgreSQL installed and know the password.
1. Create a working directory where you will perform this process, such as `C:\<password_setup_location>` (Windows), and copy the Thingworx.war to that location.
 2. Unzip the Thingworx.war.
 3. Open a command prompt, cd to your working directory, and set your CLASSPATH by doing the following:
 - a. Go to **Control Panel > System Properties > Environment Variables**.
 - b. Create a new environment variable: PG_PW_UTIL
`C:\<password_setup_location>\WEB-INF\lib\slf4j-api-1.7.21.jar;C:\<password_setup_location>\WEB-INF\lib\logback-core-1.0.13.jar;C:\<password_setup_location>\WEB-INF\lib\logback-classic-1.0.13.jar;C:\<password_setup_location>\WEB-INF\lib\thingworx-common-<release-version>.jar`
 - c. Add the variable to the CLASSPATH.
`CLASSPATH`
`<don't touch existing classpath>; %PG_PW_UTIL%`
 - d. 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:
`java com.thingworx.security.keystore.ThingworxKeyStore encrypt.db.password <unique postgres_password>`

In the second argument, enter your unique PostgreSQL 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. Once you have created the encrypted password, remove the updates to the CLASSPATH.

Installing ThingWorx (Windows)

1. If you have not already done so, create a folder called **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. NOTE: If you are performing a disconnected installation, you do not need to add to the **platform-settings.json** file. Refer to the [Licensing Guide for disconnected sites](#) and skip this step.

Add your PTC support site **username**, **password**, and **timeout** (optional) to the **platform-settings.json**:

- a. Open the **platform-settings.json** file and add the following to the **PlatformSettingsConfig** section (reference [Appendix B: Sample platform-settings.json](#) for more information on placement).

```
"LicensingConnectionSettings":{
  "username":"PTC Support site user name",
  "password":"PTC Support site password",
  "timeout":"60"
}
```

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).

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

Open a case with Technical Support if you are doing the manual disconnected mode of licensing and have any questions or need assistance with generating a license.

3. 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 C:\ (Windows), and copy the Thingworx.war 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:
 - i. Go to **Control Panel > System Properties > Environment Variables**.
 - ii. Create a new environment variable:

```
PG_PW_UTIL
"C:\<location where zip file is extracted>\WEB-INF\lib\thingworx-platform-common-8.1.0-bxx.jar;
C:\<location where zip file is extracted>\password_setup\WEB-INF\lib\slf4j-api-1.7.21.jar;
C:\<location where zip file is extracted>\WEB-INF\lib\logback-core- 1.0.13.jar;
C:\<location where zip file is extracted>\WEB-INF\lib\logback-classic-1.0.13.jar;
C:\<location where zip file is extracted>\WEB-INF\lib\thingworx-common-8.1.0-bxx.jar"
```

- iii. Add the variable to the CLASSPATH:


```
CLASSPATH ; %PG_PW_UTIL%
```
- iv. In your command shell, enter 'java -version'. It should respond with a Java version.

- d. 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.
- e. 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>
```

4. Locate the appropriate **Thingworx.war** file.

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

5. Copy the **Thingworx.war** file and place it in the following location of your Tomcat installation:

\Apache Software Foundation\Tomcat 8.5\webapps

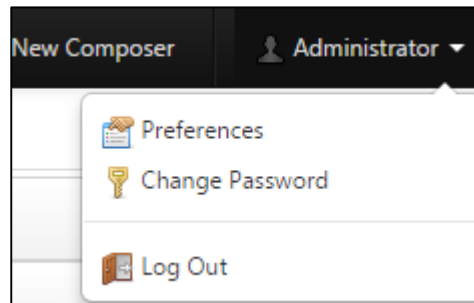
6. To launch ThingWorx, go to **<servername>/Thingworx** in a Web browser.

Use the following login information:

Login Name: Administrator

Password: trUf6yuz2?_Gub

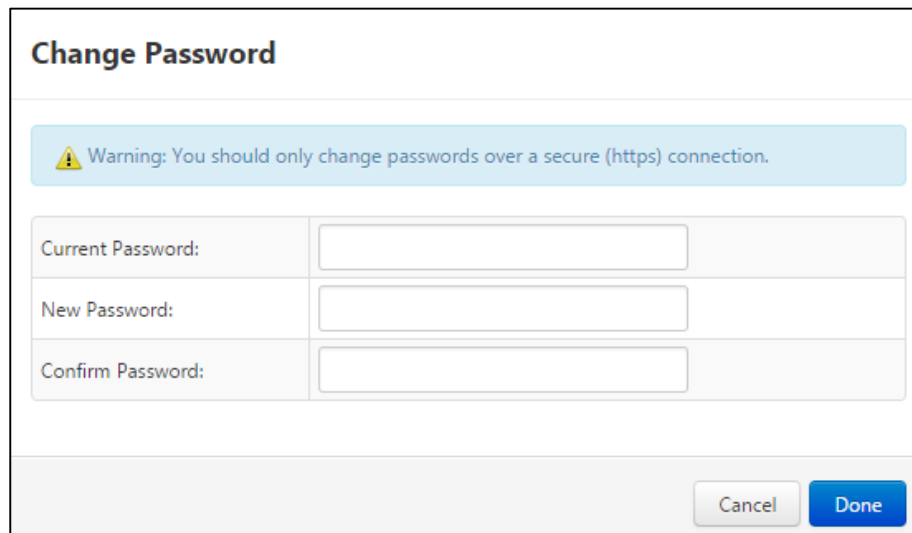
7. Change the default password.
 - a. In Composer, click **Administrator>Change Password**.



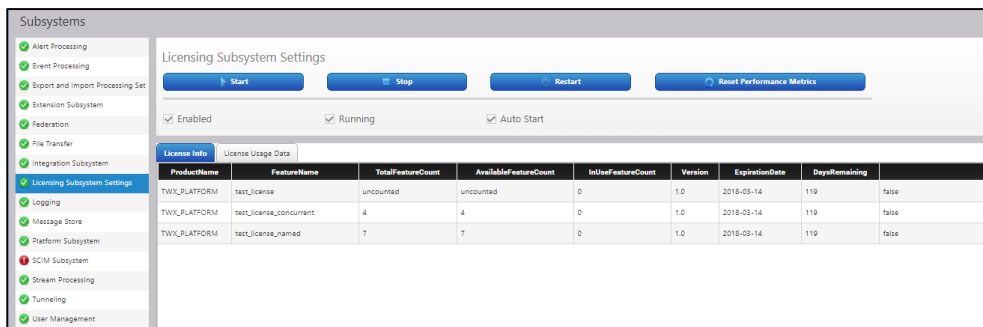
- b. In the **Change Password** window, enter **Current Password**, **New Password**, and **Confirm Password**.

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

- c. Click **Done**.

A screenshot of the 'Change Password' dialog box. The title bar says 'Change Password'. Below the title, there is a light blue warning box with a yellow triangle icon and the text: 'Warning: You should only change passwords over a secure (https) connection.' Below the warning box, there are three rows of input fields. The first row is labeled 'Current Password:', the second 'New Password:', and the third 'Confirm Password:'. Each label is followed by a text input field. At the bottom right of the dialog, there are two buttons: 'Cancel' and 'Done'.

- To determine the status of the license, administrators can open the Licensing Subsystem Settings in ThingWorx to confirm the list of features (licensed entities) included with the license. If there are no licensed entities present, you are in limited mode.



Installing ThingWorx for the First Time: PostgreSQL or H2 on Ubuntu

Oracle Java, and Apache Tomcat, and PostgreSQL (if using PostgreSQL) must be installed prior to installing ThingWorx. Refer to the [System Requirements and Compatibility Matrix](#) for specific version requirements.

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

Installing Oracle Java and Apache Tomcat (Ubuntu)

- Update Ubuntu packages:

```
$ sudo apt-get update
```

- 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: This version of ThingWorx has 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** :

```
$ 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.

NOTE: 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-b14)  
Java HotSpot(TM) 64-Bit Server VM (build 24.75-b04, mixed mode)
```

12. Download Apache Tomcat:

```
$ wget http://archive.apache.org/dist/tomcat/tomcat-8.5/v8.5.23/bin/apache-tomcat-8.5.23.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.

NOTE: This version of ThingWorx has been tested with Tomcat 8.5.23. Other versions are not supported and may not work.

13. Extract tar file:

```
$ tar -xf apache-tomcat-8.5.23.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 addgroup --system tomcat8.5 --quiet --force-badname  
$ sudo adduser --system --home /usr/share/tomcat8.5/ --no-create-home --ingroup tomcat8.5 --disabled-password --shell --force-badname/bin/false tomcat8.5  
$ sudo chown -R tomcat8.5:tomcat8.5 /usr/share/tomcat8.5  
$ sudo mv apache-tomcat-8.5.23 /usr/share/tomcat8.5/8.5.23
```

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

```
$ export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_131  
$ export CATALINA_HOME=/usr/share/tomcat8.5/8.5.23
```

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
```

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
```

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

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

22. Modify the shutdown string and protocol used by the SSL Connector in **\$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}/8.5.23/conf/.keystore"
keystorePass="<keystore password> " clientAuth="false"
sslProtocol="TLS" />
```

Comment out the following non-SSL Connector:

```
<!--
<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 assure 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
```

23. Define a user in **\$CATALINA_HOME/conf/tomcat-users.xml**:

```
<user username="<Tomcat user name> " password="<Tomcat password> "
roles="manager"/>
```

24. Determine uid of tomcat8.5 user:

```
$ id -u tomcat8.5
110
```

25. Using this number, create an ID file in **/etc/authbind/byuid/**

NOTE: The number returned may be different than this example. These examples use 110. Change this to what was returned in the previous step.

```
$ sudo touch /etc/authbind/byuid/110
```

Edit this file and paste in the following:

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

26. Change owner and access permissions of **/etc/authbind/byuid/110**:

```
$ sudo chown tomcat8.5:tomcat8.5 /etc/authbind/byuid/110
$ sudo chmod 700 /etc/authbind/byuid/110
```

27. Modify **\$CATALINA_HOME/bin/startup.sh** to always use authbind:

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

Add the following to the end of the file:

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

28. In **/etc/init.d**, create **tomcat8.5** file:

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

Edit the file and enter the following contents:

```
CATALINA_HOME=/usr/share/tomcat8.5/8.5.23

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
```

29. 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
```

30. Setup Tomcat as a service to start on boot. First, build JSVC:

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

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

```
$ cd /usr/share/tomcat8.5/8.5.23/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 ../..
```

31. Create the Tomcat service file:

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

32. Open **/etc/systemd/system/tomcat8.5 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/tomcat8.5/8.5.23
Environment=CATALINA_BASE=/usr/share/tomcat8.5/8.5.23
Environment=CATALINA_OPTS=

ExecStart=/usr/share/tomcat8.5/8.5.23/bin/jsvc \
    -Dcatalina.home=${CATALINA_HOME} \
    -Dcatalina.base=${CATALINA_BASE} \
    -Djava.awt.headless=true -
Djava.net.preferIPv4Stack=true -Dserver -Dd64 -XX:+UseNUMA \
    -XX:+UseConcMarkSweepGC -
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/tom
cat-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
```

NOTE: If you get an error, create a new file in the tomcat **/bin** file named **setenv.sh**

```
CATALINA_OPTS="$CATALINA_OPTS -
Djava.library.path=/usr/share/tomcat8.5/8.5.23/webapps/Thingworx/WE
B-INF/extensions"
```

33. OPTIONAL STEP: If you want to increase the default cache settings that affect static file caching, add the following line within the <context></context> tags in the \$TOMCAT_HOME/conf/context.xml file:

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

34. If you are not installing PostgreSQL, skip to the [Installing ThingWorx](#) section.

Installing and Configuring 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](#).

Installing PostgreSQL and Creating a New User Role in PostgreSQL (Ubuntu)

1. Download and install the appropriate version of PostgreSQL.

In Ubuntu 14.4, the PostgreSQL repository can be added allowing the application to be installed directly from the package manager:

```
$ sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/
trusty-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-9.4 -y
```

NOTE: This version of ThingWorx has been tested with PostgreSQL versions 9.4.5 through 9.4.10. Other versions are not supported and may not work.

2. Install PgAdmin III, the PostgreSQL admin tool:

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

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, multithreaded query, and data editing tools and support for all PostgreSQL object types.

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 passwd postgres
```

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. Set up the PostgreSQL user in psql:

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

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

6. Configure pgAdmin III:

```
$ sudo pgadmin3
```

- * In the pgAdmin III GUI, click on file->Open postgresql.conf
- * Open /etc/postgresql/9.4/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/9.4/main/pg_hba.conf
- * Double-click on the database 'all' line with address 127.0.0.1/32
- * Set Method to md5
- * Double-click on the database 'all' line with address 1/128
- * Set Method to md5
- * Click OK
- * Save and exit
- * Close pgAdmin III

7. Restart the PostgreSQL service:

```
$ sudo service postgresql restart
```

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

```
$ sudo pgadmin3
```

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

Fill out the following:

Name: PostgreSQL 9.4
Host: localhost
Port: 5432
Service: <blank>
Maintenance DB: postgres
Username: postgres
Password: <unique PostgreSQL password as set previously >
Store password: Checked
Group: Servers

10. Click **OK**.

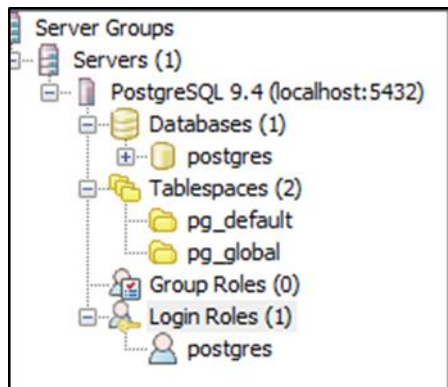
11. Create a new user role:

- a. Right click **PostgreSQL9.4(localhost:5432)**.
- b. Select **NewObject>NewLogin Role**. On the **Properties** tab, enter a name in the **Role name** field.
- c. On the **Definition** tab, in the **Password** field, enter a unique password (you will be prompted to enter it twice).

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

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

12. Click **OK**.



Configuring PostgreSQL Database Located on a Separate Server from Thingworx (Ubuntu)

By default, the PostgreSQL server is installed in a locked-down state. On Linux, the server will only listen for connections from the local machine, and only from the postgres user. In order to get ThingWorx to talk 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 need to know where your PostgreSQL data directory resides to perform 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).

On Ubuntu, when installed via apt-get, the configuration files are located at **/etc/postgresql/9.4/main/**. This location is referred to as <PGDATA> in this guide.

Modify the <PGDATA>/**pg_hba.conf** file and add the following lines based on your desired configuration:

If you want to allow all IPv4 addresses to connect:

If you want to allow all IPv4 addresses to connect	host all all 0.0.0.0/0 md5
If you want to allow only a specific IPv4 address to connect (Replace <ipAddress> with the IP address of the machine making the connection)	host all all <ipAddress>/32 md5
If you want to allow all IPv6 addresses to connect:	host all all ::0/0 md5
If you want to allow only a specific IPv6 address to connect (Replace <ipv6Address> with the appropriate address)	host all all <ipv6Address>/128 md5

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.

For additional information about configuring the pg_hba.conf file, see the Official PostgreSQL Documentation (9.4)

Enabling PostgreSQL to Listen for all Connections (Linux)

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

1. In the <PGDATA>/**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.

Configuring and Executing the PostgreSQL Database Script (Ubuntu)

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

1. Create a directory named **ThingworxPostgresqlStorage** on your Thingworx Storage drive.

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

NOTE: You must specify the **-l** option to a path that exists. For example, **-l ThingworxPostgresqlStorage**. The script does not create the folder for you.

As shown above, this folder needs have appropriate ownership and access rights. It should be owned by the PostgreSQL user and have the read, write, and execute assigned to the owner.

NOTE: If you create with the **-d<dbname>**, you do not have to use the PostgreSQL user.

2. Obtain and open **thingworxPostgresDBSetup.sh** from the ThingWorx software download.
3. Configure the script. Reference the configuration options in the table below.

NOTE: This example uses the 8.0.0 download from the PTC site. If necessary, change the file name to the version you are using.

```
$ sudo unzip MED-61111-CD-074_F000_ThingWorx-Platform-Postgres-8-0-0.zip
$ cd install
```

Various parameters such as **server**, **port**, **database**, **tablespace**, **tablespace location** and **thingworxusername** can be configured in the script, depending on the requirements. Execute this script with the **--help** option for usage information.

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

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

thingworxPostgresDBSetup.sh 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

Option	Parameter	Default	Description	Example
-l or -L	tablespace_location	/ThingworxPostgresqlStorage	Required. Location in the file system where the files representing database objects are stored. *	-l or -L
-a or -A	adminusername	postgres	Administrator Name	-a postgres
-u or -U	thingworxusername	twadmin	User name that has permissions to write to the database.	-u twadmin

Configuring and Executing the Model/Data Provider Schema Script (Ubuntu)

To set up the PostgreSQL model/data provider schema, the **thingworxPostgresSchemaSetup.sh** 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.sh** from the ThingWorx software download package.
2. Configure the script. Reference the configuration options in the table below or execute the script with **--help** option for usage information. The script can be run with the default parameters as:

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

thingworxPostgresSchemaSetup.sh Script Options

Option	Parameter	Default	Description	Example
-h or -H	server	localhost For RDS: rds-host	IP or host name of the database For RDS: Hostname or IP of RDS PostgreSQL instance	-h localhost For RDS: -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
-u or -U	username	twadmin	Username to update the database schema	-u twadmin

Option	Parameter	Default	Description	Example
-o or -O	option	all	There are three options: all : Sets up the model and data provider schemas into the specified database. model : Sets up the model provider schema into the specified database. data : Sets up the data provider schema into the specified database.	-o data

Configuring platform-settings.json (Ubuntu)

1. Create a folder called **ThingworxPlatform** at the root (for example, **/ThingworxPlatform** or as a system variable (for example, **THINGWORX_PLATFORM_SETTINGS=/data/ThingworxPlatform**).

```
$ sudo mkdir /ThingworxPlatform
```

2. Copy the **platform-settings.json** file from the install package to the **/ThingworxPlatform** folder. From the directory containing the JSON file:

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

The JSON file contains default settings. Refer to alternate configuration options in:
[Appendix C: platform-settings.json Options](#) and
[Appendix B: platform-settings.json sample](#).

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

(OPTIONAL) Encrypting the PostgreSQL Password (Ubuntu)

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

Note: This encryption process is optional.

Prerequisites

- You must have Java installed and on your path.
- You must have PostgreSQL installed and know 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.
2. Unzip the Thingworx.war.
3. Open a command prompt, cd to your working directory, and set your CLASSPATH:

```
$ export CLASSPATH= WEB-INF/lib/logback-core-1.0.13.jar:WEB-INF/lib/logback-classic-1.0.13.jar:./WEB-INF/lib/thingworx-common-  
<release-version>.jar
```

4. Open `/ThingworxPlatform/platform-settings.json` and change the password value to `'encrypt.db.password'`.

For example:

```
"<unique PostgreSQL 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:

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

NOTE: In the second argument, enter your PostgreSQL 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>
```

Installing ThingWorx (Ubuntu)

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
```

3. 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
```

4. NOTE: If you are performing a disconnected installation, you do not need to add to the **platform-settings.json** file. Refer to the [Licensing Guide for disconnected sites](#) and skip this step.

Add your PTC support site **username**, **password**, and **timeout** (optional) to the **platform-settings.json**:

Open the **platform-settings.json** file and add the following to the **PlatformSettingsConfig** section (reference [Appendix B: Sample platform-settings.json](#) for more information on placement).

```
"LicensingConnectionSettings":{
  "username":"PTC Support site user name",
  "password":"PTC Support site password",
  "timeout":"60"
}
```

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).

Open a case with Technical Support if you are doing the manual disconnected mode of licensing and have any questions or need assistance with generating a license.

Further information on obtaining a ThingWorx disconnected site license through our [License Management site](#) can be found in our [Licensing guide](#).

5. 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 C:\ (Windows), and copy the Thingworx.war to that location.
- b. Unzip the Thingworx.war.
- c. Open a command prompt, cd to your working directory, and set your CLASSPATH with the following:

```
export CLASSPATH=
<location where zip file is extracted>\WEB-
INF\lib\thingworx-platform-common-8.1.0-bxx.jar;
<location where zip file is
extracted>\password_setup\WEB-INF\lib\slf4j-api-
1.7.21.jar;
<location where zip file is extracted>\WEB-
INF\lib\logback-core- 1.0.13.jar;
<location where zip file is extracted>\WEB-
INF\lib\logback-classic-1.0.13.jar;
<location where zip file is extracted>\WEB-
INF\lib\thingworx-common-8.1.0-bxx.jar
```

i. Add the variable to the CLASSPATH:

- ii. In your command shell, enter 'java -version'. It should respond with a Java version.

- d. 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.
- e. 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>
```

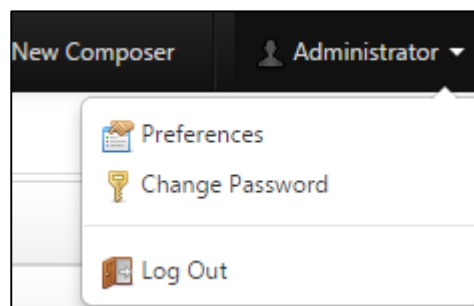

6. Start Tomcat to deploy the ThingWorx web application:

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

7. You should now be able to connect to ThingWorx Composer through a web browser at **https://localhost/Thingworx**

Username: Administrator
Password: trUf6yuz2?_Gub

8. Change the default password.
 - a. In Composer, click **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 include a mix of uppercase and lowercase letters, numbers, and special characters.

- c. Click **Done**.

Change Password

Warning: You should only change passwords over a secure (https) connection.

Current Password:	<input type="password"/>
New Password:	<input type="password"/>
Confirm Password:	<input type="password"/>

Cancel Done

9. To determine the status of the license, administrators can open the Licensing Subsystem Settings in ThingWorx to confirm the list of features (licensed entities) included with the license. If there are no licensed entities present, you are in limited mode.

Subsystems

- Alert Processing
- Event Processing
- Export and Import Processing Set
- Extension Subsystem
- Federation
- File Transfer
- Integration Subsystem
- Licensing Subsystem Settings**
- Logging
- Message Store
- Platform Subsystem
- SCM Subsystem
- Stream Processing
- Tunneling
- User Management

Licensing Subsystem Settings

Start Stop Restart Reset Performance Metrics

☒ Enabled ☒ Running ☒ Auto Start

Product Name	Feature Name	Total Feature Count	Available Feature Count	In Use Feature Count	Version	Expiration Date	Days Remaining	Is Valid
TWPL_PLATFORM	test_license	uncounted	uncounted	0	1.0	2018-03-14	119	false
TWPL_PLATFORM	test_license_concurrent	4	4	0	1.0	2018-03-14	119	false
TWPL_PLATFORM	test_license_named	7	7	0	1.0	2018-03-14	119	false

Installing ThingWorx for the First Time: PostgreSQL or H2 on Red Hat Enterprise Linux (RHEL)

Oracle Java, Apache Tomcat, and PostgreSQL (if using PostgreSQL) must be installed prior to installing ThingWorx.

NOTE: This version of ThingWorx has been tested on RHEL 7.1. Other RHEL versions are not supported and may not work.

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.

Installing Oracle Java and Apache Tomcat (RHEL)

1. Download the Java (JDK) RPM file from Oracle's website, or run the following:

NOTE: This version of ThingWorx has 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.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:

```
$ 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:

```
$ wget https://archive.apache.org/dist/tomcat/tomcat-8/v8.5.23/bin/apache-tomcat-8.5.23.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-8.5.23.tar.gz
```

11. Move Tomcat to **/usr/share/tomcat8.5**:

```
$ sudo mkdir -p /usr/share/tomcat8.5  
$ sudo mv apache-tomcat-8.5.23 /usr/share/tomcat8.5/8.5.23
```

12. Define environment variables in **/etc/environment**:

```
$ export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_131  
$ export CATALINA_HOME=/usr/share/tomcat8.5/8.5.23
```

13. Change directory to **/usr/share/tomcat8.5/8.5.23**:

```
$ cd /usr/share/tomcat8.5/8.5.23
```

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 -R640 conf
```

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
```

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/tomcat8.5/8.5.23/conf/
$ sudo chown root:tomcat8.5 /usr/share/tomcat8.5/8.5.23/conf/.keystore
$ sudo chmod 640 /usr/share/tomcat8.5/8.5.23/conf/.keystore
```

19. Uncomment the Manager element in **context.xml** to prevent sessions from persisting across restarts.

Open **/usr/share/tomcat8.5/8.5.23/conf/context.xml** in a text editor (as root) and remove the '`<!--`' before '`<Manager pathname="" />`' and the '`-->`' after

20. Save the file.

21. Modify the shutdown string and protocol used by the SSL Connector in **server.xml**:

Open **/usr/share/tomcat8.5/8.5.23/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" />
-->
```

Paste in this section directly below:

```
<Connector port="443"
protocol="org.apache.coyote.http11.Http11NioProtocol"
maxThreads="150" SSLEnabled="true" scheme="https"
secure="true"
    keystoreFile="${user.home}/8.5.23/conf/.keystore"
    keystorePass="<PostgreSQL keystore password>"
    clientAuth="false" sslProtocol="TLS" />
```

22. Define an Apache Manager user in tomcat-users.xml:

Open **/usr/share/tomcat8.5/8.5.23/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"/>
```

23. Save the file.

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

24. 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. If so, continue.

```
$ cd /usr/share/tomcat8.5/8.5.23/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 ../..
```

25. Create the Tomcat service file:

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

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/tomcat8.5/8.5.23
Environment=CATALINA_BASE=/usr/share/tomcat8.5/8.5.23
Environment=CATALINA_OPTS=

ExecStart=/usr/share/tomcat8.5/8.5.23/bin/jsvc \
    -Dcatalina.home=${CATALINA_HOME} \
    -Dcatalina.base=${CATALINA_BASE} \
    -Djava.awt.headless=true -
Djava.net.preferIPv4Stack=true -Dserver -Dd64 -XX:+UseNUMA \
    -XX:+UseConcMarkSweepGC -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
```

NOTE: If you get an error, create a new file in the tomcat **/bin** file named **setenv.sh**

```
CATALINA_OPTS="$CATALINA_OPTS -
Djava.library.path=/usr/share/tomcat8.5/8.5.23/webapps/Thingworx/WEB-
INF/extensions"
```


26. 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
```

27. Start the Tomcat service and test:

```
$ sudo systemctl start tomcat
```

You should now be able to connect to the Tomcat server by entering **https://localhost** in a browser.

29. If you are not installing PostgreSQL, skip to the [Installing ThingWorx](#) section.

Installing and Configuring PostgreSQL (RHEL)

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

NOTE: This guide assumes 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).

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

Installing PostgreSQL and Creating a New User Role in PostgreSQL (RHEL)

1. Add the PostgreSQL repository to Yum and install:

```
$ sudo rpm -Uvh
http://download.postgresql.org/pub/repos/yum/9.4/redhat/rhel-7-
x86_64/pgdg-redhat94-9.4-3.noarch.rpm
$ sudo yum install postgresql94 postgresql94-server postgresql94-
contrib
```

NOTE: This version of ThingWorx has been tested with PostgreSQL 9.4. Other versions are not supported and may not work.

2. Install PgAdmin III:

```
$ 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-9.4/bin/postgresql94-setup initdb
```

4. Set the PostgreSQL service to start on boot:

```
$ sudo chkconfig postgresql-9.4 on
$ sudo service postgresql-9.4 start
```

5. Set up the password for the PostgreSQL user:

```
$ sudo passwd postgres
```

Enter the password for the PostgreSQL user.

Take note of this password; you will use it 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:

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

NOTE: The *<unique postgres password>* value is the same as in the previous step.

7. If the PostgreSQL database is not located on the same server as ThingWorx, then refer to the section [Configuring PostgreSQL Database Located on a Separate Server than Thingworx \(Ubuntu\)](#) and skip the next two steps.

8. Configure postgresql.conf with pgAdmin 3.

```
$ sudo pgadmin3
```

In the pgAdmin 3 GUI, click on **file->Open postgresql.conf**

- * Open **/var/lib/pgsql/9.4/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

9. Configure **pg_hba.conf** with pgAdmin 3.

- * Click on **file->Open pg_hba.conf**
- * Open **/var/lib/pgsql/9.4/data/pg_hba.conf**
- * Double-click on the line with address **127.0.0.1/32**
- * Set Method to **md5**
- * Double-click on the line with address **1/128**
- * Set Method to **md5**
- * Click **OK**
- * Save and exit
- * Close pgAdmin 3

9. Restart the PostgreSQL service:

```
$ sudo service postgresql-9.4 restart
```

10. Set up pgAdmin 3 to connect to the database.

```
$ sudo pgadmin3
```

Click the plug **Add a connection to a server** in the top left corner.

Fill out the following:

Name: PostgreSQL 9.4

Host: localhost

Port: 5432

Service: <blank>

Maintenance DB: postgres

Username: postgres

Password: <unique PostgreSQL password as set in step above>

Store password: Checked

Group: Servers

Click **OK**

11. Create a new user role:

Right click **PostgreSQL9.4 (localhost:5432)**.

Note: It may be possible to activate some extensions. Click **Databases** and select **postgres** in the main window. A dialog displays. Click **Fix it!**

Select **NewObject>New Login Role**.

On the **Properties** tab, in the **Role name** field, enter a user role name.

On the **Definition** tab, enter the unique PostgreSQL password twice.

Click **OK**.

Close pgAdmin 3.

12. Click **OK**.

13. Create the **ThingworxPostgresqlStorage** and **/ThingworxPlatform** directories:

```
$ sudo mkdir /ThingworxPostgresqlStorage
$ sudo chmod 775 /ThingworxPostgresqlStorage
$ sudo chown postgres:postgres /ThingworxPostgresqlStorage/
$ sudo mkdir /ThingworxPlatform
$ sudo chmod 775 /ThingworxPlatform
$ sudo chown tomcat8.5:tomcat8.5 /ThingworxPlatform
```

14. Download the ThingWorx installer from the PTC downloads page:

<https://support.ptc.com/appserver/auth/it/esd/index.jsp>

Unzip and open a terminal in the ***/installer/install** directory.

15. Execute the PostgreSQL Database Script:

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

16. Execute the Model/Data Provider Schema Script:

```
$ sh thingworxPostgresSchemaSetup.sh
```

NOTE: When prompted, enter the password that you entered in previous steps.

17. Startup configuration of platform-settings.json:

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

NOTE: Refer to [Appendix C: platform-settings.json Options](#) for information about the configuration options.

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.

Configuring PostgreSQL Database Located on a Separate Server than ThingWorx (RHEL)

By default, the PostgreSQL server is installed in a locked-down state. On Linux, the server will only listen for connections from the local machine, and only from the postgres user. In order to get ThingWorx to talk 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 need to know where your PostgreSQL data directory resides to perform 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).

On RHEL, when installed via apt-get, the configuration files are located at /var/lib/pgsql/9.4/data/ This location is referred to as <PGDATA> in this guide.

Modify the <PGDATA>/pg_hba.conf file and add the following lines based on your desired configuration:

If you want to allow all IPv4 addresses to connect	host all all 0.0.0.0/0 md5
If you want to allow only a specific IPv4 address to connect (Replace <ipAddress> with the IP address of the machine making the connection)	host all all <ipAddress>/32 md5
If you want to allow all IPv6 addresses to connect	host all all ::0/0 md5
If you want to allow only a specific IPv6 address to connect (Replace <ipv6Address> with the appropriate address)	host all all <ipv6Address>/128 md5

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.

For additional information about configuring the `pg_hba.conf` file, see the Official PostgreSQL Documentation (9.4)

Enabling PostgreSQL to Listen for all Connections (Linux)

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

1. In the `<PGDATA>/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.

(OPTIONAL) Encrypting the PostgreSQL Password (RHEL)

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

Prerequisites

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

1. Create a working directory where you will be performing this process, such as `~/<password_setup>`, and copy the `Thingworx.war` to that location.
2. Unzip the `Thingworx.war`.
3. Open a command prompt, `cd` to your working directory, and set your `CLASSPATH` to the following:

```
$ export CLASSPATH= WEB-INF/lib/slf4j-api-1.7.21.jar:WEB-INF/lib/logback-core-1.0.13.jar:WEB-INF/lib/logback-classic-1.0.13.jar:./WEB-INF/lib/thingworx-common-<release-version>.jar
```
4. Open `/ThingworxPlatform/platform-settings.json` and change the password value to `'encrypt.db.password'`.
For example, `"<unique 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:

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

NOTE: In the second argument, enter your PostgreSQL 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 com. -classpath $CLASSPATH
thingworx.security.keystore.ThingworxKeyStore encrypt.db.password
<unique postgres_password> <Password location> <Keystore location>
```

Installing ThingWorx (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
```

3. Move **Thingworx.war** to the Tomcat **/webapps** directory:

Change your working directory to the **installer** folder in the unzipped ThingWorx download package and run these commands:

```
$ sudo mv Thingworx.war /usr/share/tomcat8.5/8.5.23/webapps/
$ sudo chown tomcat8.5:tomcat8.5
/usr/share/tomcat8.5/8.5.23/webapps/Thingworx.war
$ sudo chmod 775
/usr/share/tomcat8.5/8.5.23/webapps/Thingworx.war
```

4. NOTE: If you are performing a disconnected installation, you do not need to add to the **platform-settings.json** file. Refer to the [Licensing Guide for disconnected sites](#) and skip this step.

Add your PTC support site **username**, **password**, and **timeout** (optional) to the **platform-settings.json**:

- a. Open the **platform-settings.json** file and add the following to the **PlatformSettingsConfig** section (reference [Appendix B: Sample platform-settings.json](#) for more information on placement).

```
"LicensingConnectionSettings":{
  "username":"PTC Support site user name",
  "password":"PTC Support site password",
  "timeout":"60"
}
```

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).

Open a case with Technical Support if you are doing the manual disconnected mode of licensing and have any questions or need assistance with generating a license.

Further information on obtaining a ThingWorx disconnected site license through our [License Management site](#) can be found in our [Licensing guide](#). Further information on obtaining a ThingWorx disconnected site license through our [License Management site](#) can be found in our [Licensing guide](#).

5. 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 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:

```
export CLASSPATH=
<location where zip file is extracted>\WEB-
INF\lib\thingworx-platform-common-8.1.0-bxx.jar;
<location where zip file is
extracted>\password_setup\WEB-INF\lib\slf4j-api-
1.7.21.jar;
<location where zip file is extracted>\WEB-
INF\lib\logback-core- 1.0.13.jar;
<location where zip file is extracted>\WEB-
INF\lib\logback-classic-1.0.13.jar;
<location where zip file is extracted>\WEB-INF\lib\thingworx-
common-8.1.0-bxx.jar
```

i. Add the variable to the CLASSPATH:

- ii. In your command shell, enter 'java -version'. It should respond with a Java version.

- d. 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.
- e. 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>
```

6. Restart Tomcat to start ThingWorx:

```
$ sudo systemctl restart tomcat
```

7. Log into ThingWorx Composer:

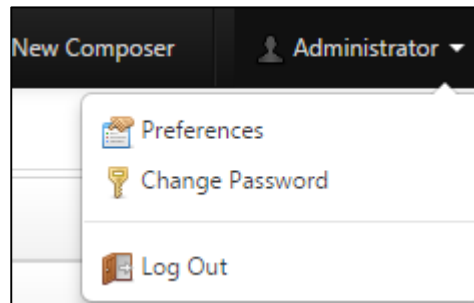
In a browser, open <https://localhost/Thingworx>.

The login information below is for the Administrator user only.

User: Administrator

Password: trUf6yuz2?_Gub

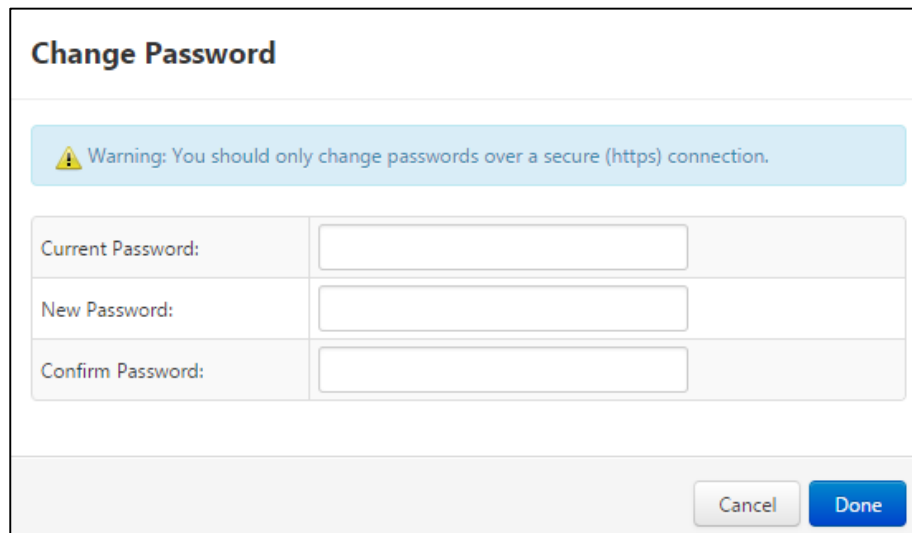
8. Change the default password.
 - a. In Composer, click **Administrator>Change Password**.



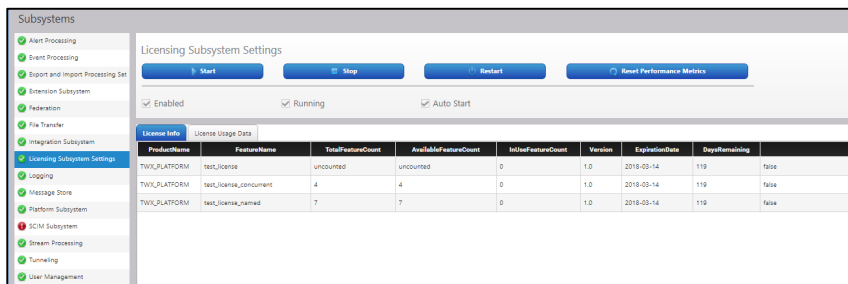
- b. In the **Change Password** window, enter **Current Password**, **New Password**, and **Confirm Password**.

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

- c. Click **Done**.

A screenshot of the 'Change Password' dialog box. The title bar says 'Change Password'. Below the title is a light blue warning banner with a yellow triangle icon and the text: 'Warning: You should only change passwords over a secure (https) connection.' Below the banner is a table with three rows: 'Current Password:', 'New Password:', and 'Confirm Password:'. Each row has a corresponding text input field. At the bottom right of the dialog are two buttons: 'Cancel' and 'Done'.

- To determine the status of the license, administrators can open the Licensing Subsystem Settings in ThingWorx 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 Installation and Configuration with Amazon RDS

Installing PostgreSQL RDS Instance

NOTE: Before performing this section, perform the steps for installing Apache Tomcat and Java for your setup:

- Windows: [Installing Oracle Java and Apache Tomcat](#)
- Ubuntu: [Installing Oracle Java and Apache Tomcat](#)
- RHEL: [Installing Oracle Java and Apache Tomcat](#)

- 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.
- On the **Specify DB Details** page, specify the following information:
 - In the **DB Engine Version** field, select the latest 9.4 version available. (9.4.9 in this example).
 - In the **DB Instance Class** field, select the appropriate class. NOTE: **m3.2xlarge** is recommended for production use.
 - In the **Multi-AZ Deployment** field, select **Yes** if you have an HA environment.
 - 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.
5. On the RDS Dashboard, click **Security Groups**.
6. Create a DB security group to control the DB access later.

Create Parameter Group

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

Parameter Group Family: postgres9.4

Group Name:

Description:

RDS Dashboard

Instances

Clusters

Reserved Purchases

Snapshots

Security Groups

Create DB Security Group

Filter: Search DB Security

	Name
<input type="checkbox"/>	default
<input type="checkbox"/>	pgbaselinesg

Create DB Security Group

Name:

Description:

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.

DB Security Group: default

Connection Type	Details
This DB Security Group has no authorizations. You will not be able to connect to DB Instances a Type below to add an a	

Connection Type: EC2 Security Group

This account: ☒ AWS Account ☐ Another Account

AWS Account ID: 029736143729

EC2 Security Group Name: twx-test-sg

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_2016-09-2
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

Creating a New User Role in PostgreSQL (RDS)

A PgAdmin III client is required to connect to the Amazon RDS PostgreSQL instance and configure it. Install PgAdmin per the client machine's operating system before creating a new user role in RDS:

- [Installing PgAdmin 3 \(Windows\)](#)
- [Installing PgAdmin 3 \(Ubuntu\)](#)
- [Installing PgAdmin 3 \(RHEL\)](#)

1. Create a new user role:

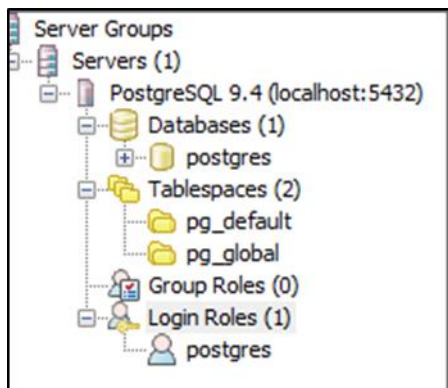
NOTE: The admin user created above while installing the RDS instance is *<admin user name>*.

- a. Right click **PostgreSQL9.4**.
- b. Select **NewObject>New Login Role**. On the **Properties** tab, in the **Role name** field, enter a user role name.
- a. On the **Definition** tab, in the **Password** field, enter a unique postgres 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.

- a. Click **OK**.
NOTE: Remember the user role name created in these steps for later use.



Configuring and Executing the Model/Data Provider Schema Script (RDS)

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

- [Configuring and Executing the Model/Data Provider Schema Script \(Windows\)](#)
- [Configuring and Executing the Model/Data Provider Schema Script \(Ubuntu\)](#)
- [Configuring and Executing the Model/Data Provider Schema Script \(RHEL\)](#)

Configuring platform-settings.json (RDS)

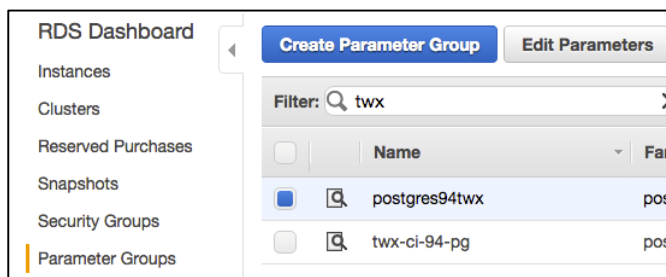
This is identical to the sections above except for the Amazon RDS PostgreSQL host name/ip.

- [Configuring platform-settings.json \(Windows\)](#)
- [Configuring platform-settings.json \(Ubuntu\)](#)
- [Configuring platform-settings.json \(RHEL\)](#)

Installing and Configuring PostgreSQL DB Host Servers (RDS)

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

1. Edit the parameter group created earlier.



2. Edit the parameters listed below to suit your environment.

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/current/static/runtime-config-resource.html#GUC-WORK-MEM

maintenance_work_mem	512MB	Optional performance tuning. Specifies the maximum amount of memory to be used by maintenance operations. Refer to http://www.postgresql.org/docs/current/static/runtime-config-resource.html#GUC-MAINTENANCE-WORK-MEM
effective_cache_size		Should be set to an estimate of how much memory is available for disk caching by the OS and within the database. It is recommended to set this to half the memory available on the machine.
checkpoint_segments		Depends on the size of the PostgreSQL box. Should set to 32/64/128/256, depending upon machine size.
checkpoint_completion_target		If the checkpoint_segments is changed from the default value of 3, change this to 0.9.
ssl_renegotiation_limit		If PostgreSQL is deployed 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.

Installing ThingWorx (RDS)

Thingworx server installation is identical to the previous section since the only difference is using RDS instead of installing PostgreSQL database on your own. Refer to the relevant OS section to install ThingWorx:

- [Windows](#)
- [Ubuntu](#)
- [RHEL](#)

Appendix A: Tomcat Java Option Settings

Mandatory Settings

Setting	Description
-server	Explicitly tells the JVM to run in server mode. This is true by default when using 64-bit JDK, but it is best practice to declare it.
-d64	Explicitly tells the JVM to run in 64-bit mode. The current JVM automatically detects this, but it is best practice to declare it.
XX:+UseG1GC	Tells the JVM to use the Garbage First Garbage Collector.
-Dfile.encoding=UTF-8	Tells the JVM to use UTF-8 as the default character set so that non-Western alphabets are displayed correctly.
-Djava.library.path	Specifies the path to the native library .
-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.
-Dcom.sun.management.jmxremote.authenticate=false	No authentication required.
-Djava.rmi.server.hostname=<host or IP>	The hostname or IP that the underlying RMI client connection will use.

Appendix B: Sample platform-settings.json

The platform-settings.json file is available in the software download.

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

```
{
  "PlatformSettingsConfig": {
    "BasicSettings": {
      "BackupStorage": "/ThingworxBackupStorage",
      "DatabaseLogRetentionPolicy": 7,
      "EnableBackup": true,
      "EnableHA": false,
      "EnableSystemLogging": false,
      "EnableSSO": false,
      "HTTPRequestHeaderMaxLength": 2000,
      "HTTPRequestParameterMaxLength": 2000,
      "InternalAesCryptographicKeyLength": 128,
      "Storage": "/ThingworxStorage"
    },
    "HASettings": {
      "CoordinatorConnectionTimeout": 15000,
      "CoordinatorHosts": "127.0.0.1:2181",
      "CoordinatorMaxRetries": 3,
      "CoordinatorRetryTimeout": 1000,
      "CoordinatorSessionTimeout": 60000,
      "LoadBalancerBase64EncodedCredentials":
"QWRtaW5pc3RyYXRvcjphZG1pbG=="
    }
  },
  "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,
        "jdbcUrl":
"jdbc:sqlserver://localhost:1433;databaseName=thingworx;applicationName=Thing
worx;",
        "maxConnectionAge": 0,
        "maxIdleTime": 0,
        "maxIdleTimeExcessConnections": 300,
        "maxPoolSize": 100,
        "maxStatements": 100,

```

```

        "minPoolSize": 5,
        "numHelperThreads": 8,
        "password": "Password@123",
        "testConnectionOnCheckout": false,
        "unreturnedConnectionTimeout": 0,
        "username": "msadmin"
    },
    "StreamProcessorSettings": {
        "maximumBlockSize": 2500,
        "maximumQueueSize": 250000,
        "maximumWaitTime": 10000,
        "numberOfProcessingThreads": 5,
        "scanRate": 5,
        "sizeThreshold": 1000
    },
    "ValueStreamProcessorSettings": {
        "maximumBlockSize": 2500,
        "maximumWaitTime": 10000,
        "maximumQueueSize": 500000,
        "numberOfProcessingThreads": 5,
        "scanRate": 5,
        "sizeThreshold": 1000
    },
    "PersistentPropertyProcessorSettings": {
        "maximumBlockSize": 2500,
        "maximumWaitTime": 1000,
        "maximumQueueSize": 100000,
        "numberOfProcessingThreads": 20,
        "scanRate": 25,
        "sizeThreshold": 1000
    }
}
}
}

```

Appendix C: platform-settings.json Options

NOTE: The platform-settings.json file is available for administrators to adjust settings for fine-tuning.

platform-settings.json Options		
Setting	Default	Description
Basic Settings		
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.

platform-settings.json Options		
Setting	Default	Description
EnableSystemLogging	false	Determines whether system logging is enabled. NOTE: DO NOT TURN THIS ON UNLESS INSTRUCTED BY THINGWORX SUPPORT.
EnableSSO	false	Set to true to enable SSO for ThingWorx Platform. When SSO is enabled, all authentication is redirected to the central authorization server that is configured in the sso-settings.json file. Edge websocket authentication is not affected
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 are optional, and are ignored if the EnableHA setting above is set to false .		
CoordinatorConnectionTimeout	15000	How long to wait (in milliseconds) for a connection to be established with process/server used to coordinate ThingWorx leadership.
CoordinatorHosts	127.0.0.1:2181	A comma-delimited list of server IP addresses on which the processes used to coordinate ThingWorx leadership exist (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 process/server used to coordinate ThingWorx leadership.

platform-settings.json Options		
Setting	Default	Description
CoordinatorRetryTimeout	1000	How long to wait (in milliseconds) for each retry attempt.
CoordinatorSessionTimeout	60000	How long ThingWorx waits (in milliseconds) without receiving a "heartbeat" from the process/server used to coordinate ThingWorx leadership.
LoadBalancerBase64EncodedCredentials	QWRtaW5pc3RyYXRvcjphZG1pbG==	<p>The Base64-encoded credentials for the HA Load Balancer, in the format of <code><user>:<unique password></code>.</p> <p>NOTE: You can use any utility that Base64 encodes the matching <code><user>:<unique password></code> string used in your load balancer setup.</p>
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 Neo is not the Persistence Provider, then this entire section should be ignored.		
StreamProcessorSettings		
maximumBlockSize	2500	The maximum number of stream writes to process in one block.
maximumQueueSize	250000	The maximum number of stream entries to queue (will be rejected after that).
maximumWaitTime	10000	The maximum wait time (in milliseconds) before flushing stream buffer.
scanRate	5	The rate (in milliseconds) at which to check the buffer status.
sizeThreshold	1000	The maximum number of items to accumulate before flushing stream buffer.
ValueStreamProcessorSettings		
maximumBlockSize	2500	The maximum number of stream writes to process in one block.

platform-settings.json Options		
Setting	Default	Description
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).
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.
H2PersistenceProviderPackage		
Contains H2 Database Engine-specific Persistence Provider settings. If H2 is not the Persistence Provider, then this entire section should be ignored.		
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.

platform-settings.json Options		
Setting	Default	Description
idleConnectionTestPeriod	6	Time period (in seconds) where connections will be tested so that 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. NOTE: If you are experiencing "No connection to model provider" errors, review this setting. Compare to firewall defaults. Lowering the default will alleviate disconnection issues.
initialPoolSize	10	Initial number of database connections created and maintained within a pool upon startup. Should be between minPoolSize and maxPoolSize.
maxConnectionAge	0	Seconds, effectively a time to live. A Connection older than maxConnectionAge will be destroyed and purged from the pool.
maxIdleTime	0	Seconds a connection can remain pooled but unused before being discarded. Zero means idle connections never expire.

platform-settings.json Options		
Setting	Default	Description
maxIdleTimeExcessConnections	36000	The number of seconds that connections in excess of minPoolSize are permitted to remain in idle in the pool before being culled. Intended for applications that wish to aggressively minimize the number of open connections, shrinking the pool back towards 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 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.
testConnectionOnCheckout	false	If true, an operation will be performed at every connection checkout to verify that the connection is valid.

platform-settings.json Options		
Setting	Default	Description
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.
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.

platform-settings.json Options		
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 PostgreSQL-specific persistence provider settings. If PostgreSQL is not the persistence provider, then this entire section should be ignored.		
Connection Information		
acquireIncrement	5	Determines how many connections at a time the platform will try to acquire when the pool is exhausted.
acquireRetryAttempts	3	Defines how many times ThingWorx will try to acquire a new Connection from the database before giving up.
acquireRetryDelay	10000	The time (in milliseconds) ThingWorx will wait between acquire attempts.
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.

platform-settings.json Options		
Setting	Default	Description
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"	<p>The jdbc url used to connect to PostgreSQL.</p> <p>NOTE: If the default schema name is changed (from public), you must add <databasename>?currentSchema=<name of schema></p> <p>For example, if the schema name is mySchema, it would be:</p> <p>jdbc:postgresql://<DBServer>:<DBPort>/<databasename>?currentSchema=mySchema</p> <p>NOTE: If you are configuring an HA solution, this should reflect the server IP that the pgPool process is running on. Change the port to the port that pgPool is serving.</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.

platform-settings.json Options		
Setting	Default	Description
maxIdleTimeExcessConnections	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, 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	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 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.

platform-settings.json Options		
Setting	Default	Description
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 (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).

platform-settings.json Options		
Setting	Default	Description
scanRate	5	The rate (in milliseconds) before flushing the stream buffer.
sizeThreshold	1000	Maximum number of items to accumulate before flushing the value stream buffer.
PersistentPropertyProcessorSettings		
maximumBlockSize	2500	The maximum number of property writes to process in one block.
maximumWaitTime	1000	The maximum wait time (in milliseconds) before flushing the property buffer.
maximumQueueSize	100000	The maximum number of property entries to queue (will be rejected after that).
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		
Contains MS SQL Server-specific Persistence Provider settings. If MS SQL Server is not the Persistence Provider, then this entire section should be ignored.		
Connection Information		
acquireIncrement	5	Determines how many connections at a time the ThingWorx will try to acquire when the pool is exhausted.
acquireRetryAttempts	3	Defines how many times ThingWorx will try to acquire a new Connection from the database before giving up.
acquireRetryDelay	10000	The time (in milliseconds) ThingWorx will wait between acquire attempts.
checkoutTimeout	1000000	The number of milliseconds a client calling getConnection() will wait for a Connection to be checked-in or acquired when the pool is exhausted.
driverClass	"com.microsoft.sqlserver.jdbc.SQLServerDriver"	The fully-qualified class name of the JDBC driverClass that is expected to provide Connections.
fetchSize	5000	The count of rows to be fetched in batches instead of caching all rows on the client side.

platform-settings.json Options		
Setting	Default	Description
idleConnectionTestPeriod	60	Time period (in seconds) where connections will be tested so that 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. NOTE: If you are experiencing "No connection to model provider" errors, review this setting. Compare to firewall defaults. Lowering the default will alleviate disconnection issues.
initialPoolSize	5	Initial number of database connections created and maintained within a pool upon startup. Should be between minPoolSize and maxPoolSize.
jdbcUrl	"jdbc:sqlserver://localhost:1433;databaseName=thingworx;applicationName=Thingworx;"	The jdbcUrl 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.

platform-settings.json Options		
Setting	Default	Description
maxIdleTimeExcessConnections	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, 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	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	Password@123	The password 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.

platform-settings.json Options		
Setting	Default	Description
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	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 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).

platform-settings.json Options		
Setting	Default	Description
scanRate	5	The rate (in milliseconds) before flushing the stream buffer.
sizeThreshold	1000	Maximum number of items to accumulate before flushing the value stream buffer.
PersistentPropertyProcessorSettings		
maximumBlockSize	2500	The maximum number of property writes to process in one block.
maximumWaitTime	1000	The maximum wait time (in milliseconds) before flushing the property buffer.
maximumQueueSize	100000	The maximum number of property entries to queue (will be rejected after that).
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.

Appendix D: Metrics Reporting

By default, your ThingWorx Core 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. For more information, see the Platform Subsystem topic in the [Help Center](#).

Appendix E: Installing PostgreSQL Client Package and PostgreSQL User

In order to issue PostgreSQL commands from the client machine to the PostgreSQL server, do so from a PostgreSQL user. To do so, the postgresql-client-9.4 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** that are discussed later in the [monitoring/administration](#) section

Windows

PostgreSQL client comes along with PgAdmin III. Follow the same instructions for installation.

Ubuntu

```
sudo apt-get install postgresql-client
```

RHEL

```
sudo yum install postgresql94.x86_64
```

Appendix F: Licensing Troubleshooting

You must have a license file for ThingWorx 8.0 and later. Some possible situations that may require troubleshooting are described below:

Issue	Possible Resolution
Problem deploying thingworx.war .	Verify that the ThingworxStorage/extensions/web-inf folder contains the licensing libraries (DLL files).
<p>The following error is received when deploying ThingWorx:</p> <pre>org.apache.catalina.core.Applicati onContext.log HTMLManager: FAIL - Deploy Upload Failed, Exception: org.apache.tomcat.util.http.fileup load.FileUploadBase\$SizeLimitExcee dedException: the request was rejected because its size (90883556) exceeds the configured maximum (52437800) java.lang.IllegalStateException: org.apache.tomcat.util.http.fileup load.FileUploadBase\$SizeLimitExcee dedException: the request was rejected because its size (90883556) exceeds the configured maximum (52437800) at org.apache.catalina.connector.Requ est.parseParts(Request.java:2871</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.
<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.</p> <p>If you need further assistance with your licenses, please contact the License Management team.</p>

Issue	Possible Resolution
<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 -Djava.library.path from Tomat's Java configuration before undeployment.</p>
<p>An error message similar to the following is seen in the <i>ConfigurationLog.log</i>:</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>

Issue	Possible Resolution
<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] ===== = 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>	<p>The license file may have been opened/edited/saved in a browser. Download the license file again, rename it to license_capability_response.bin, and place in ThingworxPlatform folder without editing or saving it.</p>