PTC®

PTC® IoT Academic Program

Administration Guide

How to Administrate a PTC IoT Academic Program Hosted Instance of ThingWorx

Revision #	Date	ThingWorx Revision	Changes	Owner
1.0	**13.01.2015**	**5.2.0.437**		**Adrian Petrescu**
2.0	**10.02.2015**	**5.2.0.437**		**Adrian Petrescu**
3.0	**18.02.2015**	**5.2.0.437**		**Adrian Petrescu**
4.0	**27.02.2015**	**5.2.0.437**		**Adrian Petrescu**
5.0 – current	**07.04.2015**	**5.2.0.437**		**Adrian Petrescu**

Table of Contents

1. INTRODUCTION

1.1 Purpose

The purpose of this document is to help educational institutions manage their hosted instance of the ThingWorx Application development platform.

1.2 Scope

This Administration Guide helps those in an educational institution who have administrative privileges perform basic functions such as monitor performance, troubleshoot, manage users and ensure security. The goal is to enable all users to receive the most benefit from the ThingWorx Application development platform.

For more details on the purpose and functions of the ThingWorx Application development platform, visit the ThingWorx Website (http://ThingWorx.com). A glossary and acronyms list is included in this Administration Guide.

2. DESCRIPTION

2.1 About ThingWorx

ThingWorx is the premiere software platform designed to build and run innovative applications for the connected world. ThingWorx reduces the time, cost, and risk required to build innovative Internet of Things (IoT) applications.

The platform provides a complete application design, runtime, and intelligence environment.

ThingWorx is a rapid, model-based application development platform. By employing modeling instead of coding, the content developer is able to focus on agility and application composition, rather than debugging, maintaining, and updating code. The model artifacts become a set of reusable building blocks to assemble new applications

- See more at: ThingWorx.com/learn-more and the ThingWorx Marketplace at http://marketplace.thingworx.com

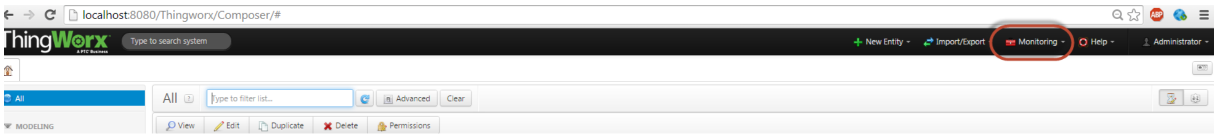
3. OPERATIONS

3.1 Monitoring Capabilities

The monitoring capabilities within ThingWorx become a valuable tool to help you keep your hosted instance in good working order and performing well. It is also a valuable tool to help you trouble shoot.

With the information displayed in the Monitoring option you can monitor the main activities from the ThingWorx platform (for example, you can use the LOGS section to see and solve any type of message error that may occur or the STATUS section to see when you have a device connected to your instance, what users are logged in at the time, etc).

You can access the Monitoring capability from the menu as is shown in the picture below:



The following administrative screens can be accessed from the Monitoring menu:

LOGS

- Application
- Communication
- Composer
- Configuration
- Security
- Script

ACTIVE EDITS

- All Users
- Mine

STATUS

- Connection Servers
- Remote Things
- Subsystems
- System
- Users Logged In

ALERTS

- Alert Summary
- Alert History

LOGS

The logs record information in regards to what is running and what is happening, and is particularly helpful in identifying errors. A Recording Level can be set for each log in

Configuration/Settings. Depending on the Log Level, you will receive more granularity of what is going on in the system.

Each of these logs can be filtered using the **Date/Time** filter, the **User** filter, the **Levels** range, and can be limited to a number of rows. The content that is logged to a particular log is governed by the settings configured in the Settings Editor.

ACTIVE EDITS

Active Edits is a ThingWorx functionality that allows a user to see what he or she is working on, or to see what everybody else is currently working on. Using Active Edits you can see what is open in ThingWorx. For example, if you have three tabs open with a thing, a datashape and a thing template, you will be able to see all this information by accessing the **Mine** option from **ACTIVE EDITS.** Also, you are able to see what everyone is working on by accessing the **All Users** option.

View all active edits in the Composer

- **All Users**: Displays active edits and details by all users. Includes edit name and type, user name, and time of edit.
- **Mine**: Displays your active edit details

STATUS

Status lets users know the status of the server, remote things, subsystems, system and users. Using the status function, you can see the status of any thirrd party device that you want to integrate with ThingWorx or what devices you have integrated with ThingWorx. Also, you can see which users are logged into the ThingWorx application at a certain time.

View the status of the following:

- **Connection Servers**
- **Remote Things**
- **Subsystems**
- **System**
- **Users Logged In**

ALERTS

Alert History and Alert Summary streams provide the functionality to monitor alerts in the system.

The following Alert Monitoring streams can be accessed:

- **Alerts Summary**: Compiles data from the last reset of the server to the current state. Within the Alert Summary page, alerts can be viewed, acknowledged, and sorted (by acknowledged or unacknowledged).
- **Alert History**: A comprehensive log that records all information recorded into the alert stream, and the data is stored until manually removed.

3.2 Import/Export Capabilities

ThingWorx provides import and export functionality for all ThingWorx entities as well as the data contained within certain entities, including streams, tables, blogs, and wikis. This can be helpful in transferring objects and applications across ThingWorx servers. For example, Import and Export might be used to share a mashup with another university or could be used to import a plug in from PTC MarketPlace.

Import/Export can be used for either the Server or Client side. Import/Export distinguishes between Entities and Data, where Entities are anything you have modeled and mashed up, and data is the content of the following entities:

- DataTables
- Streams
- Value Streams
- Blogs
- Wikis

You can export anything, from a single entity in your ThingWorx model, to your entire model. There is filtering available based on Entity Type, Modified Date (if Entities have been previously modified and/or if there is a Data time stamp) and Tags.

These are the permissions that the two default ThingWorx groups (**Developers** and **Administrators**) have:

Administrator		**Developer**	
Action	**File Types**	**Action**	**File Types**
Import	XML, JSON, ZIP	Import	XML[1]
Export	XML, JSON	Export	XML[2], JSON[3]
Monitoring	Full Access	Monitoring	Developers can view the information in the Remote Things part under the Monitoring feature and can also view and read log files

Legend:

1 – Can be done by using the Postman extension from the Google Chrome browser; You can add this extensions by go into Settings in Google Chrome. From there, select the Extensions Tab, press the Get more extensions button, search for Postman and install the Postman – REST Client

2 - Can be done by typing the following URL format in the Google Chrome browser: http://hostname/ThingWorx/Exporter/Things/thethingname

3 - Can be done by typing the following URL format in the Google Chrome browser: http://hostname/ThingWorx/DataExporter/thetdataname

XML – Entities files
JSON – data files
ZIP – Extensions files

From the Import/Export menu, the following Administrative functions are available, and descriptions of each are located below:

Manually Importing and Exporting

In Composer, only administrator users can view the Import/Export menu. This menu enables users to import and export data, entities, and extensions. However, by using the API and setting Design Time permissions, non-administrative users can manually import data in and export data from ThingWorx.

The following Design Time permissions apply when manually importing and exporting:

- To export, users must have Design Time Read permission
- To import, users must have Design Time Create permission

A. Exporting

From a browser, use the calls in the table below to export data and entities.

- Entity data is exported as an xml file
- Blog, wiki, stream, and value stream data is exported as a json file

Description	URL Example
Exporting Entities	
Exporting entities	*/Thingworx/Exporter/Things*
Exporting a single entity	*/Thingworx/Exporter/Things/MyThingName*
Exporting by tag	*/Thingworx/Exporter/searchTags=Vocab1:Term1;Vocab2:Term2*
Exporting Data	
Exporting all blog data	*/Thingworx/DataExporter/Blogs*
Exporting a single blog's data	*/Thingworx/DataExporter/Blogs/BlogName*
Exporting all wiki data	*/Thingworx/DataExporter/Wikis*
Exporting a single wiki's data	*/Thingworx/DataExporter/Wikis/WikiName*
Exporting all stream data	*/Thingworx/DataExporter/Streams*
Exporting a single stream's data	*/Thingworx/DataExporter/Streams/StreamName*
Exporting all value stream data	*/Thingworx/DataExporter/ValueStreams*
Exporting a single value stream's data	*/Thingworx/DataExporter/ValueStreams/ValueStreamName*
Exporting by tag	*/Thingworx/DataExporter/searchTags=Vocab1:Term1;Vocab2:Term2*
Exporting data within a date range	*/Thingworx/Exporter/?startDate=2014-11-17T14:57:28&endDate=2014-11-24T14:57:33&Accept=application/json*

B. Importing

Importing is a multi-part request. It is recommended that you build a simple HTML form such as the example below:

```
<html>
<body>
<FORM action="/Thingworx/Importer?purpose=import"
      enctype="multipart/form-data"
      method="post">
  <P>
  <BR>
  What files are you sending? <INPUT type="file" name="files"><BR>
  <INPUT type="submit" value="Send"> <INPUT type="reset">
 </FORM>
 </body>
 </html>
```

IMPORT

- From File
- From ThingWorxStorage

EXPORT

- To File
- To ThingWorxStorage
- Source Control Entities

EXTENSIONS

- Import
- Manage

Import from File

Imports entities or data from a selected (local) file (not asynchronous).

Import from ThingWorxStorage

Imports entities and data (optional) asynchronously from a selected (server-side) file located in the ThingWorxStorage/exports folder on the server. If data is included in the import, Include Data imports the Data in addition to the Entities. The files used for import must be located in the ThingWorxStorage/exports folder. ThingWorx automatically pairs the Entity file with the Data file.

The information contained within Streams, ValueStreams, DataTables, Blogs, and Wikis (row entries, blog posts, wiki pages) is considered Data.

Export to File

Exports a single entity, multiple entities, or data to a selected (local) destination (not asynchronous).

Export entities from specified collections (blogs, datatables, streams, value streams, or wikis), with model tags, or by date (determined by the “last modified date” time stamp on the file).

Export data from specified collections (blogs, datatables, streams, value streams, or wikis), with model tags, or by date (determined by the data’s recorded “timestamp”).

Export to ThingWorxStorage

Exports entities and data (optional) asynchronously to the server (in the ThingWorxStorage/exports folder).

ThingWorx provides a single Date Range filter that uses the Entity's Data Last Modified so you can narrow down what to export. For data, it is tied to the Entity's Last Modified Data, (unlike in Export to File, where the Data Export Date Filter ties to the DateTime field of the data entries).

Source Control Entities

The Export Source Control Entities function exports ThingWorx entities by collection, model tags, start and end dates. The specified entities can be directed to a repository or path and organized into a zip folder. When entities are exported (as XML files), a file structure is created at the destination that resembles the way a source control system manages its artifacts.

The following options are available when specifying the export of the entities:

- **Collections**: Select Application Keys, DataShapes, DataTags, DirectoryServices, ExtensionPackages, Groups, Localization Tables, Subsystems, DirectoryServices, Logs, Mashups, MediaEntities, Menus, Logs, ModelTags, Networks, Organizations, Resources, ScriptFunctionLibraries, StateDefinitions, SytleDefinitions, ThingPackages, Things, ThingShapes, ThingTemplates, Users.
- **Model Tags**: Use the magic picker to export by model tags.
- **Start Date**: Use the start date to limit the content of exported entities based on the timestamp recorded with your entities. Start date exports entities timestamped after the start date.
- **End Date**: Use the end date to limit the content of exported entities based on the timestamp as recorded with your data entries. End date exports entities timestamped before the specified end date.
- **Repository**: Specify a repository for the export.
- **Path**: Specify the location of the export.
- **Zip File:** Places the exported entities into a zip folder in the specified location. A file name must be provided.

Import Extensions

Import a single extensions or import multiple extensions by packaging into a .zip folder.

Manage Extensions

Displays all installed extensions and their components. Extensions can be deleted from this page.

3.3 Managing Storage and Bandwidth Limitations

To manage the storage and bandwidth limitations in a ThingWorx environment, an extension was created. The name of that extension is **StorageExtensionApp.** You have everything you need to import the extension below:

dataexport.json Entities.xml StorageExtensionApp.zip

In the next steps you learn how to use this extension.

3.3.1 Managing Storage and Bandwidth Limitations – How to Use the StorageExtensionApp Extension

1. First you need to import the StorageExtensionApp. To do this, to the upper-right corner of the composer under the Import/Export menu and click **Import**. Select your extension and import it.

2. Now that you have added the extension, you need to create a new Thing. To add a Thing, type **+Thing** in the spotlight search and press ENTER. This opens a new Thing entity. Name your Thing.

3. In the Thing Template field, click the wand icon and search for the StorageExtensionApp template.

4. Select **Configuration** tab under **Entity Information**. Set the path of the folder that you want to monitor. You also have to set the size in GB for this folder.

5. In the folderTarget field specify the targeted folder (for example /ThingWorxStorage).

6. In the limit field specify the size limit value in GB.

7. Save your progress.

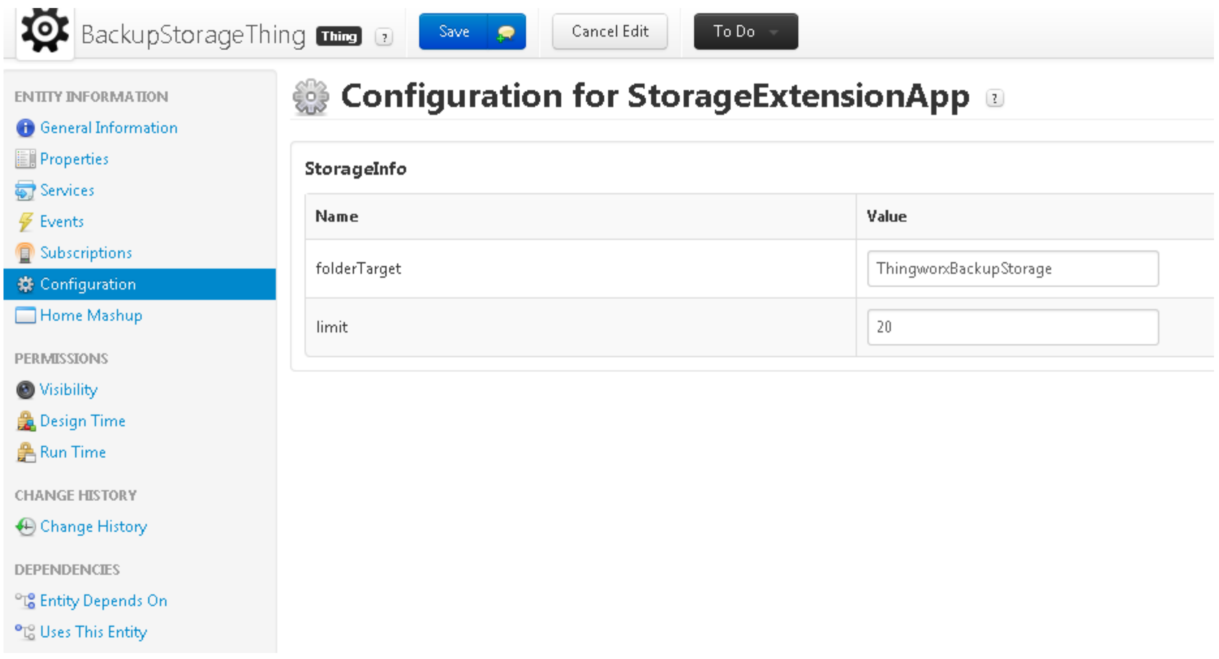
8. To test this functionality, browse to the Services part under Entity Information. Click the **Test** button of the FolderInformations service. This opens a new tab. Click the **Execute Service** button. Now you see three display fields indicating the folder size in GB, the folder size in MB and the usage percentage of the folder. If you receive the following error message: Unable to Invoke Service FolderInfomations on “your ThingName”: null the path to your targetfolder is not properly configured, or the location specified does not exist.

3.3.2 Managing Storage and Bandwidth Limitations – Storage Checker Mashup

1. Import the StorageExtensionApp Extension in ThingWorx.
2. Import the Entities xml file.

3. Import the dataexport Json file.

4. By importing the entities and the data, you now have three Things: StorageThing, BackupStorageThing and StorageSumThing. The first two Things are based on the StorageExtensionApp extension. The third one uses the GenericThing template. To point to the ThingWorxStorage and ThingWorxBackupStorage folders, must navigate to the Configuration tab of each of the two Things and set the folderTarget properly (such as: C:/ThingWorxStorage or ThingWorxBackupStorage), as shown in the picture below and also set the storage limit in GB. The third Thing (StorageSumThing) contains two services: CurrentUsageInGb, which provides an average of the used space of both folders and displays them as a percentage and the second service TotalStorageSum which provides the sum of the used space from the Storage and Backup folders.

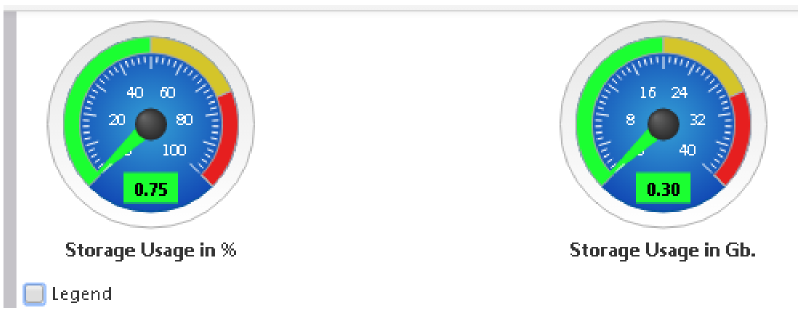


Configuration for StorageExtensionApp

StorageInfo

Name	Value
folderTarget	ThingworxBackupStorage
limit	20

5. There is now a new mashup named StorageMashup available. The mashup displays as shown:



6. On the left side of the mashup there is a gauge which displays the percentage of Total Storage Usage for both folders. On the right side there is a gauge which displays the usage in GB. Both Things have a pre-set limit of 20 GB, so if you add this, you will have a total of 40 GB.

3.3.3 Managing Storage and Bandwidth Limitations – Configuring the ThingWorx StorageNotifier

The StorageNotifier is a feature of ThingWorx which automatically sends an e-mail to the administrator after noticing that the Storage Usage of the Hosted Instance has reached 80%.

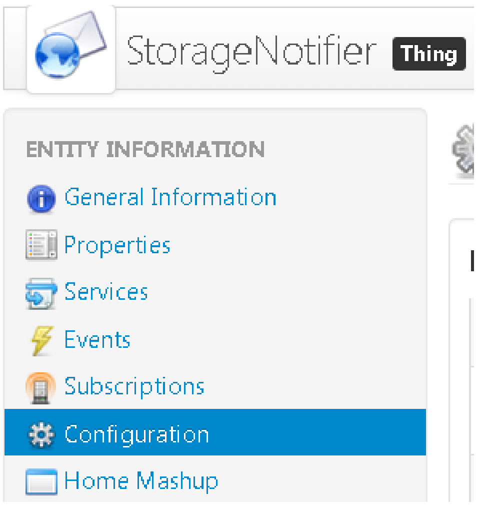
If the usage reaches 90%, the user receives a similar e-mail alert from ThingWorx.

The Notifier is configured to check the storage every hour to alert the administrator by sending an e-mail if a concern is identified.

After importing the entities and data from the previous step, you also have a Thing named StorageNotifier and one named StorageTimer available in ThingWorx.

The StorageNotifier uses the MailServer as Thing Template. As a result, this Thing can be configured properly to send or receive e-mails.

To configure the Thing, navigate to the Configuration tab of the Thing.



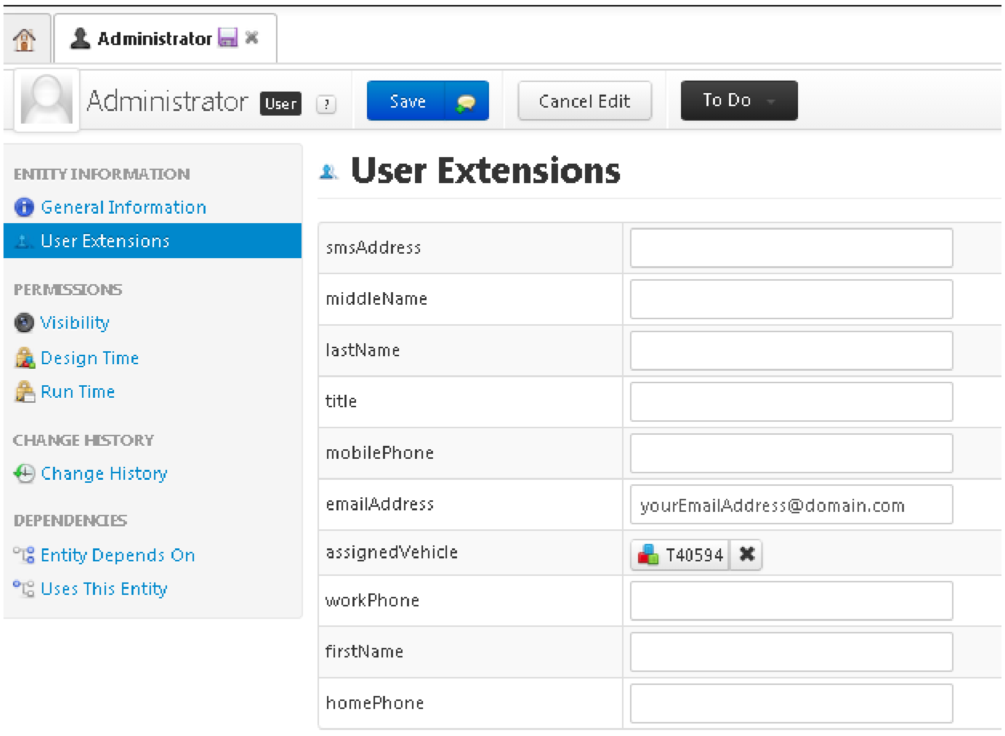
This opens the Configuration tab for the MailThing. Populate the fields with relevant information for your e-mail address. Typically you can leave the Use TLS and Use SSL check boxes unselected. In the SMTP and POP3 Server and Ports fields, specify the information provided by your e-mail provider.

In the Mail Account User field type in your e-mail address. To specify your e-mail password, click the change password button and type in the password. The other fields can be left unmodified.

Save your progress.

Now you have successfully configured your mail server. To receive e-mail alerts for the storage occupation, you also need to provide ThingWorx with the e-mail address to which you wish to have the notification sent.

Only the administrator receives the e-mail notification, so you need to configure the administrator user extensions properly. To do this, select the Users tab on the left side of the Composer. Click the Administrator User and proceed to the User Extensions tab. Populate the emailAddress category with your e-mail address, as shown:



User Extensions

smsAddress	
middleName	
lastName	
title	
mobilePhone	
emailAddress	yourEmailAddress@domain.com
assignedVehicle	T40594
workPhone	
firstName	
homePhone	

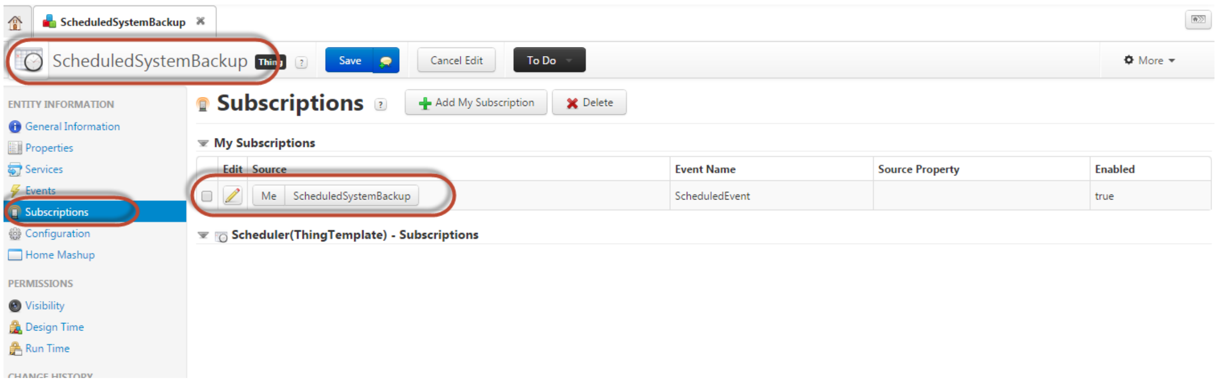
Save your progress.

Now every time the storage usage goes above 80%, you will be notified via e-mail.

4. BACKUP/RESTORE

4.1 Backup/Restore the Content

In ThingWorx, by default there is a Scheduler Thing called ScheduledSystemBackup (in the **Things** section) which you can use to run scheduled automatic backups of your ThingWorx development content. You can review how it is used by navigating to the Subscription tab as shown:



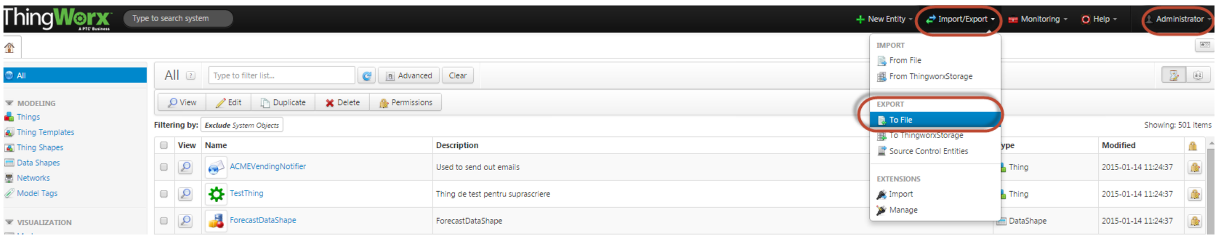
Edit	Source	Event Name	Source Property	Enabled
	Me ScheduledSystemBackup	ScheduledEvent		true

FULL - Completes a full back up each time

INCREMENTAL - Completes a full back up the first time and only the changes/updates thereafter.

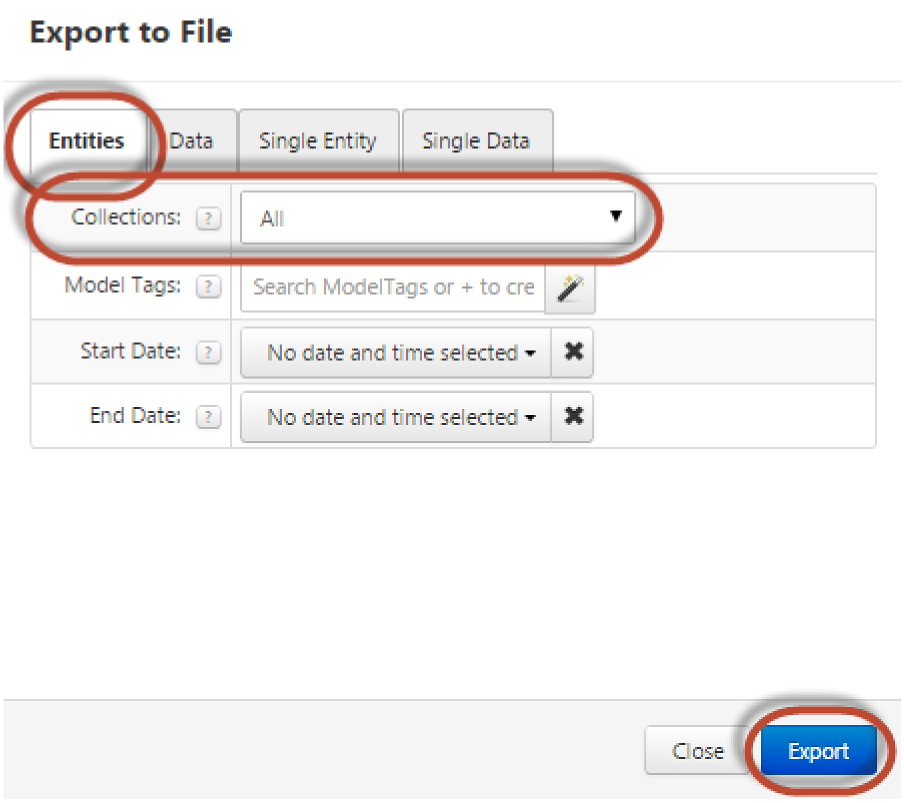
A best practice for backing up your content from a hosted ThingWorx instance is to export all of your entities and data from that instance (a monthly backup should be sufficient).

You can achieve this with an administrator user. Log into the ThingWorx platform with administrator user credentials and select the Export to file option from the **Import/Export** feature as shown:



View	Name	Description	Type	Modified
	ACMEVendingNotifier	Used to send out emails	Thing	2015-01-14 11:24:37
	TestThing	Thing de test pentru suprascriere	Thing	2015-01-14 11:24:37
	ForecastDataShape	ForecastDataShape	DataShape	2015-01-14 11:24:37

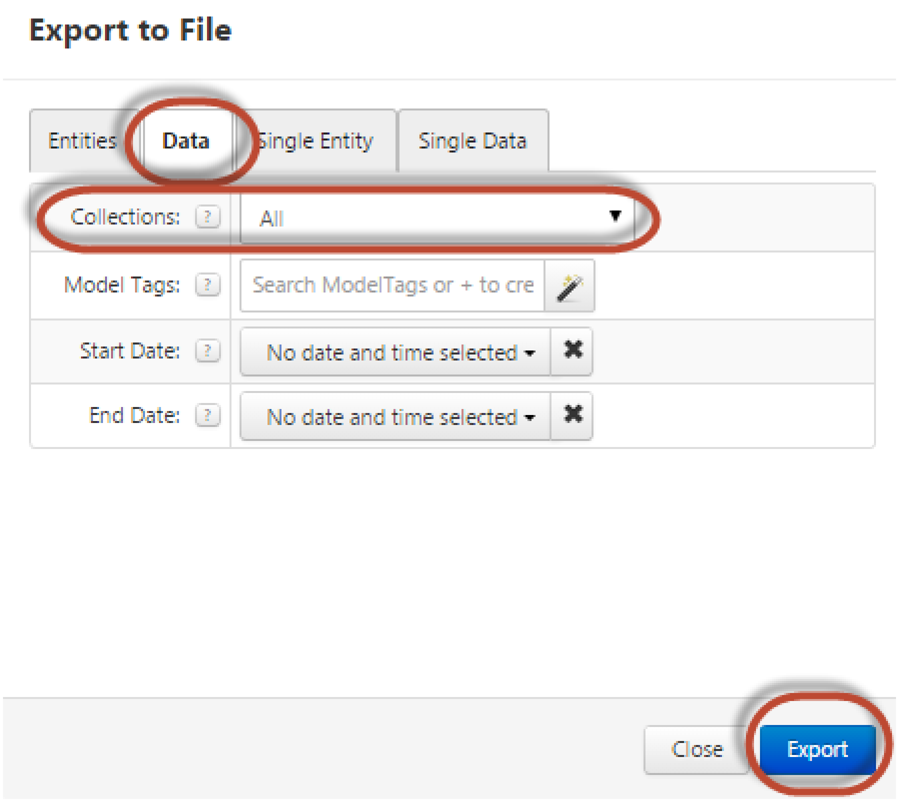
This opens the **Export to file** window. Then, proceed to the Entities tab and select **All** from the Collections drop-down menu (if that is not the default selection) and click the **Export** button as shown:


Export to File

Collections:	All
Model Tags:	Search ModelTags or + to cre
Start Date:	No date and time selected
End Date:	No date and time selected

After completing this step, an XML file will automatically be downloaded.

Next, select the **Data** tab, verify that **All** has been selected from the Collections drop-down menu and click the **Export** button as shown:


Export to File

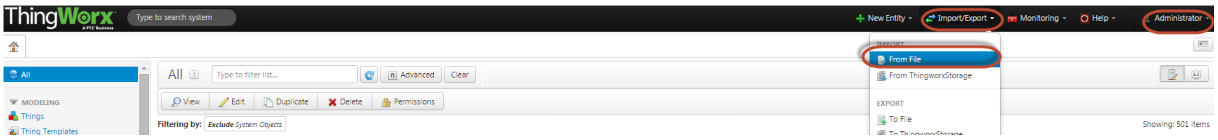
Collections:	All
Model Tags:	Search ModelTags or + to cre
Start Date:	No date and time selected
End Date:	No date and time selected

After completing this, a JSON file is automatically downloaded.

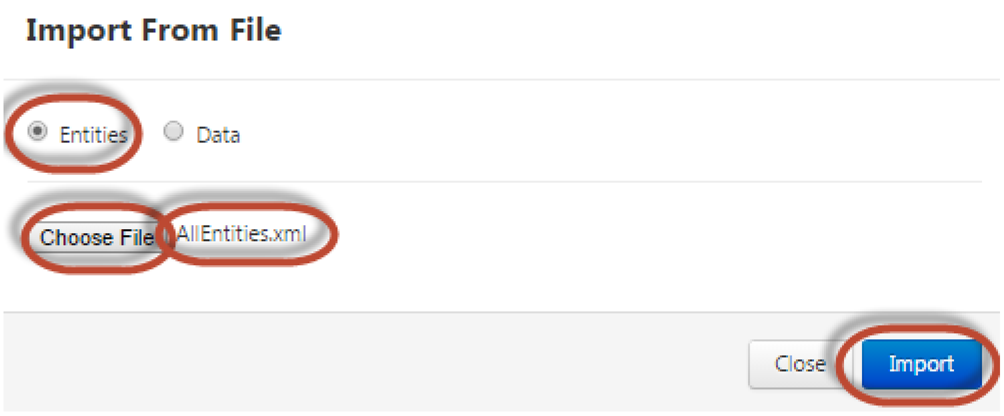
The two (XML and JSON) files store the ThingWorx hosted instance content.

You can now use these two files to restore the content on a different instance. For example, consider that you were provided with a newly hosted instance of ThingWorx and you wanted to restore on this instance all of the content from the previous hosted instance. All you would need to do is simply use the Import/Export feature to import the two XML and JSON files to the newly provided hosted instance. However, you must ensure that all extensions have already been installed on the new server. Once that is done, the steps are as follows:

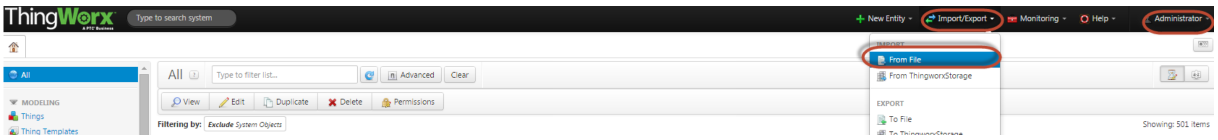
1. **Import the XML file -** Log in to the ThingWorx platform with an administrator user. Select **Import from file** from the **Import/Export** drop-down menu as shown:



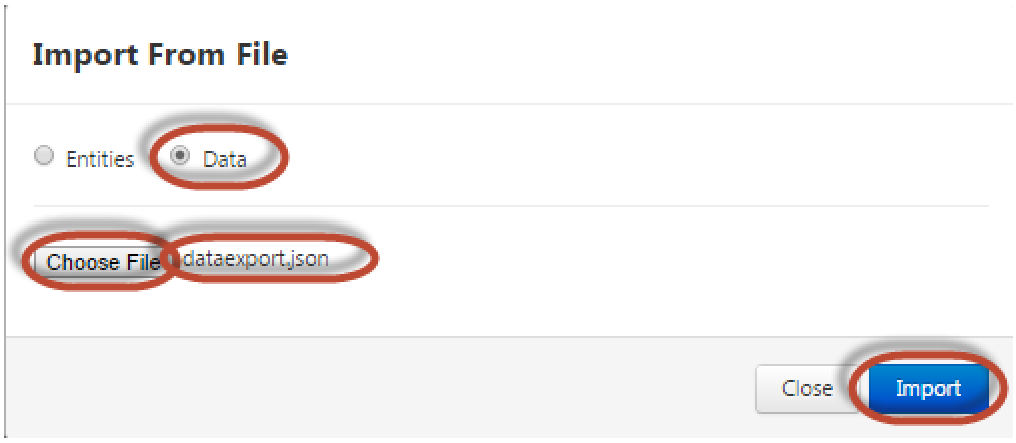
This opens the **Import from file** window. Here, you can select the **Entities** check box and then click the **Choose File** button to browse for the XML file that you want to import. Then, click the **Import** button as shown:



2. **Import the JSON file -** Log in to the ThingWorx platform with an administrator user. Select **Import from file** from the **Import/Export** drop-down menu, as shown:



This opens the **Import from file** window. Here, you can select the **Data** check box and then click the **Choose File** button to browse for the JSON file that you want to import. Then, simply click the **Import** button as shown:


Import From File

○ Entities [✓ ● Data]

Choose File dataexport.json

5 TROUBLESHOOTING

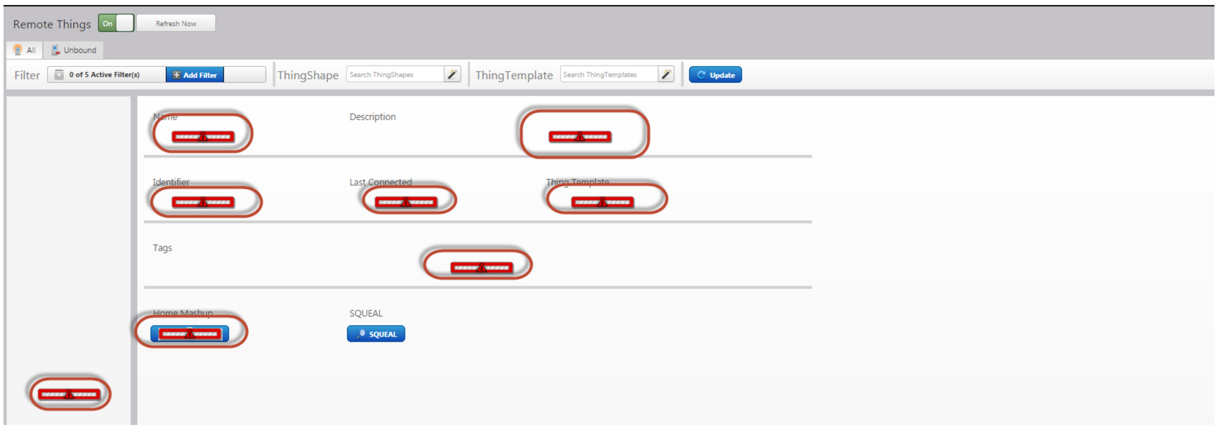
5.1 Platform Logging

System level logs assist administrators with diagnosing problems and understanding the communications between devices and the server. Both server and device logging options are available.
Click the **Monitoring** drop-down menu and select any option under **LOGS.**

The most used log file is the Application Log, which contains all of the messages that the ThingWorx application logs while operating. Depending on your settings, this can display errors only or each individual execution that the platform performs.

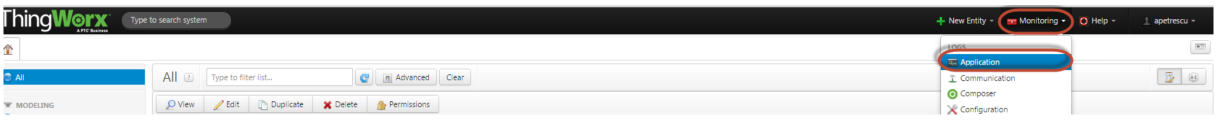
With the help of the Application log, you can detect any type of error that you encounter in your daily work with ThingWorx. After you manage to identify the message error in the *Log Message* dialog box, you can use that information to resolve the encountered issue.

As an example, consider that you are a user that is part of the default Developers group and you are accessing the **Remote Things** option from the **Monitoring** drop-down menu, to see if you have some devices connected to the hosted instance. As soon as you accesses the Remote Things, the window opens and displays errors as shown:

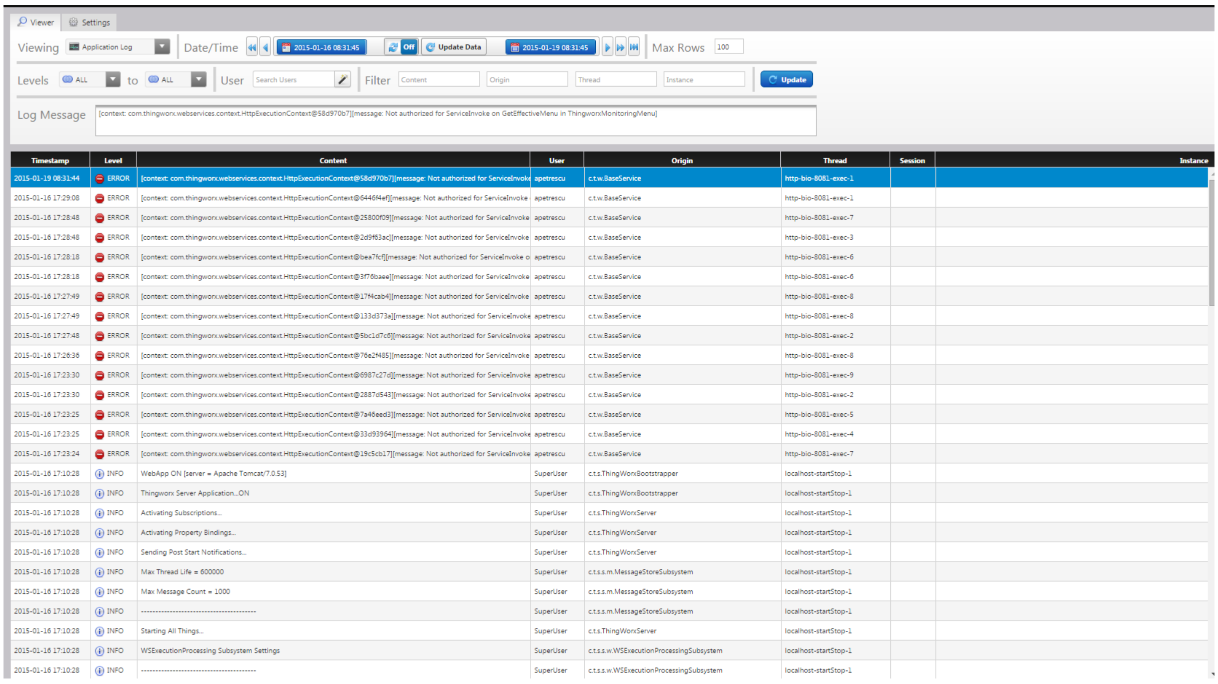


Your first step in this case is to check the Application.log file to determine which error messages have been logged.

You can do this by clicking the **Monitoring** drop-down menu and selecting **Application** under **LOGS** as shown:



This opens the Application.log where you can review the message error, as shown:

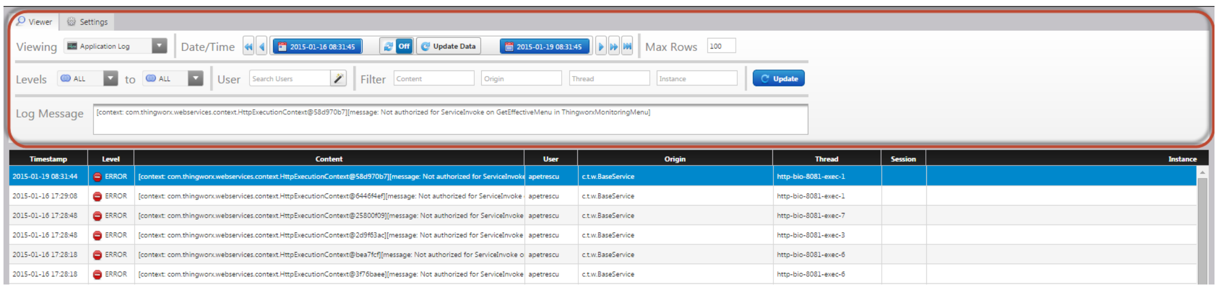


[context: com.thingworx.webservices.context.HttpExecutionContext@58d970b7][message: Not authorized for ServiceInvoke on GetEffectiveMenu in ThingworxMonitoringMenu]

Timestamp	Level	Content	User	Origin	Thread	Session	Instance
2015-01-19 08:31:44	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@58d970b7][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-1		
2015-01-16 17:29:08	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@6446f4ef][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-1		
2015-01-16 17:28:48	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@25800f09][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-7		
2015-01-16 17:28:48	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@2d9f63ac][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-3		
2015-01-16 17:28:18	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@bea7fcf][message: Not authorized for ServiceInvoke o	apetrescu	c.t.w.BaseService	http-bio-8081-exec-6		
2015-01-16 17:28:18	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@3f76baee][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-6		
2015-01-16 17:27:49	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@17f4cab4][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-8		
2015-01-16 17:27:49	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@133d373a][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-8		
2015-01-16 17:27:48	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@5bc1d7c6][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-2		
2015-01-16 17:26:36	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@76e2f485][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-8		
2015-01-16 17:23:30	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@6987c27d][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-9		
2015-01-16 17:23:30	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@2887d543][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-2		
2015-01-16 17:23:25	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@7a46eed3][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-5		
2015-01-16 17:23:25	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@33d93964][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-4		
2015-01-16 17:23:24	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@19c5cb17][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-7		
2015-01-16 17:10:28	INFO	WebApp ON [server = Apache Tomcat/7.0.53]	SuperUser	c.t.s.ThingWorxBootstrapper	localhost-startStop-1		
2015-01-16 17:10:28	INFO	Thingworx Server Application...ON	SuperUser	c.t.s.ThingWorxBootstrapper	localhost-startStop-1		
2015-01-16 17:10:28	INFO	Activating Subscriptions...	SuperUser	c.t.s.ThingWorxServer	localhost-startStop-1		
2015-01-16 17:10:28	INFO	Activating Property Bindings...	SuperUser	c.t.s.ThingWorxServer	localhost-startStop-1		
2015-01-16 17:10:28	INFO	Sending Post Start Notifications...	SuperUser	c.t.s.ThingWorxServer	localhost-startStop-1		
2015-01-16 17:10:28	INFO	Max Thread Life = 600000	SuperUser	c.t.s.s.m.MessageStoreSubsystem	localhost-startStop-1		
2015-01-16 17:10:28	INFO	Max Message Count = 1000	SuperUser	c.t.s.s.m.MessageStoreSubsystem	localhost-startStop-1		
2015-01-16 17:10:28	INFO	--	SuperUser	c.t.s.s.m.MessageStoreSubsystem	localhost-startStop-1		
2015-01-16 17:10:28	INFO	Starting All Things...	SuperUser	c.t.s.ThingWorxServer	localhost-startStop-1		
2015-01-16 17:10:28	INFO	WSExecutionProcessing Subsystem Settings	SuperUser	c.t.s.s.w.WSExecutionProcessingSubsystem	localhost-startStop-1		
2015-01-16 17:10:28	INFO	--	SuperUser	c.t.s.s.w.WSExecutionProcessingSubsystem	localhost-startStop-1		

Next, you need to learn how to read the Application.log. It is divided into two sections.

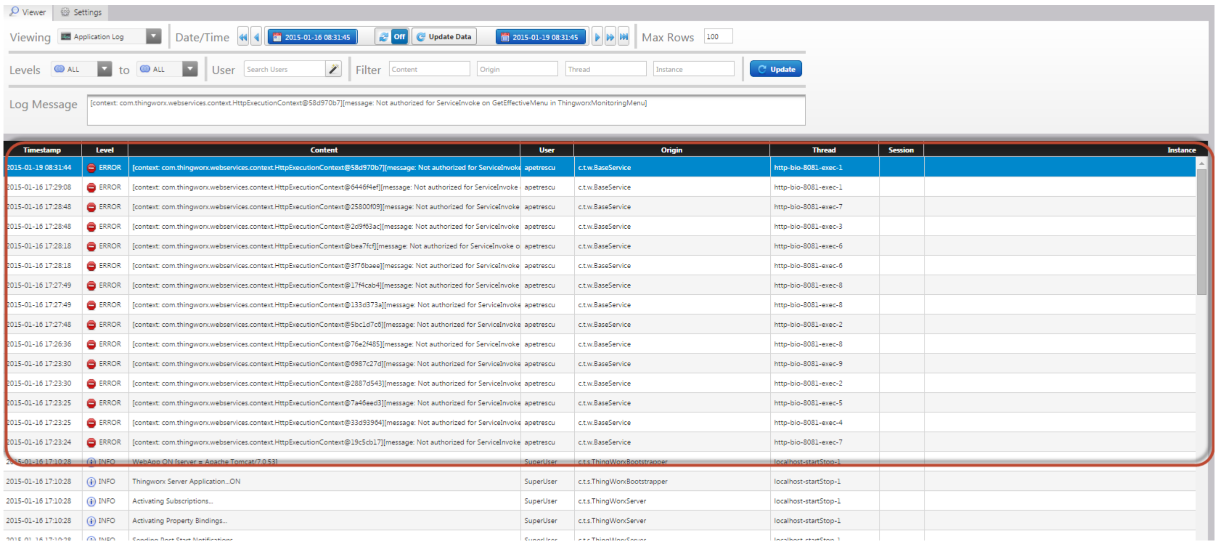
The upper section is where you can select the period of time for which you want to view the Application.log, the tracking level, a filter option for any user or all users from the hosted instance, a filter option and the actual **Log message.** This is where you can read the actual message error as shown:



[context: com.thingworx.webservices.context.HttpExecutionContext@58d970b7][message: Not authorized for ServiceInvoke on GetEffectiveMenu in ThingworxMonitoringMenu]

Timestamp	Level	Content	User	Origin	Thread	Session	Instance
2015-01-19 08:31:44	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@58d970b7][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-1		
2015-01-16 17:29:08	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@6446f4ef][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-1		
2015-01-16 17:28:48	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@25800f09][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-7		
2015-01-16 17:28:48	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@2d9f63ac][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-3		
2015-01-16 17:28:18	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@bea7fcf][message: Not authorized for ServiceInvoke o	apetrescu	c.t.w.BaseService	http-bio-8081-exec-6		
2015-01-16 17:28:18	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@3f76baee][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-6		

The bottom section contains a table that is comprised of the following columns: where you have a table that is made out of the following columns: ***TimeStamp***, ***Level***, ***Content***, ***User***, ***Origin***, ***Thread***, ***Session*** and ***Instance*** as shown:



Viewing: Application Log — Date/Time: 2015-01-16 08:31:45 Off 2015-01-19 08:31:45 — Max Rows: 100

Levels: ALL to ALL — User: Search Users — Filter: Content, Origin, Thread, Instance

Log Message: [context: com.thingworx.webservices.context.HttpExecutionContext@58d970b7][message: Not authorized for ServiceInvoke on GetEffectiveMenu in ThingworxMonitoringMenu]

Timestamp	Level	Content	User	Origin	Thread	Session	Instance
2015-01-19 08:31:44	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@58d970b7][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-1		
2015-01-16 17:29:08	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@6446f4ef][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-1		
2015-01-16 17:28:48	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@25800f09][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-7		
2015-01-16 17:28:48	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@2d9f63ac][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-3		
2015-01-16 17:28:18	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@bea7fcf][message: Not authorized for ServiceInvoke o	apetrescu	c.t.w.BaseService	http-bio-8081-exec-6		
2015-01-16 17:28:18	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@3f76baee][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-6		
2015-01-16 17:27:49	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@17f4cab4][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-8		
2015-01-16 17:27:49	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@133d373a][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-8		
2015-01-16 17:27:48	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@5bc1d7c6][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-2		
2015-01-16 17:26:36	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@76e2f485][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-8		
2015-01-16 17:23:30	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@6987c27d][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-9		
2015-01-16 17:23:30	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@2887d543][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-2		
2015-01-16 17:23:25	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@7a46eed3][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-5		
2015-01-16 17:23:25	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@33d93964][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-4		
2015-01-16 17:23:24	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@19c5cb17][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-7		
2015-01-16 17:10:28	INFO	WebApp ON [server = Apache Tomcat/7.0.53]	SuperUser	c.t.s.ThingWorxBootstrapper	localhost-startStop-1		
2015-01-16 17:10:28	INFO	Thingworx Server Application...ON	SuperUser	c.t.s.ThingWorxBootstrapper	localhost-startStop-1		
2015-01-16 17:10:28	INFO	Activating Subscriptions...	SuperUser	c.t.s.ThingWorxServer	localhost-startStop-1		
2015-01-16 17:10:28	INFO	Activating Property Bindings...	SuperUser	c.t.s.ThingWorxServer	localhost-startStop-1		
2015-01-16 17:10:28	INFO	Sending Post Start Notifications	SuperUser	c.t.s.ThingWorxServer	localhost-startStop-1		

The most commonly used columns for reading the log are:

TimeStamp – Displays the exact time information was logged;

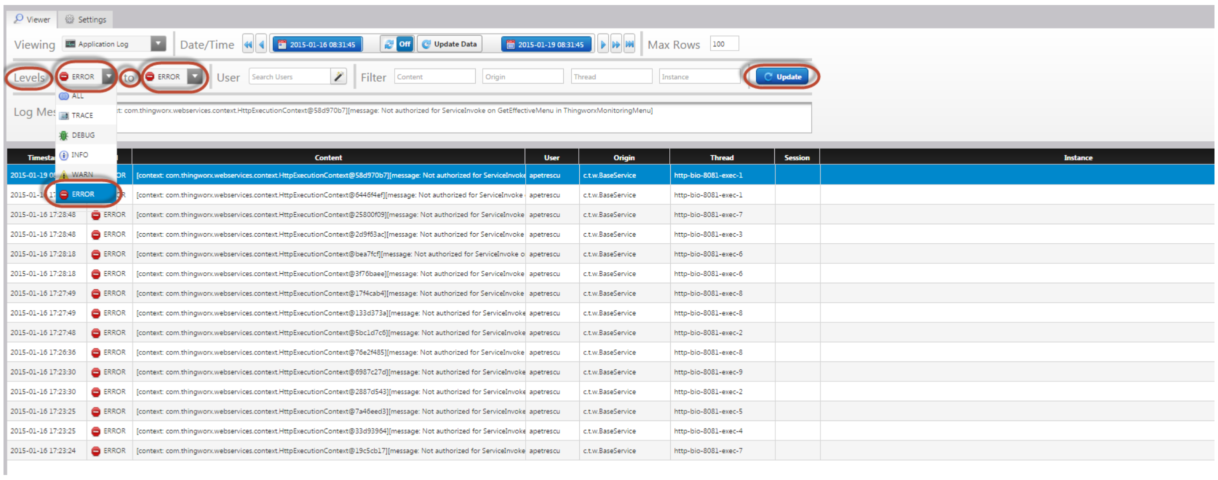
Level – Displays the logging level of the Application.log file. These are: *TRACE, DEBUG, INFO, WARN* and *ERROR.* The most important level of logging is ERROR, because this level displays the errors when you encounter a message error;

Content –Displays the exact message error*;*

User – Displays the user that encountered the message*;*

For this example, you need to perform some filtering in the Application.log file.

First, you filter the logging level to ERROR (to view only the error messages) by selecting the value ERROR in the **Levels** and **to** options and then simply clicking the **Update** button on the upper side of the log file as shown:



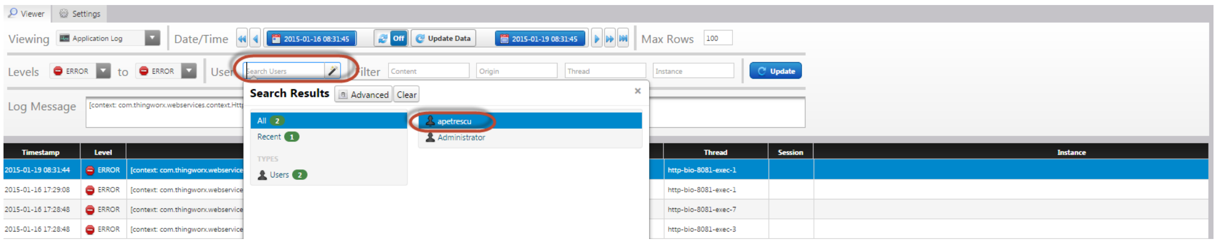
Viewing: Application Log — Date/Time: 2015-01-16 08:31:45 Off 2015-01-19 08:31:45 — Max Rows: 100

Levels: ERROR to ERROR — User: Search Users — Filter: Content, Origin, Thread, Instance

Log Mes t: com.thingworx.webservices.context.HttpExecutionContext@58d970b7][message: Not authorized for ServiceInvoke on GetEffectiveMenu in ThingworxMonitoringMenu]

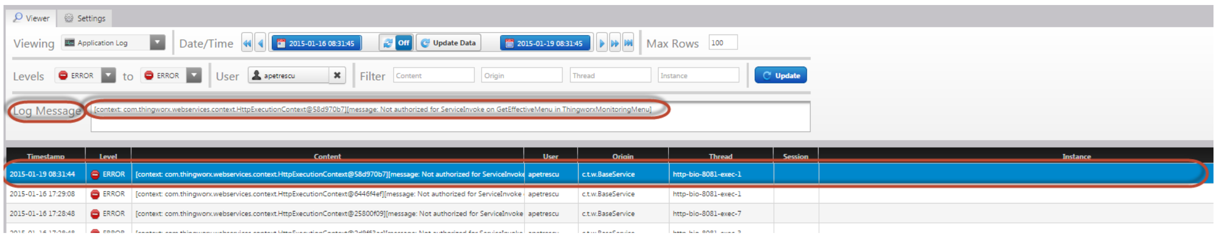
Timesta		Content	User	Origin	Thread	Session	Instance
2015-01-19 0	OR	[context: com.thingworx.webservices.context.HttpExecutionContext@58d970b7][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-1		
2015-01-1 17	R	[context: com.thingworx.webservices.context.HttpExecutionContext@6446f4ef][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-1		
2015-01-16 17:28:48	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@25800f09][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-7		
2015-01-16 17:28:48	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@2d9f63ac][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-3		
2015-01-16 17:28:18	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@bea7fcf][message: Not authorized for ServiceInvoke o	apetrescu	c.t.w.BaseService	http-bio-8081-exec-6		
2015-01-16 17:28:18	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@3f76baee][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-6		
2015-01-16 17:27:49	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@17f4cab4][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-8		
2015-01-16 17:27:49	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@133d373a][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-8		
2015-01-16 17:27:48	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@5bc1d7c6][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-2		
2015-01-16 17:26:36	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@76e2f485][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-8		
2015-01-16 17:23:30	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@6987c27d][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-9		
2015-01-16 17:23:30	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@2887d543][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-2		
2015-01-16 17:23:25	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@7a46eed3][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-5		
2015-01-16 17:23:25	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@33d93964][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-4		
2015-01-16 17:23:24	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@19c5cb17][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-7		

Next, if you want to go even deeper with your filter, you can always filter the log file to only show information for your user (in this case for the user: apetrescu). You can accomplish this by selecting the desired user in the **User** option from the upper side of the log file, as shown:



Timestamp	Level		Thread	Session	Instance
2015-01-19 08:31:44	ERROR	[context: com.thingworx.webservice	http-bio-8081-exec-1		
2015-01-16 17:29:08	ERROR	[context: com.thingworx.webservice	http-bio-8081-exec-1		
2015-01-16 17:28:48	ERROR	[context: com.thingworx.webservice	http-bio-8081-exec-7		
2015-01-16 17:28:48	ERROR	[context: com.thingworx.webservice	http-bio-8081-exec-3		

Now you are ready to read the log file. You need to read all of the lines and identify the message errors. In the example, there are multiple message errors, and they all need to be read. To do this, select every line that is displayed and read the message error in the **Log Message** field on the upper side as shown:

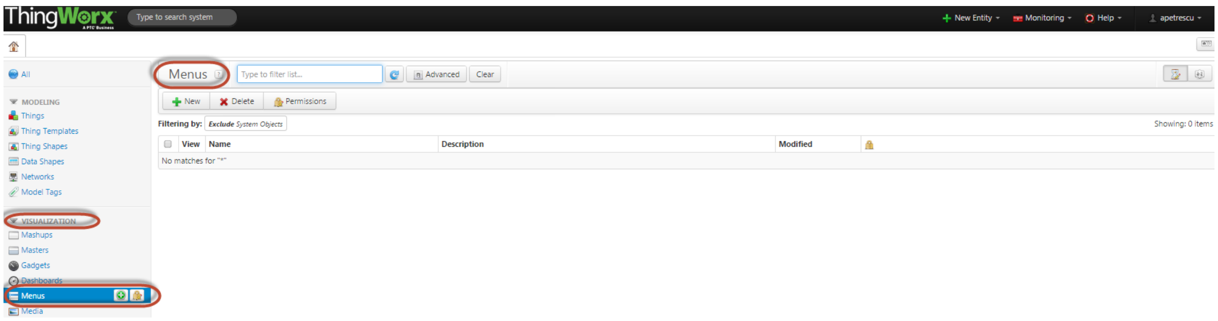


```
[context: com.thingworx.webservices.context.HttpExecutionContext@58d970b7][message: Not authorized for ServiceInvoke on GetEffectiveMenu in ThingworxMonitoringMenu]
```

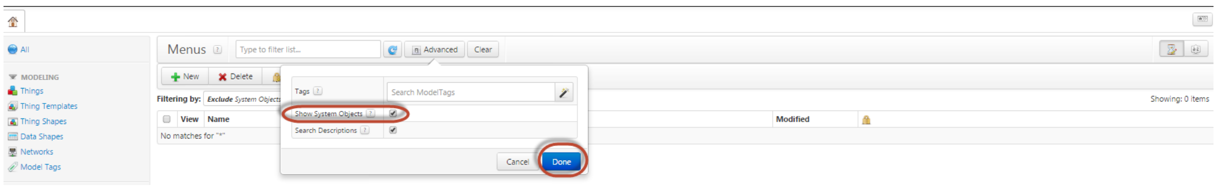
Timestamp	Level	Content	User	Origin	Thread	Session	Instance
2015-01-19 08:31:44	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@58d970b7][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-1		
2015-01-16 17:29:08	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@6446f4ef][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-1		
2015-01-16 17:28:48	ERROR	[context: com.thingworx.webservices.context.HttpExecutionContext@25800f09][message: Not authorized for ServiceInvoke	apetrescu	c.t.w.BaseService	http-bio-8081-exec-7		

In the first line, the following message error is displayed: *[context: com.ThingWorx.webservices.context.HttpExecutionContext@58d970b7][message: Not authorized for ServiceInvoke on GetEffectiveMenu in ThingWorxMonitoringMenu]*

In this message, you only need to be concerned about the part of the message that follows the word *message*, so you only need to interpret the following part:*[message: Not authorized for ServiceInvoke on GetEffectiveMenu in ThingWorxMonitoringMenu]*. With the help of this information, you know that you need to grant the Service Execute permission on the *GetEffectiveMenu* service from *ThingWorxMonitoringMenu* for the user that encountered this error (in this case for apetrescu). To do this, select the *ThingWorxMonitoringMenu* from **Menus** under **VISUALIZATION** as shown:

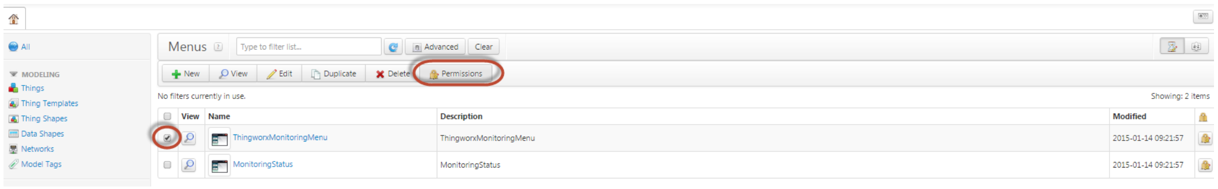


The first time you select the **Menus** option there will be no information displayed. This is because the *ThingWorxMonitoringMenu* is actually a system object. To view system objects, you need to click the **Advanced** button, select the **Show System Objects** check box and click **Done** as shown:

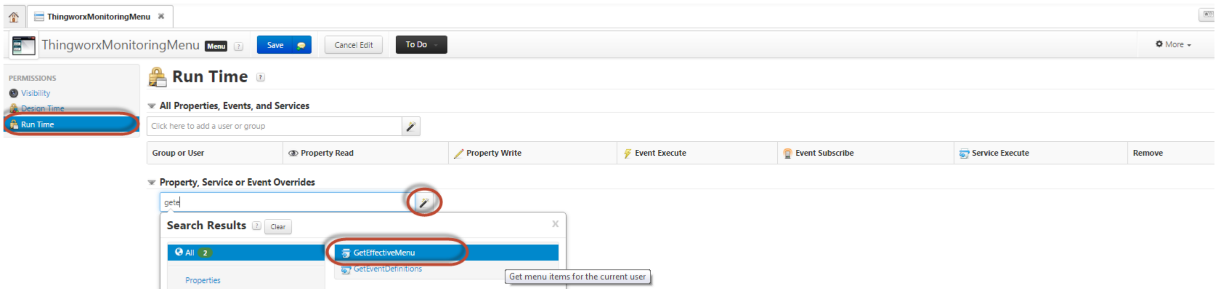


After completing these steps, you are now able to view the system objects under the **Menus** option (these are *ThingWorxMonitoringMenu* and *MonitoringStatus*).

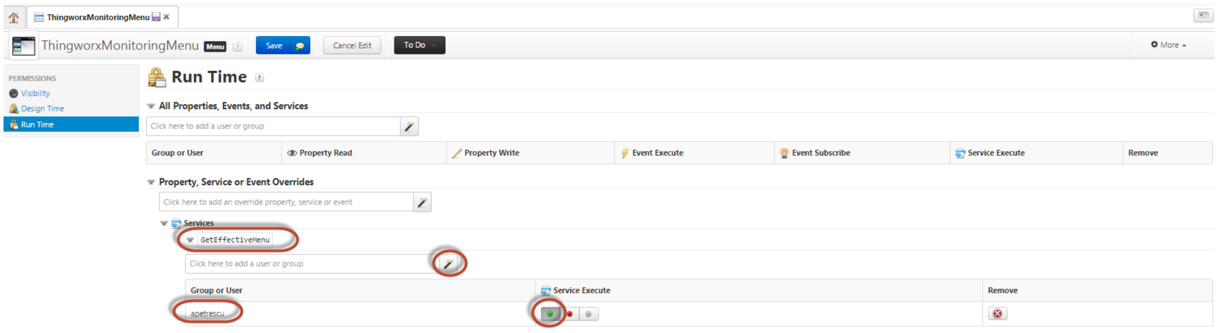
Now, you need to select the box next to *ThingWorxMonitoringMenu* and click the **Permissions** button as shown:



In the opened window, select the **Run Time** option under **PERMISSIONS.** Here, select the service for which you want to grant the service execute permission in the **Property, Service or Event Overrides** option. In the example this service is the *GetEffectiveMenu* as shown:

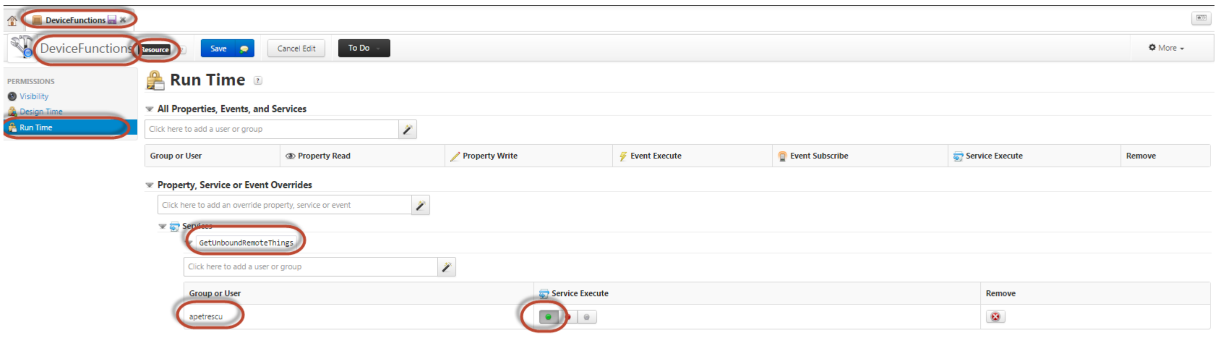


Next, you need to select the user for which you want to grant the service execute permission (in this case it is apetrescu). You can do this by selecting the user in the next box and clicking the green **Allow** button, as shown:

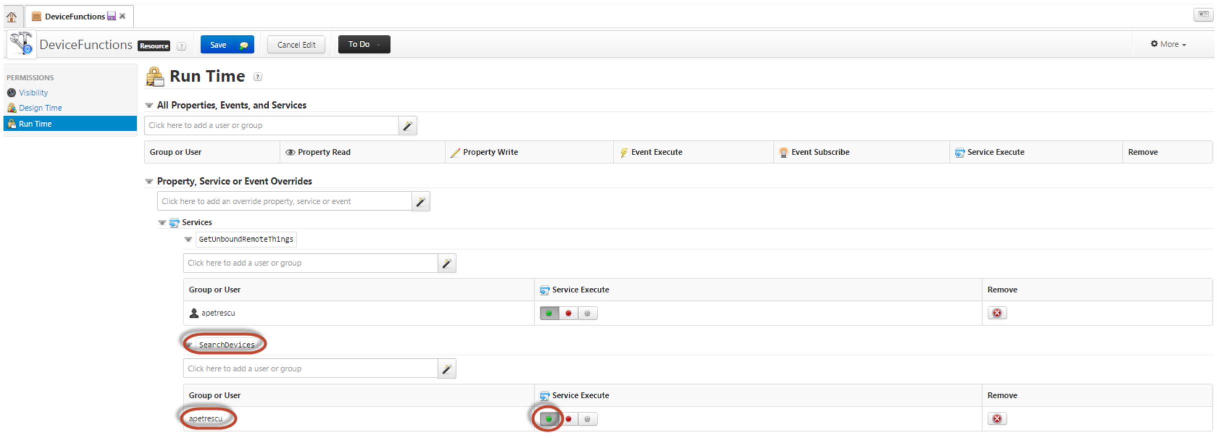

ThingworxMonitoringMenu
Save
Cancel Edit
To Do
More
PERMISSIONS
Visibility
Design Time
Run Time
All Properties, Events, and Services
Click here to add a user or group
Property, Service or Event Overrides
Click here to add an override property, service or event
Services
GetEffectiveMenu
Group or User
Service Execute
Remove
apetrescu

You will do the same with all the message errors that are left in the Application.log. Most of the message errors are repeating so you do not need to perform many modifications. In the example, although there are 15 message error lines, there are only three distinct message errors. First is the one that has already been reviewed in the example and the other two are as follows:

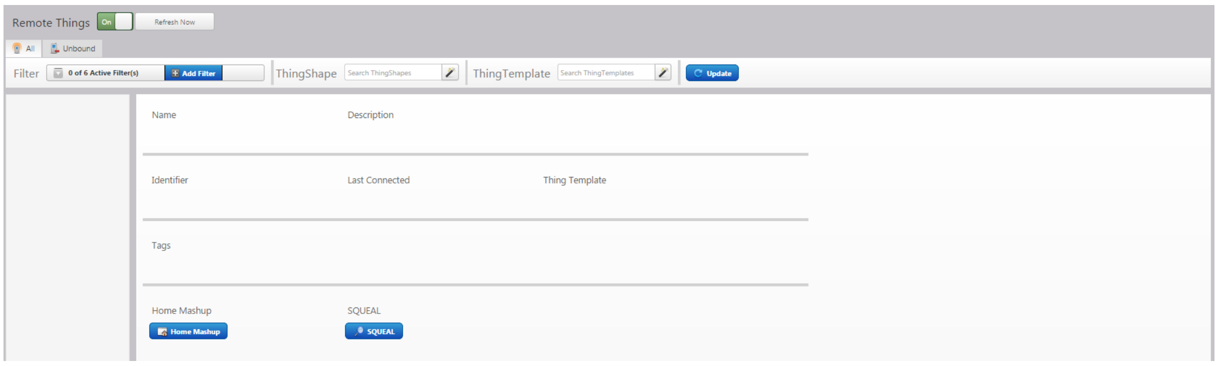
- *[context: com.ThingWorx.webservices.context.HttpExecutionContext@25800f09][message: Not authorized for ServiceInvoke on GetUnboundRemoteThings in DeviceFunctions]* – Indicates that you need to grant Service Execute to the user apetrescu on the *GetUnboundRemoteThings* service for *DeviceFunctions.* You can find DeviceFunctions as a **Resource** under the **System** option. You do the same that you did for the **GetEffectiveMenu** service from *ThingWorxMonitoringMenu.* At the end you will have the information as shown:



- *[context: com.ThingWorx.webservices.context.HttpExecutionContext@2d9f63ac][message: Not authorized for ServiceInvoke on SearchDevices in DeviceFunctions]* – Indicates that you have to grant Service Execute to the user apetrescu on the *SearchDevices* service for *DeviceFunctions*. You can find DeviceFunctions as a **Resource** under the **System** option. You do the same that you did for the **GetEffectiveMenu** service from *ThingWorxMonitoringMenu*. At the end you will have the information as shown:



Now, you have made all of the necessary modifications to view the **Remote Things** option under **Monitoring** without any errors on the page. The Remote Things page appears as shown:



5.2 FAQ (Frequently Asked Questions)

To access the latest version of the FAQ document, please click here

6 USER ADMINISTRATION AND SECURITY

A user account is required to access all aspects of ThingWorx including the Composer, Mashup Runtime and the REST API. User properties can be accessed throughout the ThingWorx modeling environment as well as the Mashup Builder and RunTime environment.

Each user of the system must be defined. A user can have any number of properties, called User Extensions. To modify the User Extensions, you need to edit its ThingShape: Modeling\ThingShapes\UserExtensions. Unlike Things, users do not have Services, Events nor Subscriptions.

A user can belong to any number of groups.

The current User properties (user who is logged in) can be accessed in the Mashup Builder from the User tab in the Data Services and is available at all times in any Mashup at Runtime. Through the Resources Service Library you can also create and clone users at runtime.

Every hosted ThingWorx instance comes with a default System user.

The System User is a built-in ThingWorx user designed with the special purpose of making the management of permissions easier. The System User is found in the Users collection in the Composer.

With the System User, if a service is called indirectly from within a service or subscription, and the System User is permitted to execute the service, the service will be permitted to execute regardless of the user who initially triggered the sequence of events, scripts, or services. However, the user will not be able to directly call 'QueryStreamEntriesWithData' or 'GetDataTableEntryByKey' unless the user is explicitly permitted.

This feature maintains a secure API while making the management of permissions for the indirect service calls simple.

The System User is optional. If the System User is not assigned to a service, ThingWorx will check the permissions of the user who called the services.

As a best practice, use of the System User to ensure proper lock down of your application.

6.1 User Administration - Creating a User

To create a ThingWorx user, a unique name is required.

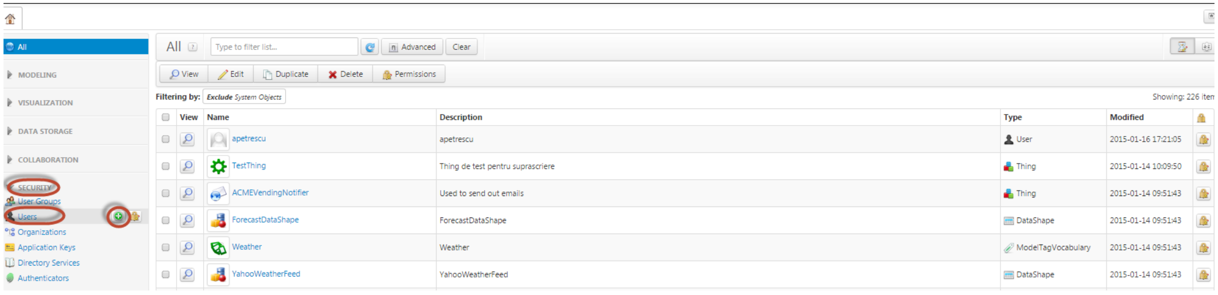
Use the following parameters when naming users (entities):

- Spaces, special characters, or leading numbers are not permitted.
    - It is acceptable to have numbers in or at the end of the name.
- Dashes and underscores are acceptable.
- Names are case-sensitive:
    - For example, johndoe@mymail.com is different than JohnDoe@mymail.com.
- Ensure that names are unique across all of your ThingWorx entities.
- User names are a special case:

- You are able to use e-mail addresses for user names.

Several methods can be used to create a new user:

- Cursor over **Users** and click the plus sign (+):

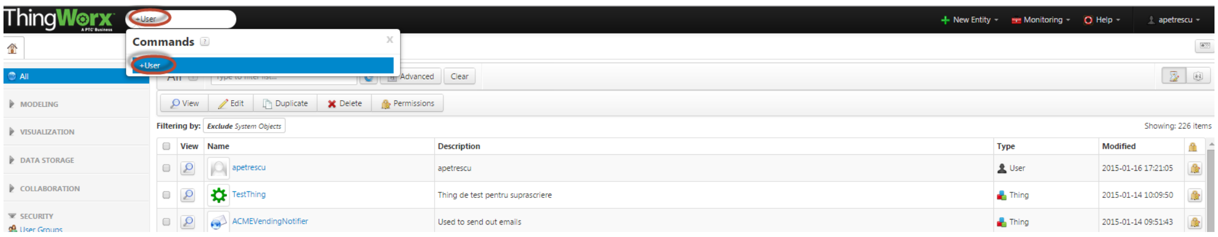


Filtering by: *Exclude System Objects*

Showing: 226 item

	View	Name	Description	Type	Modified	
		apetrescu	apetrescu	User	2015-01-16 17:21:05	
		TestThing	Thing de test pentru suprascriere	Thing	2015-01-14 10:09:50	
		ACMEVendingNotifier	Used to send out emails	Thing	2015-01-14 09:51:43	
		ForecastDataShape	ForecastDataShape	DataShape	2015-01-14 09:51:43	
		Weather	Weather	ModelTagVocabulary	2015-01-14 09:51:43	
		YahooWeatherFeed	YahooWeatherFeed	DataShape	2015-01-14 09:51:43	

- Type +**User** in the spotlight search field.

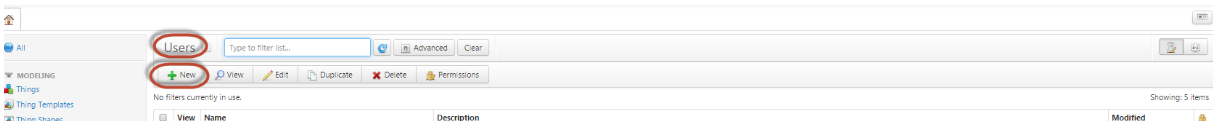


Filtering by: *Exclude System Objects*

Showing: 226 items

	View	Name	Description	Type	Modified	
		apetrescu	apetrescu	User	2015-01-16 17:21:05	
		TestThing	Thing de test pentru suprascriere	Thing	2015-01-14 10:09:50	
		ACMEVendingNotifier	Used to send out emails	Thing	2015-01-14 09:51:43	

- Click **New** on the User page.

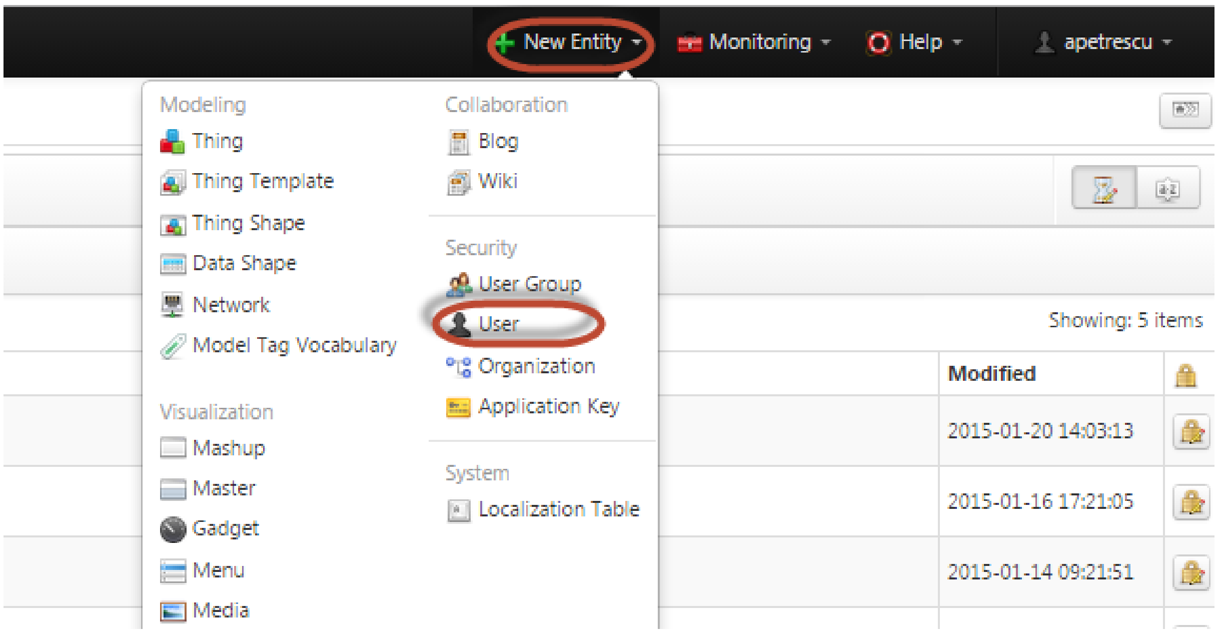


No filters currently in use.

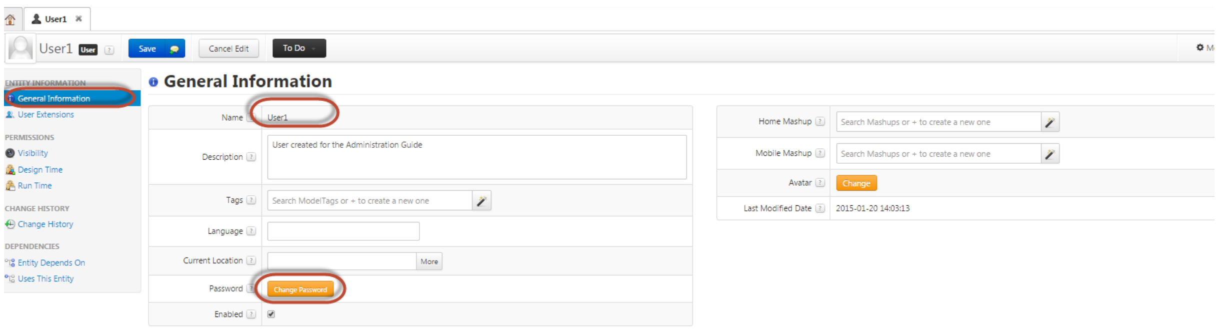
Showing: 5 items

	View	Name	Description	Modified	

- Select the New Entity drop-down menu.



Regardless of which method you use to begin the process, any option results in displaying the Create User page. On this page you must provide a name which is user friendly. You may also specify a password by clicking the **Change Password** button.

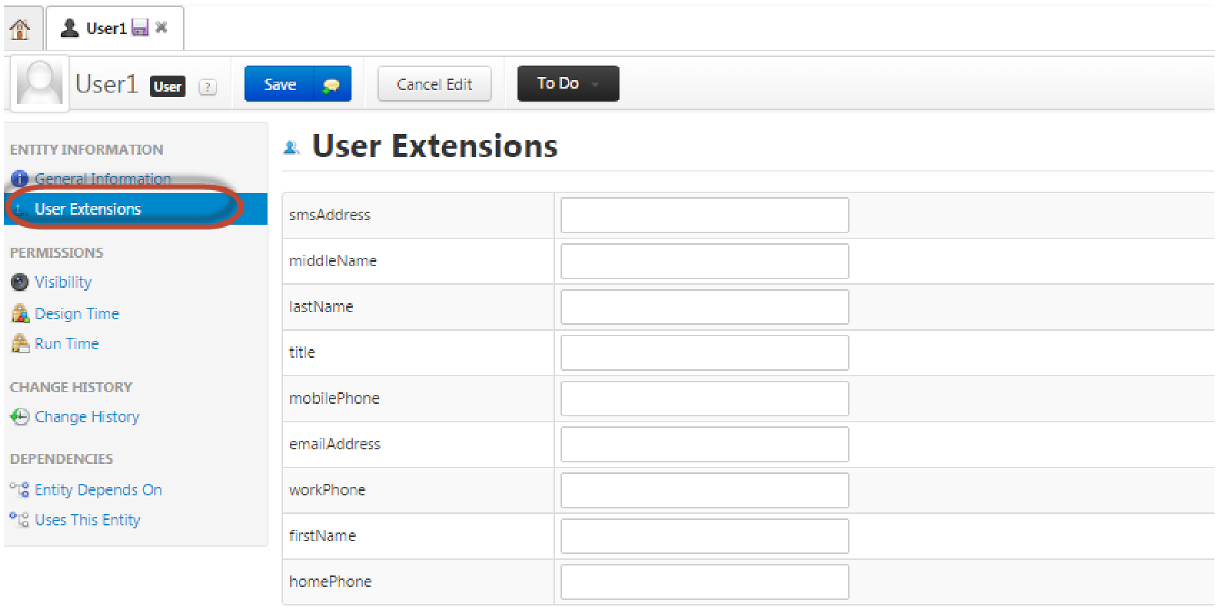


General Information

Name	User1
Description	User created for the Administration Guide
Tags	Search ModelTags or + to create a new one
Language	
Current Location	
Password	
Enabled	✓

Home Mashup	Search Mashups or + to create a new one
Mobile Mashup	Search Mashups or + to create a new one
Avatar	
Last Modified Date	2015-01-20 14:03:13

You may provide additional information about a user, such as their full name and contact information. To add information, navigate to the User Extensions page. User extensions function similarly to properties and may be accessed using services.



User Extensions

smsAddress	
middleName	
lastName	
title	
mobilePhone	
emailAddress	
workPhone	
firstName	
homePhone	

Important!: Creating a user in ThingWorx, does not mean that they will automatically have access to the PTC Web account, community site or to the eLearning libraries.

Creating a user on a hosted instance, grants the user access to that instance only.

6.2 User Administration - Creating a Group

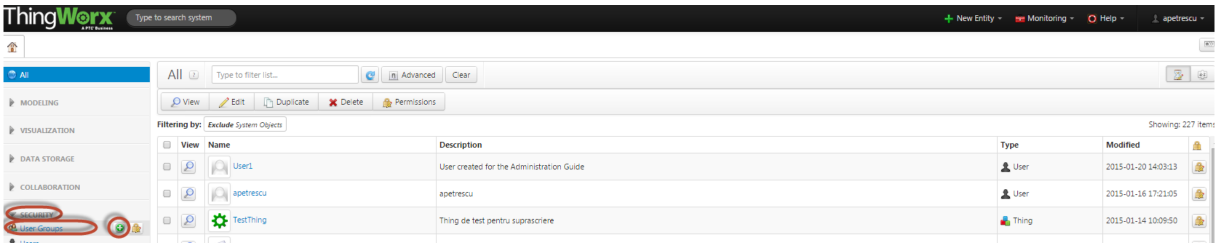
Groups are convenient ways to aggregate a number of users, and then assign permissions at the aggregate level. They are used similarly as groups are typically used in an LDAP system. Groups can also contain other groups, which enables a group to inherit the authorization scheme applied to other groups. This provides a great amount of flexibility in the authorization setup.

Groups can also be managed in Runtime by using the Resources services.

Users can be assigned to ThingWorx groups. The ThingWorx Security Model enables permissions to the Platform to be set at a group level. Assigning permissions at a group level simplifies the administration of permissions throughout the ThingWorx Composer, Runtime environment and REST API. Users and groups can be assigned to User Groups. Permissions stack, with the Overrides taking precedence.

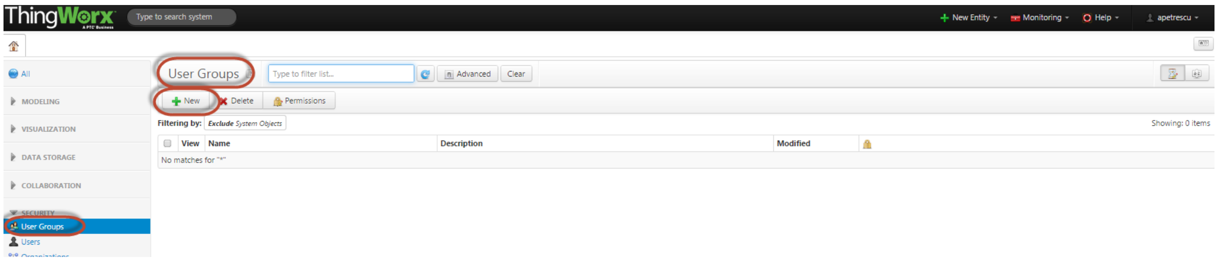
There are several ways to create a new group.

- Click the plus sign (+) next to **Users Groups**.



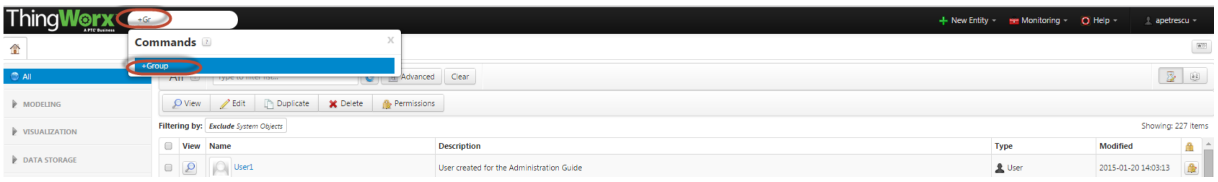
View	Name	Description	Type	Modified
	User1	User created for the Administration Guide	User	2015-01-20 14:03:13
	apetrescu	apetrescu	User	2015-01-16 17:21:05
	TestThing	Thing de test pentru suprascriere	Thing	2015-01-14 10:09:50

- Select the **Users Groups** category and click the +**New** button.



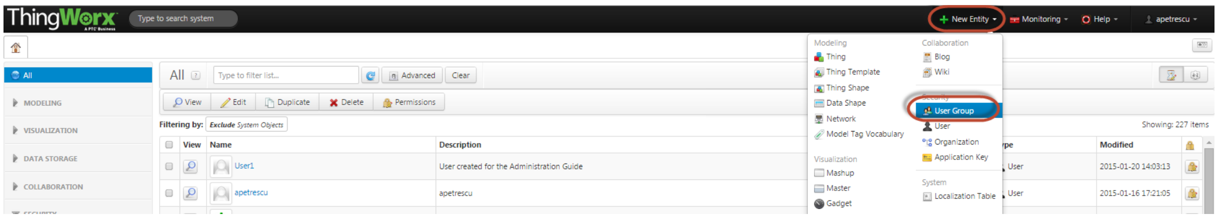
No matches for "*"

- Use the spotlight search field at the top of the Composer in command (+) mode.



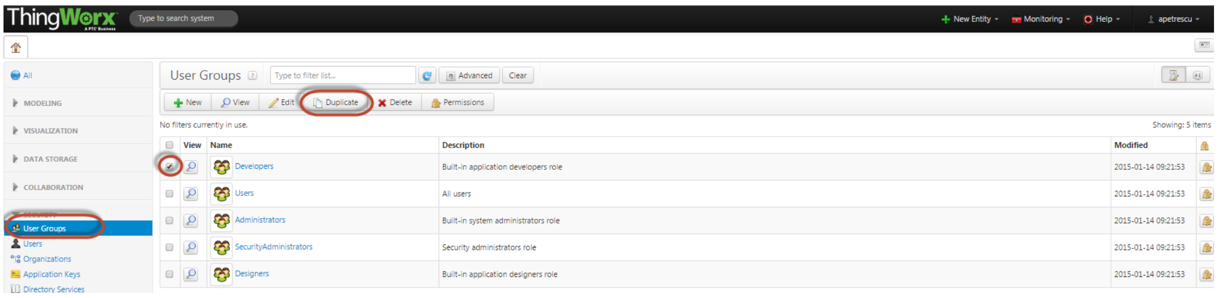
View	Name	Description	Type	Modified
	User1	User created for the Administration Guide	User	2015-01-20 14:03:13

- Select **User Groups** from the + **New Entity** drop-down menu.



View	Name	Description	Type	Modified
	User1	User created for the Administration Guide	User	2015-01-20 14:03:13
	apetrescu	apetrescu	User	2015-01-16 17:21:05

If you want to use an existing user group or a system defined user group as a starting point, you can select **Duplicate** from the Explorer.



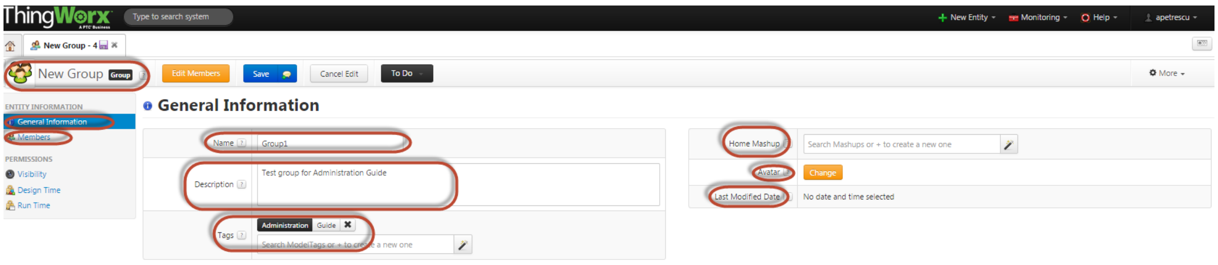
No filters currently in use.

Showing: 5 items

View	Name	Description	Modified
✓	Developers	Built-in application developers role	2015-01-14 09:21:53
	Users	All users	2015-01-14 09:21:53
	Administrators	Built-in system administrators role	2015-01-14 09:21:53
	SecurityAdministrators	Security administrators role	2015-01-14 09:21:53
	Designers	Built-in application designers role	2015-01-14 09:21:53

When you create a user group, it automatically defaults to the **General Information** section. The **Name** field is mandatory (shown as a red outline and red icon, which disappears once the name information is added.) All other information fields are optional. You can provide the following information:

- **Description**. – Type a short description about the group that you are creating.
- **Model Tag**(s) - See Model Tags for more information.
- **Home Mashup** – This defines the mashup. The mashup displays as associated with the Thing-Shape in SQUEAL, but can also be retrieved using a service for other usages.
- **Avatar** – Upload your image to be displayed with the data shape within the Composer.
- **Last Modified Date** – This is a read-only field that is generated by the system. This field displays by default in almost every entity within ThingWorx.



Name: Group1

Description: Test group for Administration Guide

Tags: Administration Guide

Search ModelTags or + to create a new one

Home Mashup: Search Mashups or + to create a new one

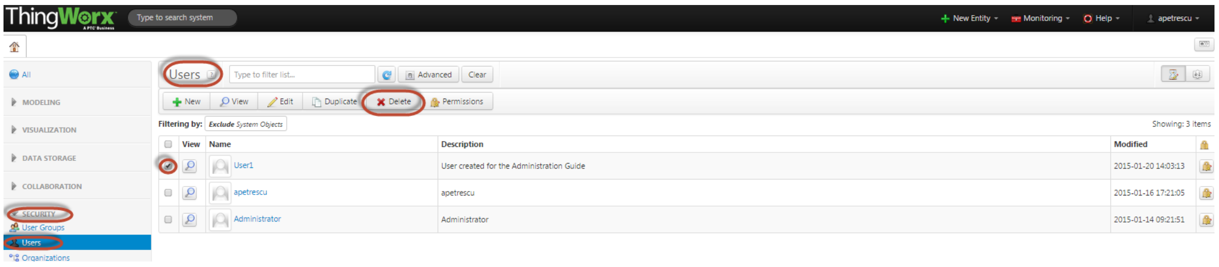
Avatar

Last Modified Date: No date and time selected

Other than general information, all that needs to be defined for a user group is the actual assigned members. Click **Members** to view current members or click **Edit Members** to edit the membership of the user group. You also have access to the **Permissions** section, which enables you to define *visibility*, *design time*, and *runtime* permissions.

6.3 User Administration – Deleting a User

To delete a user from ThingWorx, first select the **Users** category under **Security**, select the user that you want to delete and click the **Delete** button as shown:

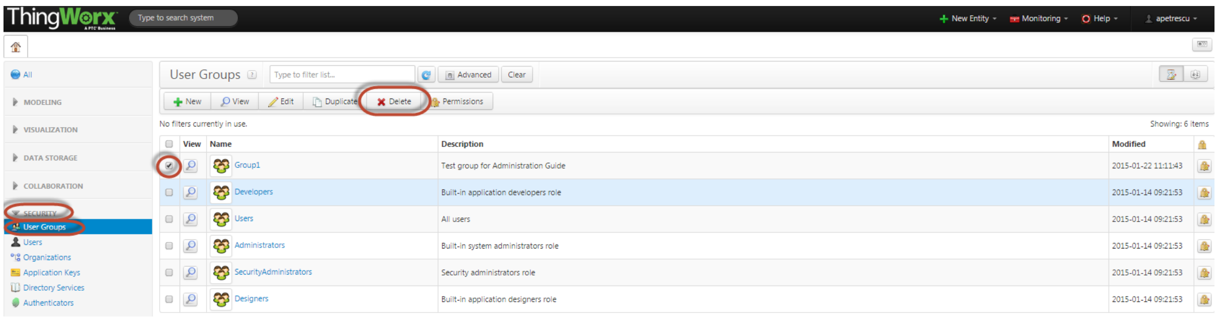


Filtering by: Exclude System Objects — Showing: 3 items

Name	Description	Modified
User1	User created for the Administration Guide	2015-01-20 14:03:13
apetrescu	apetrescu	2015-01-16 17:21:05
Administrator	Administrator	2015-01-14 09:21:51

6.4 User Administration – Deleting a Group

To delete a group from ThingWorx, first select the **User Groups** category under **Security**, select the group that you want to delete and click the **Delete** button as shown:

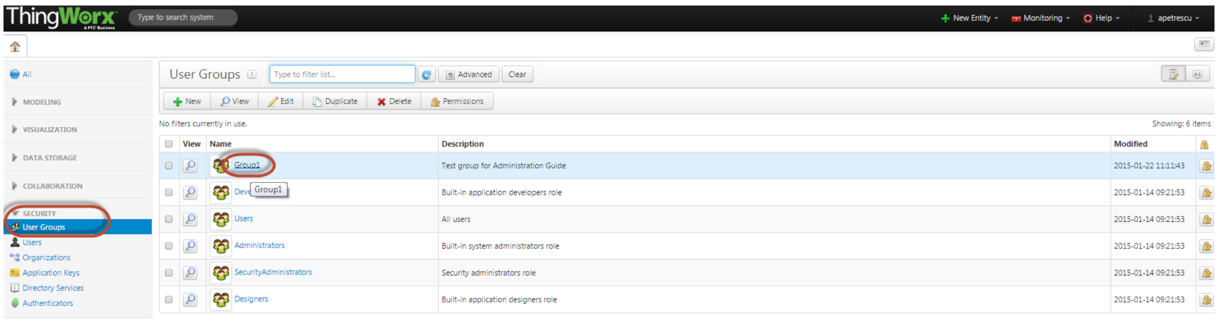


No filters currently in use. — Showing: 6 items

Name	Description	Modified
Group1	Test group for Administration Guide	2015-01-22 11:11:43
Developers	Built-in application developers role	2015-01-14 09:21:53
Users	All users	2015-01-14 09:21:53
Administrators	Built-in system administrators role	2015-01-14 09:21:53
SecurityAdministrators	Security administrators role	2015-01-14 09:21:53
Designers	Built-in application designers role	2015-01-14 09:21:53

6.5 User Administration – Adding a User to a Group

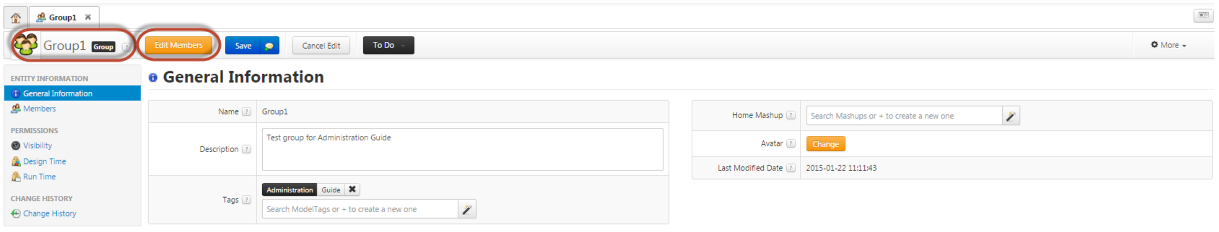
To add a user to a group, select the **User Groups** category under **Security** and click the group to which you want to add a user, as shown:



No filters currently in use. — Showing: 6 items

Name	Description	Modified
Group1	Test group for Administration Guide	2015-01-22 11:11:43
Developers	Built-in application developers role	2015-01-14 09:21:53
Users	All users	2015-01-14 09:21:53
Administrators	Built-in system administrators role	2015-01-14 09:21:53
SecurityAdministrators	Security administrators role	2015-01-14 09:21:53
Designers	Built-in application designers role	2015-01-14 09:21:53

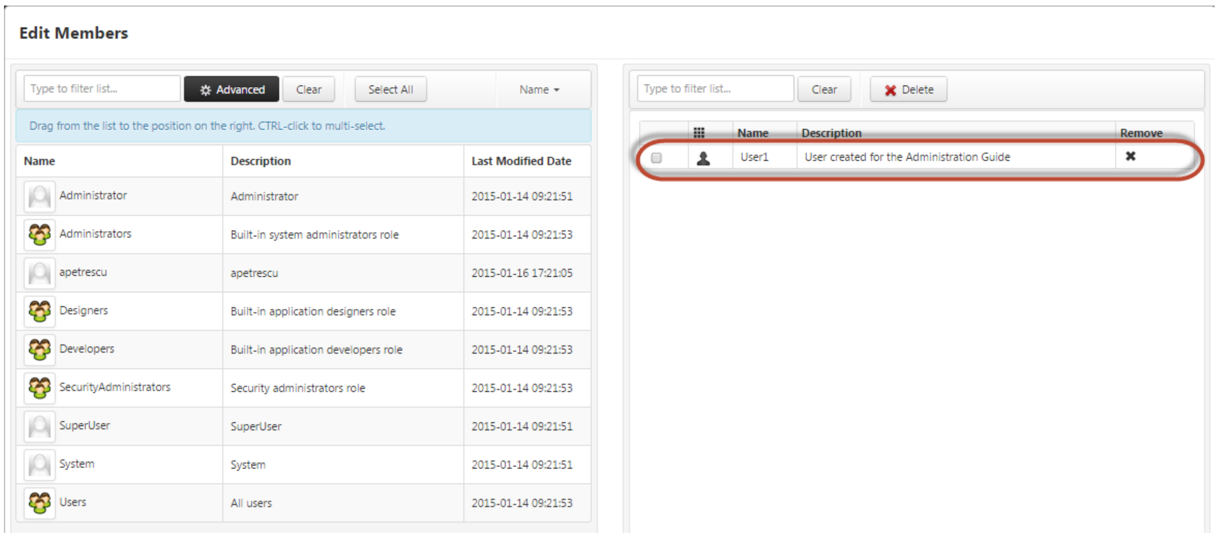
Next, click the **Edit Members** button, as shown:



General Information

Name	Group1
Description	Test group for Administration Guide
Tags	Administration Guide
Home Mashup	Search Mashups or + to create a new one
Avatar	Change
Last Modified Date	2015-01-22 11:11:43

Next, the Edit Members window opens. Here you can drag and drop the user, users or groups that you want to add in the Group1 group. Remember, a group can contain a user or even another group. As soon as you drag and drop a user from the left panel to the right panel, that user will no longer be available in the left panel (you basically perform a move operation). This helps you administrate the group, so you will not be able to add the same entity twice:



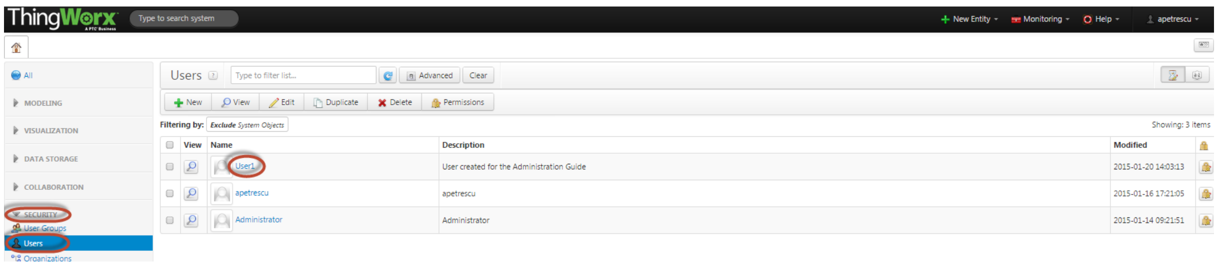
Name	Description	Last Modified Date
Administrator	Administrator	2015-01-14 09:21:51
Administrators	Built-in system administrators role	2015-01-14 09:21:53
apetrescu	apetrescu	2015-01-16 17:21:05
Designers	Built-in application designers role	2015-01-14 09:21:53
Developers	Built-in application developers role	2015-01-14 09:21:53
SecurityAdministrators	Security administrators role	2015-01-14 09:21:53
SuperUser	SuperUser	2015-01-14 09:21:51
System	System	2015-01-14 09:21:51
Users	All users	2015-01-14 09:21:53

Name	Description	Remove
User1	User created for the Administration Guide	✖

The image above shows that User1 is no longer available in the left panel because it has been dragged and dropped into the right panel.

6.6 User Administration – Changing a User's Password

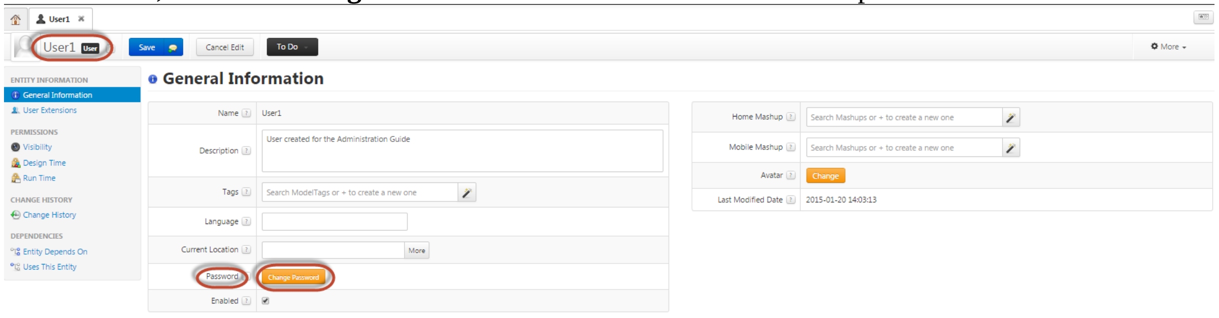
To change a user's password, select the **Users** category under **Security** and click to select the desired user:



Filtering by: *Exclude System Objects* Showing: 3 items

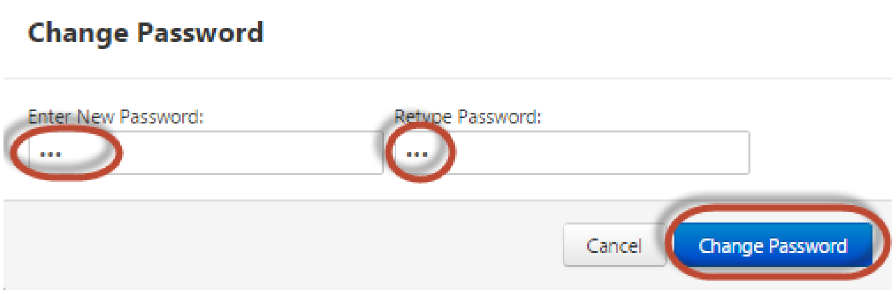
View	Name	Description	Modified
	User1	User created for the Administration Guide	2015-01-20 14:03:13
	apetrescu	apetrescu	2015-01-16 17:21:05
	Administrator	Administrator	2015-01-14 09:21:51

Next, click the **Change Password** button next to the Password option:



Name	User1
Description	User created for the Administration Guide
Tags	Search ModelTags or + to create a new one
Language	
Current Location	
Password	Change Password
Enabled	✓
Home Mashup	Search Mashups or + to create a new one
Mobile Mashup	Search Mashups or + to create a new one
Avatar	Change
Last Modified Date	2015-01-20 14:03:13

In the Change Password window, simply type the new password that you want to set and press the **Change Password** button:



6.7 Security – Managing Access Privileges

There is a very granular security model available within the ThingWorx platform. There are two sets of permissions, one for design time and one for run time. The design time permissions are for managing who is permitted to modify the model (create, read, update, and delete entities), while the run time permissions determine who can access data, execute services, and trigger events on a Thing (which includes DataTables, Streams, and Users).

For each permission, you can explicitly permit a user or group to be able to perform an action (such as edit a Thing) or explicitly deny a group the ability to perform an action (for example, the Users Group is not permitted to edit a Thing). You can apply permissions at the Group level and at the User level. An explicit denial of a privilege always overrides a privilege grant. It is also important to note that security checks default to not permit an operation. If no specific grant has been given to a user, then that operation will be denied.

The design time permissions are for standard CRUD operations (Create, Read, Update, Delete) on any entity. This controls the model definition and who can make changes to the model. These permissions can be applied to individual entities (Read, Update, or Delete a specific Thing, ThingShape, DataShape and so on) or can be broadly applied to an entire entity collection (Create, Read, Update, or Delete Things, ThingShapes, DataShapes and so on). If you apply permissions to the collection, you can override, add or subtract to/from those permissions at the individual entity level.

The run time permissions are for Property Read, Property Write, Event Execute, and Service Execute. This can be done at the All Properties or All Services level for a Thing, but you can also set specific permission overrides for individual properties, services, or events.

You can set run time permissions at a Thing, ThingTemplate, or collection level. An example of a collection is Things or ThingTemplates. These are inherited by all items in the collection. You can also set run time and design time permissions at the individual ThingTemplate level, and all Things derived from that ThingTemplate will inherit those settings. All inherited permissions can be overridden at the individual Thing level.

The abstract entities within the model do not have run time permissions. These entities include ThingShapes, DataShapes, and Groups.

Important!: The collections are not written as expected if the corresponding users/groups do not exist BEFORE you perform the Import from Storage on the target server. To have permissions imported properly, the users and groups must first exist in the platform. First you will need to create an Export of your users and groups, then you will import what you created and then import the rest of your model.

Note: Collection level permissions (things, thingtemplates, logs and so on) export or import ONLY when you perform an Export to Storage and Import from Storage. If you export or import from File the collection level permissions will apply but only on an Entity level.

How to Set Up ThingWorx Security Video.

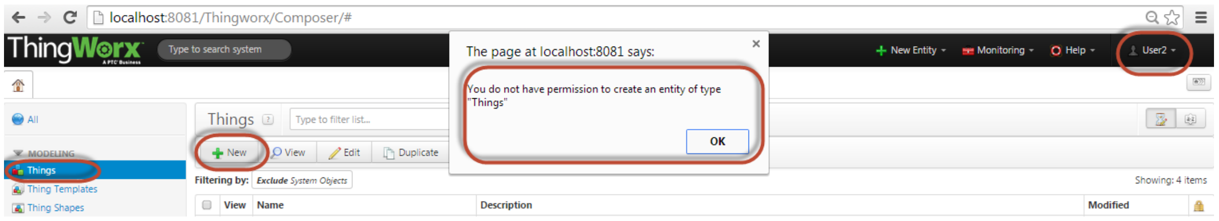
All entities in the ThingWorx model have Design Time and Run Time permissions. ThingTemplates also have Designtime Instance and Runtime Instance permissions. These additional permissions are inherited by all implementing Things of a ThingTemplate. They mimic Design Time and Run Time permissions, but apply only to Things.

This enables you to set permissions for all Things that are derived from a single ThingTemplate at the ThingTemplate level. You can then provide specific overrides/grants/denials at the Thing level, as required. Common permissions can be managed at the ThingTemplate level, making maintenance of permissions easier.

At runtime, each property read, property write, event execution, and service execution can have explicit overrides. This enables you to grant or deny specific capabilities to any group or user to meet a specific need.

To access the serviced overrides, navigate to the entity list, select a specific entity, and enter the security editor for that entity. In the runtime view, there is an override link for each action mentioned above. When you click that link, a list of specific events, properties, and services for that entity displays. You can then assign an explicit grant or denial for any action.

For example, consider that you created a new user such as User2, that by default has no permissions. To grant him certain access privileges you can add him to certain groups that you have created. You can also add him to one of the two default groups (**Developers** and **Administrators)** or you can grant him permissions on the entity to which you want like him to have access. Below is an attempt to create a new Thing entity with a newly created user (in this case, User2):



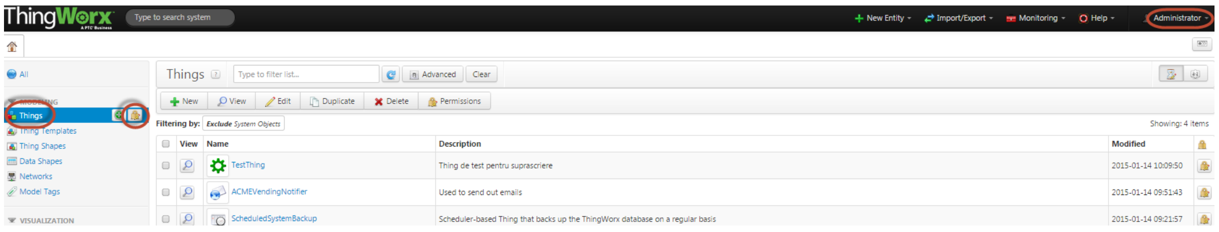
The page at localhost:8081 says:

You do not have permission to create an entity of type "Things"

OK

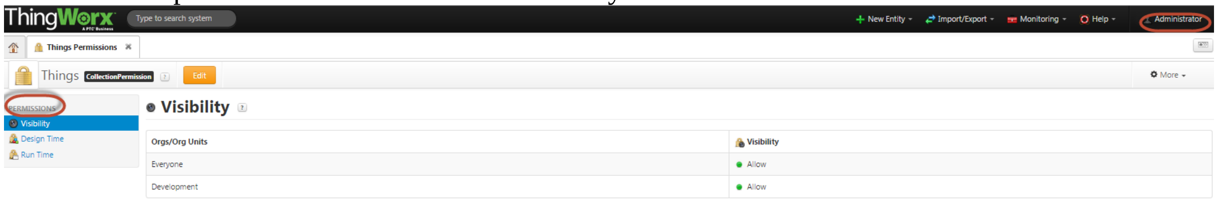
You can see that you do not have any access on the Things option with a newly created user.

To grant him access only on the Things option, you must log into ThingWorx with an administrator, cursor over the **Things** option and click the **lock** button as shown:



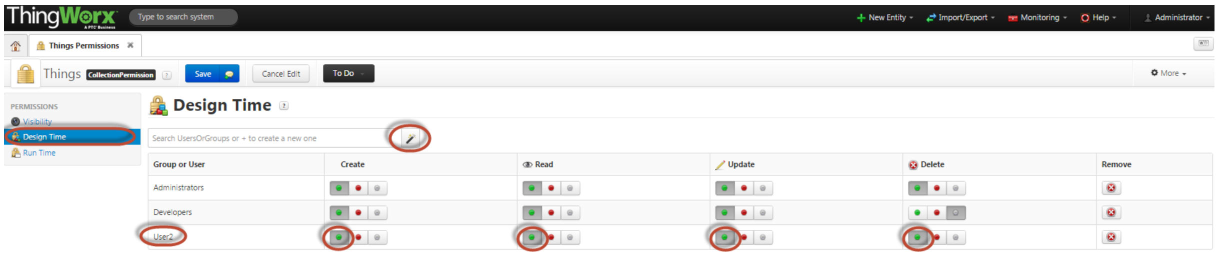
View	Name	Description	Modified
	TestThing	Thing de test pentru suprascriere	2015-01-14 10:09:50
	ACMEVendingNotifier	Used to send out emails	2015-01-14 09:51:43
	ScheduledSystemBackup	Scheduler-based Thing that backs up the ThingWorx database on a regular basis	2015-01-14 09:21:57

This opens the Permissions window where you can click the **Edit** button as shown:



Orgs/Org Units	Visibility
Everyone	● Allow
Development	● Allow

The window is by default opened on the **Visibility** permission. Navigate to the **Design Time** permission and add the newly created user (User2) and grant him the Allow permission on all four options (press the green button for *Create*, *Read*, *Update*, *Delete*) as shown:

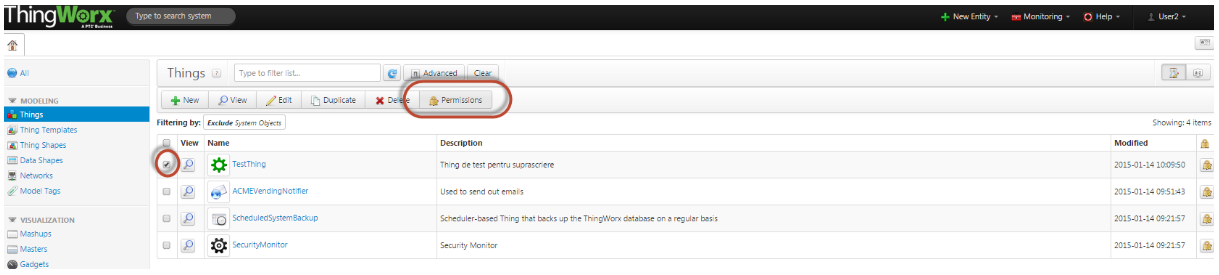


Group or User	Create	Read	Update	Delete	Remove
Administrators					
Developers					
User2					

Now, your user is able to Create, Read, Update or Delete everything from the Things entity.

If desired, you can apply this at a more granular level and simply grant or revoke access privileges to a certain user on a certain thing. When you want to revoke permissions, simply repeat the above steps for a thing and select the red button (Revoke).

When you want to grant or revoke access to a certain entity, you must select that entity and click the **Permissions** button as shown:



View	Name	Description	Modified
	TestThing	Thing de test pentru suprascriere	2015-01-14 10:09:50
	ACMEVendingNotifier	Used to send out emails	2015-01-14 09:51:43
	ScheduledSystemBackup	Scheduler-based Thing that backs up the ThingWorx database on a regular basis	2015-01-14 09:21:57
	SecurityMonitor	Security Monitor	2015-01-14 09:21:57

6.8 Security – Organizations

ThingWorx Organizations: Multi-Tenancy for the Collaborative World

Organizations in ThingWorx are hierarchical structures that enable the user to assign visibility to entities in the ThingWorx model. Multiple organizations and organizational structures can be granted visibility to the same assets within a ThingWorx model.

Click here to view a tutorial video on organizations and visibility.

Visibility in Organizations

Visibility is a simple form of access control. If an entity is visible to members of an organizational unit, then those members have access to the entity, and the underlying, granular security model determines what specific interaction any users that are members of that organization unit may have with a specific asset. If a user in the system is not granted visibility, then that asset essentially does not exist within that user's domain. That user cannot view the asset, list it, or interrogate that asset's namespace.

In the ThingWorx platform, it is possible to define the visibility rules to make specific Things only visible to a single organization, or to permit multiple organizations visibility to the same asset. An organization is comprised of organizational units.

Illustrating the Concept of Organizations

To illustrate the concept of organizations in ThingWorx, we will use a vending machine as an example of an asset. Beverage Company A leases a vending machine (named VM101 in ThingWorx) to Operating Company B. Operating Company B outsources the maintenance and inventory of the vending machine to Supplier C. Operating Company B also leases vending machines from Beverage Company D.

Using this scenario, Company A, B, and C may require visibility to the VM101 vending machine, while Beverage Company D does not. After initial visibility is set, specific data elements and services can also be configured to be accessible only to the companies that require access. The "visibility test" is always applied before other security configuration rules are evaluated.

Granting Visibility

There are three places to set up visibility in ThingWorx:

1. The collection level (Things or ThingShapes), which applies the visibility settings to all members of the collection.
2. The individual Entity level (VM101 vending machine)
3. ThingWorx ThingTemplate Instance Visibility (only applicable to ThingTemplates)
   Instance visibility settings remain intact to any Thing that is derived from that ThingTemplate.

When visibility to an asset is granted at lower levels in the organization hierarchy, they are automatically granted to the higher levels. For example, if a line operator is granted visibility to their line, but there is a supervisor organization unit for all lines in the organization hierarchy, the supervisor organization unit is automatically granted visibility to the assets that the child organizational unit has been granted.

There is one exception to the roll up model: granting an entire organization visibility to an asset. When an entire organization is added, the organization and all of its subunits are assigned visibility to that Entity.

Removing the Users Group from the Everyone Organization

The default setting in ThingWorx permits everyone to view everything. This is accomplished by having a default system organization, the Everyone Organization, be assigned visibility to every collection (Things, ThingShapes, ThingTemplates and so on.) in the system. The Everyone Organization contains the Users group, meaning all system users have been granted visibility to all system collections. When you remove the Users group from the Everyone Organization, users and groups in the system will not have visibility to any model artifacts until they are explicitly granted. The only exception is members of the Administrators group.

Follow the steps below to remove the Users group from the Everyone Organization:

1. In the ThingWorx Composer, click **Organizations**.
2. Click **Everyone**.

3. In the **Members** section, remove the Users by clicking the **x**.
4. Click **Save**.

Creating an Organization and Assigning Users or Groups

1. In the ThingWorx Composer, click **Organizations**.
2. Click the plus sign (+) to create a new organization.
3. Type a **Name**, **Description**, **Tags**, and any necessary login information.
4. Click **Save**.
5. Click **Organizations**.
6. Click **Edit**.
7. Click the organization name to add new units.
8. Click the green plus sign (+) button.
9. Type the **Name**.
10. Click **Done**.
11. Click **Save**.
12. To assign users or groups to the organization, select a group from the list or add a new group.

NOTE: Users can be added, but best practice is to add groups before users.

Configuring Visibility for an Asset

NOTE: Users and Groups must be defined before organizations can be configured.

1. In the Explorer, open the Thing or Entity.
2. Click **Visibility**.
3. Click **Add Org/Org Units**.
4. Click the magic picker and select the appropriate organization.
5. Click an organizational unit to add it to the organization.
6. Click **Done**.
7. Click **Save**.

Note: To delete an organizational unit from an organization, click the ***x*** *on the appropriate unit at the right of the page.*

Note: To add an entire organization, click ***Add Entire Organization****.*

Configuring Instance Visibility

Instance visibility applies only to ThingTemplates, and visibility set at this level is inherited by any entities that use that ThingTemplate.

1. In the Explorer, open the ThingTemplate.
2. In the Explorer, click **Visibility Instance**.
3. Click **Add Org/Org Units**.
4. Click the magic picker and select the appropriate organization.

5. Click an organizational unit to add it to the organization.
6. Click **Done**.
7. Click **Save**.

Form Login Pages for Organizations

In ThingWorx, unique login pages can be created for each of your organizations. This feature has two primary use cases:

1. To provide the ability to uniquely brand the entry point per organization
2. When a user logs in through this entry point, upon successful authentication, they are redirected to the organization's Home Mashup

If the user logs in with a mobile device, the user is redirected to the organization's Mobile Mashup.

An organization's form page is located at: *<server name>/ThingWorx/FormLogin/<Organization Name>*.

The following items can be customized:

- **Login Image**: Specify the image that displays on the login form
- **Login Prompt**: Specify the message that displays on the login form(such as: Welcome to ThingWorx)
- **Login Style**: Specify the text and background colors on the login form
- **Login Button Style**: Specify the color of the Submit button and text on the login form

Creating Form Login Pages for an Organization

1. In ThingWorx Composer, click **Organizations** and select the appropriate organization.
2. In the **Login Prompt** field, type a message to appear on the login page.
3. In the **Login Image** field, click **Change** to upload an image.
4. In the **Login Style** field, use the magic picker to select the colors of the login page.
5. In the **Login Button Style** field, use the magic picker to select the colors of the login button.
6. Click **Save**.
7. To view the login page, go to: *<server name>/ThingWorx/FormLogin/<Organization Name>*
8. Type your **Name** and **Password**.
9. Click **Submit**.

6.9 Security – Application Key

Application keys are security tokens that can be used to login to the ThingWorx application instead of using standard credentials. You can pass an application key on the URL of a request by adding the following query parameter:

appKey=1CB45438-0000-0000-B95C-892762FD0000

The request will then be run using the security context of the user associated with the application key. By default, no session is created when a request is made using an application key. This is the recommended way for other systems or applications to make a request to the ThingWorx application. However, if you wish to create a session, you can use the following query parameter in addition to passing in the application key:

x-ThingWorx-session=true

Follow the steps below to create an application key. The created key will assume the security settings granted to the associated user in the **User Name Reference** field:

1. In ThingWorx Composer, click **Application Keys**.
2. Click **New**.
3. Type a **Name** and any optional details.
4. In the **User Name Reference** field, select the user associated with the application key.
5. Click **Save**.
6. The application key displays in the **keyId** field.

You are also able to create this application key programatically through a service. The script will display similar to the script below:

```
var params = {
		description: undefined /* STRING */,
		name: undefined /* STRING */,
		user: undefined /* USERNAME */,
		whitelist: undefined /* STRING */
};
// no return
Resources["EntityServices"].CreateApplicationKey(params);


//Get the generated Key's actual appKey value
var KeyID = ApplicationKeys["YOURKEYNAME"].GetKeyID();
```

7 UPGRADE CONSIDERATIONS

7.1 Upgrade the PTC's hosted ThingWorx environment

There will be no need to upgrade any of the hosted ThingWorx environments. All of the platform's upgrades will be made by PTC.

8 Glossary

This section lists all definitions or terms unique to this document or computer operation and are subject to interpretation by the user of this document.

8.1 Glossary

Application Key - A string/token that permits the execution of a ThingWorx service without logging in. (The key is associated with a user's security profile).

Data Shape - Comprised of field definitions, it defines the column names and BaseTypes for a custom event, in memory InfoTable, DataTable, Value Stream, or Stream.

Data Table - A storage table that has a primary key and optional indexed fields. Particularly suited for lookup and reference table use.

Extensions - Additional libraries, templates, connectors and widgets that can be added to ThingWorx via import.

User Group - A collection of users.

InfoTable - A zero-indexed, ordered array of JavaScript objects that expose the same properties. The data structures can be used to insert and extract information from Data Tables, Streams, and ThingWorx APIs for configuring bindings, security, and tagging.

Organization - Hierarchical unit that can be used to assign visibility on entities/objects.

Property - Represents a property/behavior of the actual thing/process that you are modeling.

Subscription - The action that executes when an event occurs.

Tag – A way for ThingWorx to identify (label) different elements (such as ThingShapes or Things) as being part of the same project, enabling you to transfer projects across different servers

ThingShape – Building blocks that can either build a template and then build a Thing, or directly help you to build a Thing

Thing Template – A way to combine shapes (multiple sources of data) that enables you to add additional property services, subscriptions and events to create a unique template that permits for the rapid creation of any new thing

Thing – Something that takes in data and uses it to fire events, process services and so on within the rest of the system based. An in-memory representation of the asset, system or cloud service you are trying to model

Entity – A method for ThingWorx to store data that can later be used in a mashup

Service – Something that directs data, and instructs it on where to go and what to do (such as display this data on this widget). Can also be used to send commands to SCP, for example to light up an LED

Event – Something that triggers a service (such as clicking)

Mashup – An application that uses content from more than one source to create a single new service displayed in a single GUI

ThingWorxStorage - Directory where all storage directories are created/located (excluding Backup storage). It stores all active information (data, logs, extensions and so on)

ThingWorxBackupStorage - Directory where backup storage directory is created/located. It stores all backup information

Streams - Streams represent time series data. Therefore, each stream has a timestamp plus additional fields. A ThingWorx stream is a list of activities from Things or data associated with Things. A stream can be thought of as a table structure with five predefined fields and any number of user-defined fields

Value Streams - Stores data from an associated Thing’s properties.