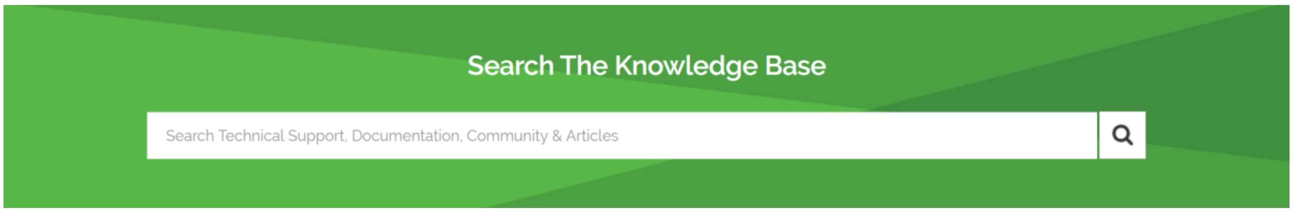
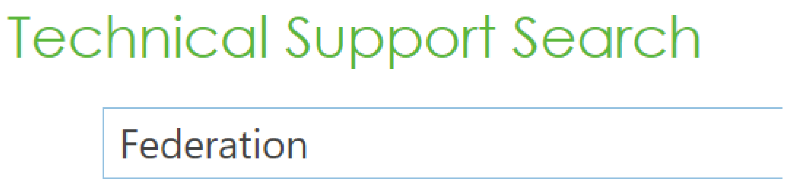
What is Federation?

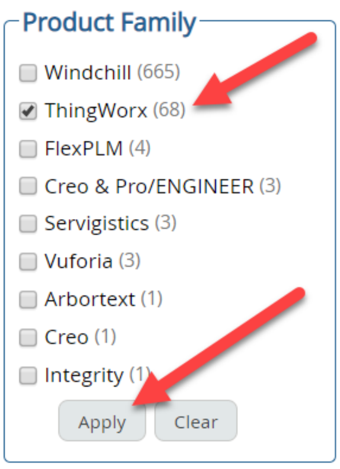
Federating ThingWorx instances means to link two or more ThingWorx instances in a way that some instances can publish their entities to a set of other subscriber instances.

Your first stop shop is the PTC Technical Support Portal: there you can perform full text searches of the topic you are interested in.

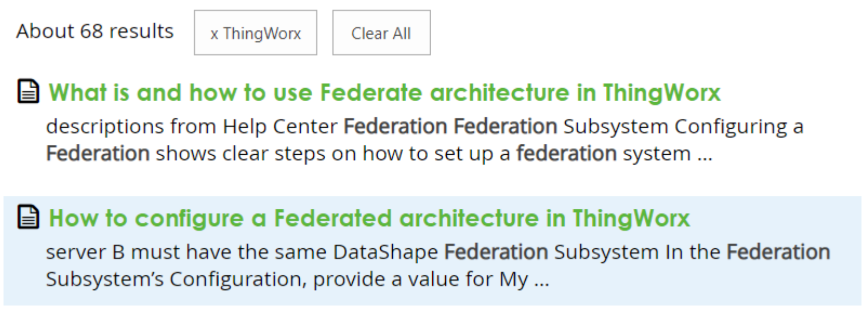

Search The Knowledge Base
Search Technical Support, Documentation, Community & Articles

You type `Federation` and then filter for `ThingWorx`.





Then browse through the results:



About 68 results | x ThingWorx | Clear All

What is and how to use Federate architecture in ThingWorx
descriptions from Help Center **Federation Federation** Subsystem Configuring a **Federation** shows clear steps on how to set up a **federation** system ...

How to configure a Federated architecture in ThingWorx
server B must have the same DataShape **Federation** Subsystem In the **Federation** Subsystem's Configuration, provide a value for My ...

The first two results above are PTC Technical Support articles:

- CS248695
  [What is and how to use Federate architecture in ThingWorx]
- CS182335
  [How to configure a Federated architecture in ThingWorx]

The first article CS248695 states that a Federation enables sharing and remote controlling Data Things like Stream, Value Stream, Data Table, Wiki and Blog between servers. **Actually Federation does not limit itself to Data Things**: it can be used with almost any entity, including the most basic things implementing the `GenericThing` template. Article CS248695 then goes on to link three official documentation topics from the ThingWorx Help Center:

- Topic 1 - **What is Federation**: this is a *concept* topic that describes how to offload workloads across ThingWorx instances. The most important sentence in my opinion is the following: "*In ThingWorx, a federation* ***can be used to obtain other Thing's services and properties****. Any published Thing can be accessed as a Remote Thing from a federated server*". This means that things can be published by one ThingWorx instance so that their properties and services can be exposed to other (federated) ThingWorx instances that subscribe to the publisher.
- Topics 2 - **What is the Federation Subsystem**: this is a *reference* topic, describing the configuration options available in the Federation Subsystem. Not very useful if we are just learning the concept, but relevant later on when performing the actual configuration.
- Topic 3 - **How to configure Federation**: this is a *procedural* topic, with the goal to explain the steps required to federate

ThingWorx instances. It describes the procedure in terms of Server A and Server B, or Local Server and Remote Server. In this article, however, I use a different terminology: I talk about a Publisher instance and a Subscriber instance; the concept can then be generalized in terms of multiple publishers and subscribers.

The second article, CS182335, explains the same concepts as in topic 3 above, but in this article I try to describe Federation by providing a different conceptual framework, which is easier to grasp for me, and I hope for you.

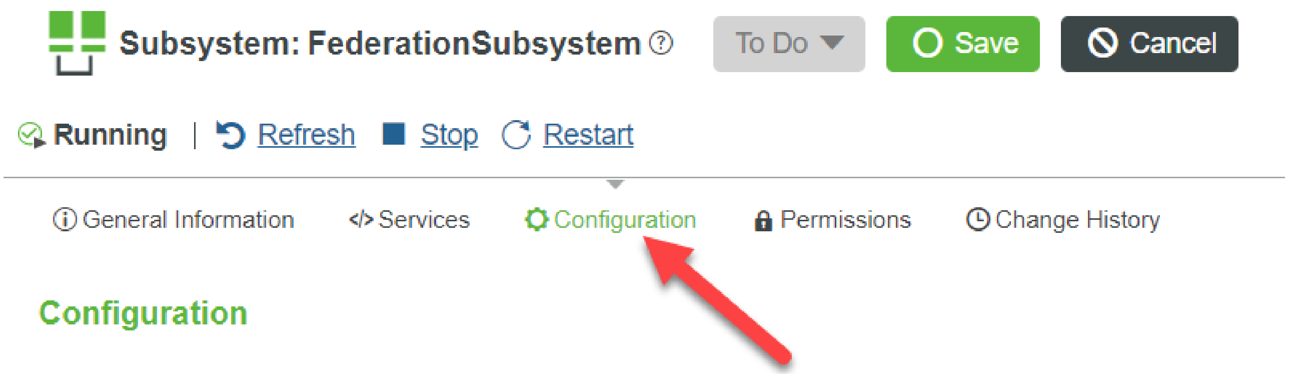
Publisher instance

As already stated in the introduction, a ThingWorx Federation is a set of ThingWorx instances: some them publish entities and others subscribe to them. The same ThingWorx instance can both publish some entities and subscribe to other instances.

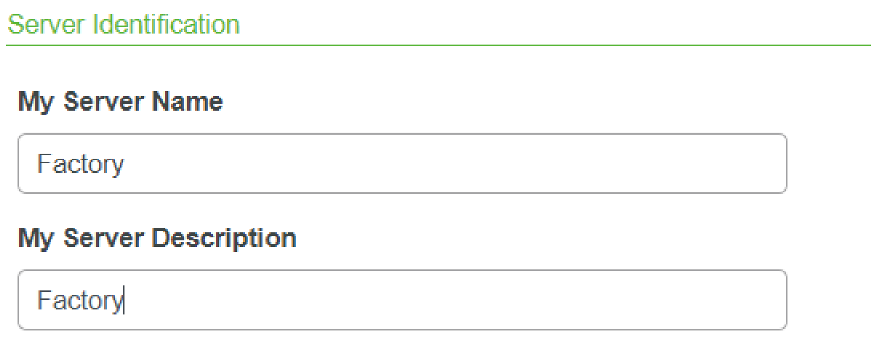
Let's start from Publisher instances.

If an instance wants to publish entities, it must be identifiable by subscribers, that is, it must have a name (called `Server Name`).

This is defined in the Federation Subsystem configuration page:



Here you will find `My Server Name` and `My Server Description`:


Server Identification

My Server Name

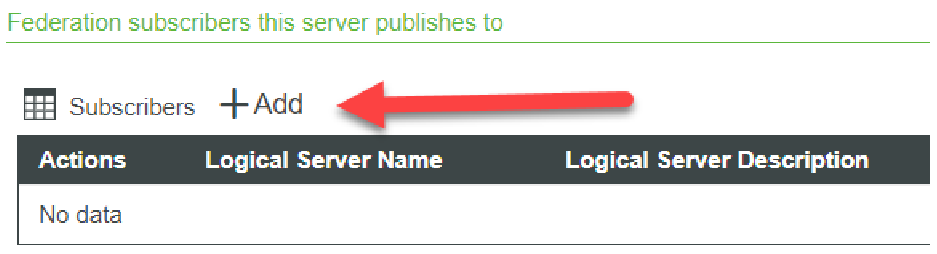
Factory

My Server Description

Factory

`My Server Name` is the name that subscribers will use to access the entities that are published by the publisher instance.

For security reasons ThingWorx restricts the subscribers that can participate in the Federation: white listed instances are called `Subscribers` in the screenshot below (taken in the context of the Federation Subsystem of a provider instance).


Federation subscribers this server publishes to

Actions	Logical Server Name	Logical Server Description
No data		

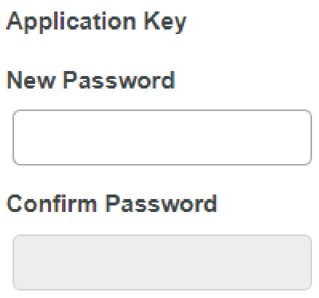
If you click on `Add`, you can list the subscribers to this publisher instance.

The subscriber instances in this list are identified by their `Server URI` (see below) and not by the `Server Name`, which in the context of this subscriber list can be different from the one used in the Federation Subsystem of the subscribers themselves.

Here are the relevant properties that describe a subscriber:

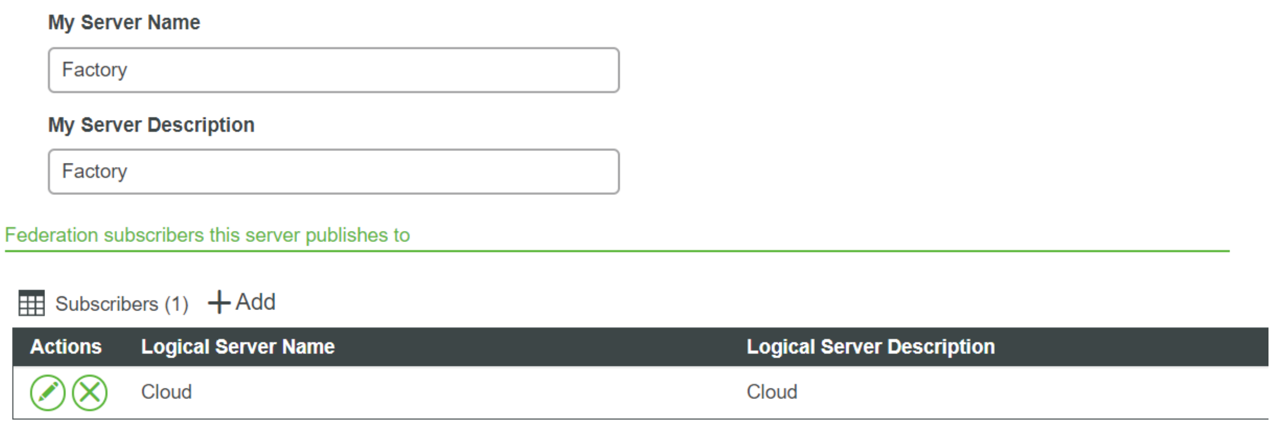
- **Logical Server Name**: The name given to this subscriber (can be different from the `My Server Name` used in the Federation Subsystem configuration of the subscriber instance)

- **Logical Server Description**: The description for the subscriber instance, seen in the context of the publisher's subscriber list (can be different from the description used in the subscriber instance Federation Subsystem configuration)
- **Enable Publishing**: Whether this publisher instance can publish to this subscriber
- **Server URI**: The WebSocket URI of the subscriber. This is usually a `wss://` URI if the subscriber has HTTPS enabled or a `ws://` URI otherwise. For example: `wss://subscriber-ip:subscriber-port/Thingworx/WS`
- **Application Key**: In order for the publisher to access the subscriber instance and publish entities to it, the Subscriber must create an Application Key that Publisher must use to authenticate. In the `New Password` and `Confirm Password` fields we must paste such application key (generated in the Subscriber instance):



- **Publish as User**: This is the user account in the publisher instance that actually performs the operation of publishing data to the subscriber instance

The final configuration on the publisher instance would then be:


My Server Name

Factory

My Server Description

Factory

Federation subscribers this server publishes to

Subscribers (1) + Add

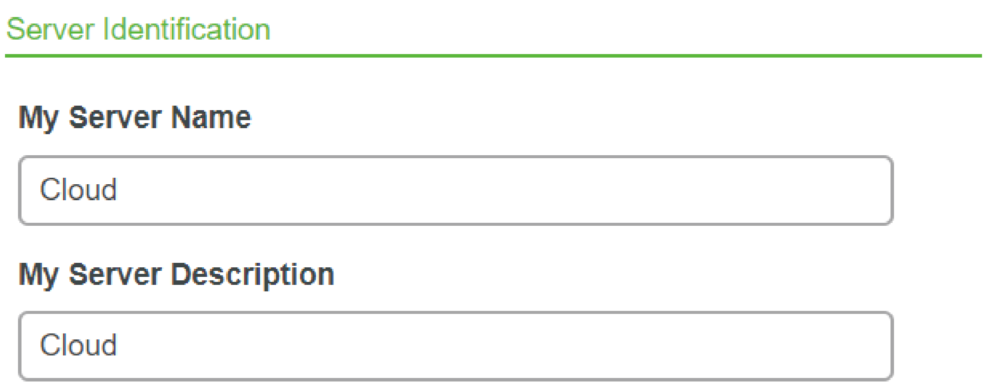
Actions	Logical Server Name	Logical Server Description
	Cloud	Cloud

In this case I have two test instances:

- The Publisher instance is called `Factory`
- The Subscriber instance is called `Cloud`

Subscriber Instance

The subscriber instance must be configured as follows:


Server Identification

My Server Name

Cloud

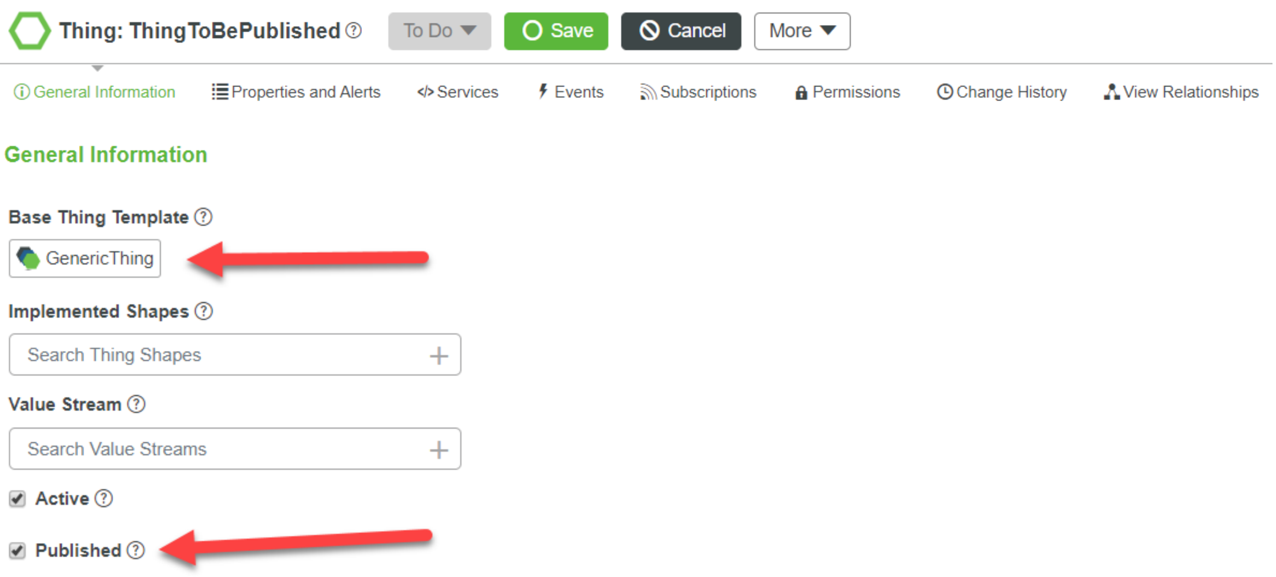
My Server Description

Cloud

Publisher Entities

In the publisher instance we can create entities and publish them.

Just create an entity implementing the `GenericThing` template and mark it as `Published`.



General Information

Base Thing Template: GenericThing

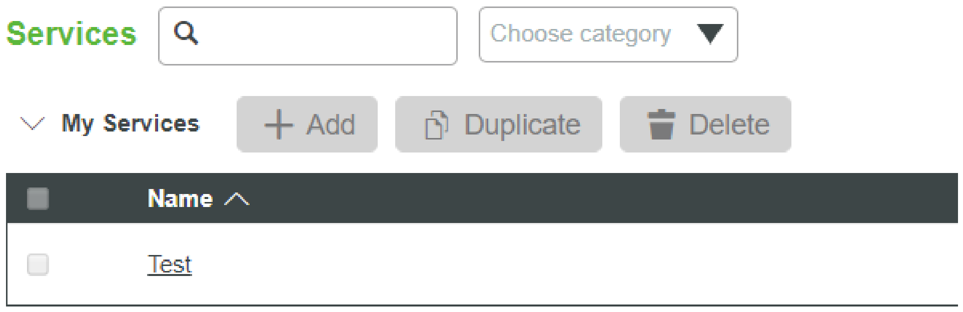
Implemented Shapes: Search Thing Shapes

Value Stream: Search Value Streams

Active ✓

Published ✓

Then, for example, implement a service called `Test`.



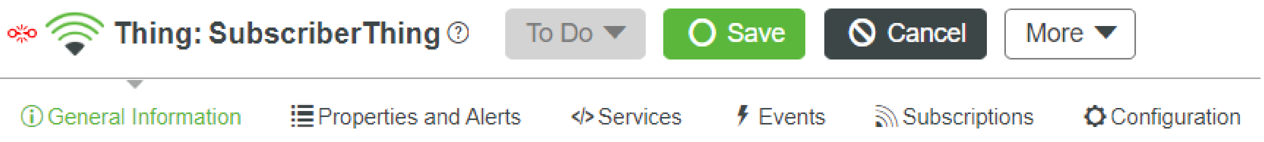
Name
Test

We will reference it from an entity in a subscriber instance: such referencing action is called `binding`.

Subscriber Entities

Now move to the subscriber instance and create an entity to bind that service.

Create an entity implementing the `RemoteThing` template and save it.

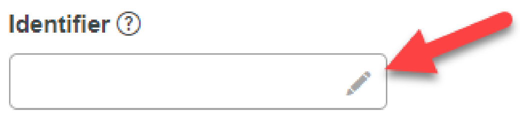


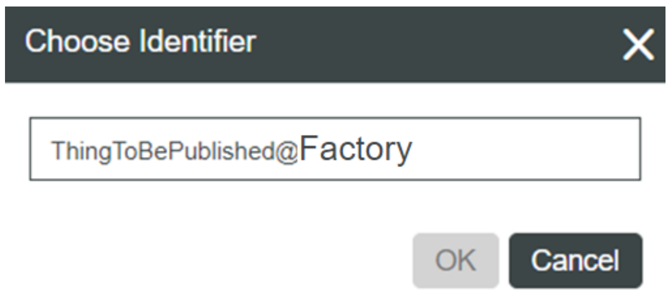
As you save it you will notice that the entity is created disconnected.

This icon means "Disconnected".

While this icon identifies a Remote Thing.

Now we connect it: while in edit mode select the pencil icon in the `Identifier` field:



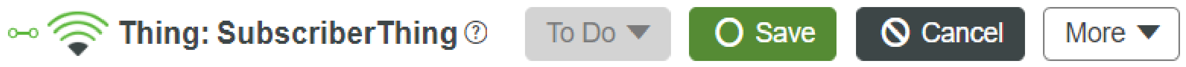


ThingToBePublished@Factory

If everything was configured correctly, you should get a list of (unbound) entities from all ThingWorx instances that are publishing entities to this subscriber.

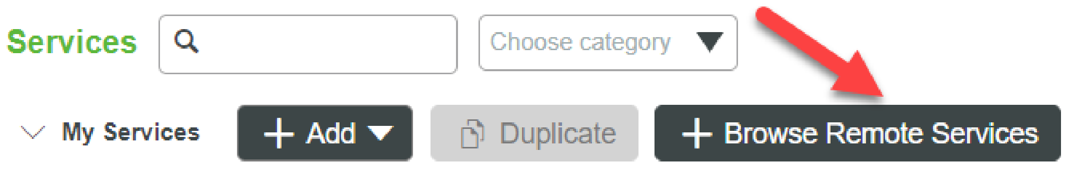
You select one of the items in the list and click `OK`, then save the entity.

Upon save, the remote thing in the subscriber instance gets connected:

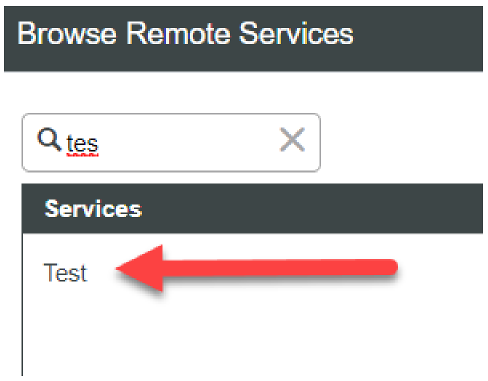


At this point all properties and services defined in the publisher instance's entity that have been bound are available to the subscriber entity.

You can prove that the remote service is available by creating a new local service in the subscriber entity and select `Browse Remote Services`:



There you will find a service called `Test`, the one that we created before on the publisher instance.



You can drag and drop it on the right-hand side to define a service bound to the remote service in the publisher instance.

Conclusions

You have successfully federated two ThingWorx instances and bound a service defined on one to a service defined on the other in order to offload execution of the service from the subscriber instance to the publisher instance.

Congratulations!