Remote Monitoring Application Starter - Documentation

Author: Pai Chung

IoT Solutions Architect

Customer Success IoT

Remote Monitoring Application Starter - Manual

Description

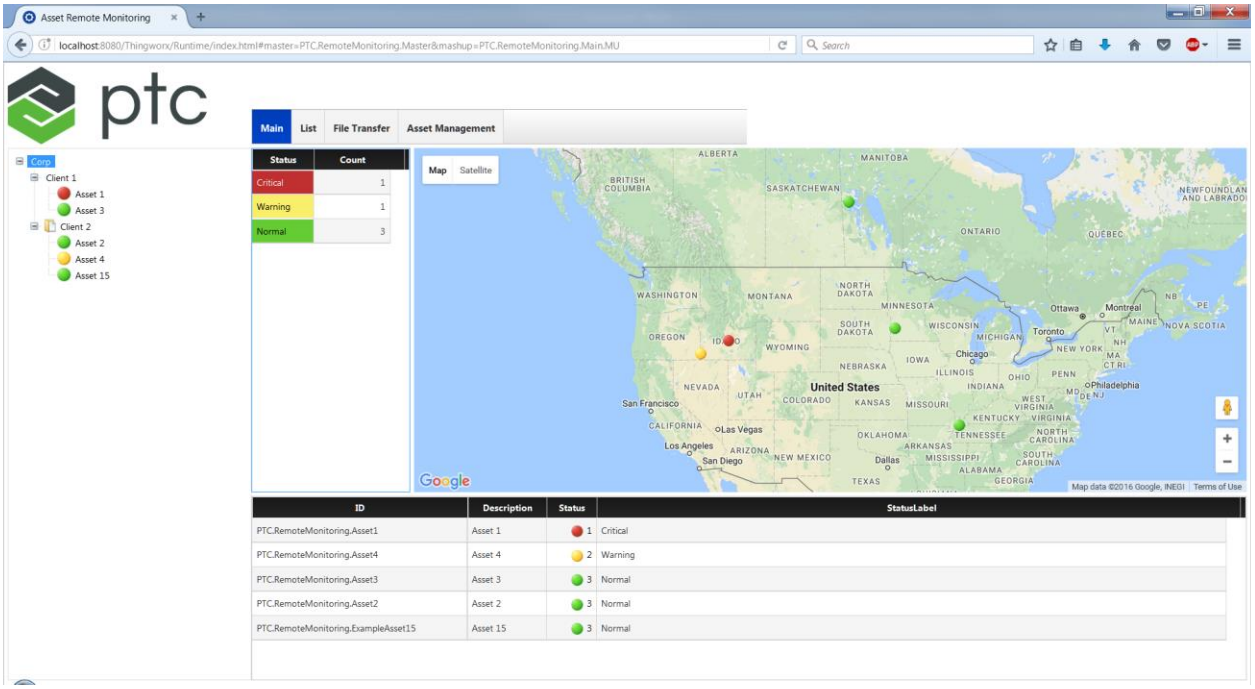
The Remote Monitoring Starter is a Starter Application created using Best Practices as well as many useful techniques.

Purpose

This Starter Application can literally get you started, but can also be a useful teaching tool

Application Runtime Walkthrough

Main screen / Asset Summary



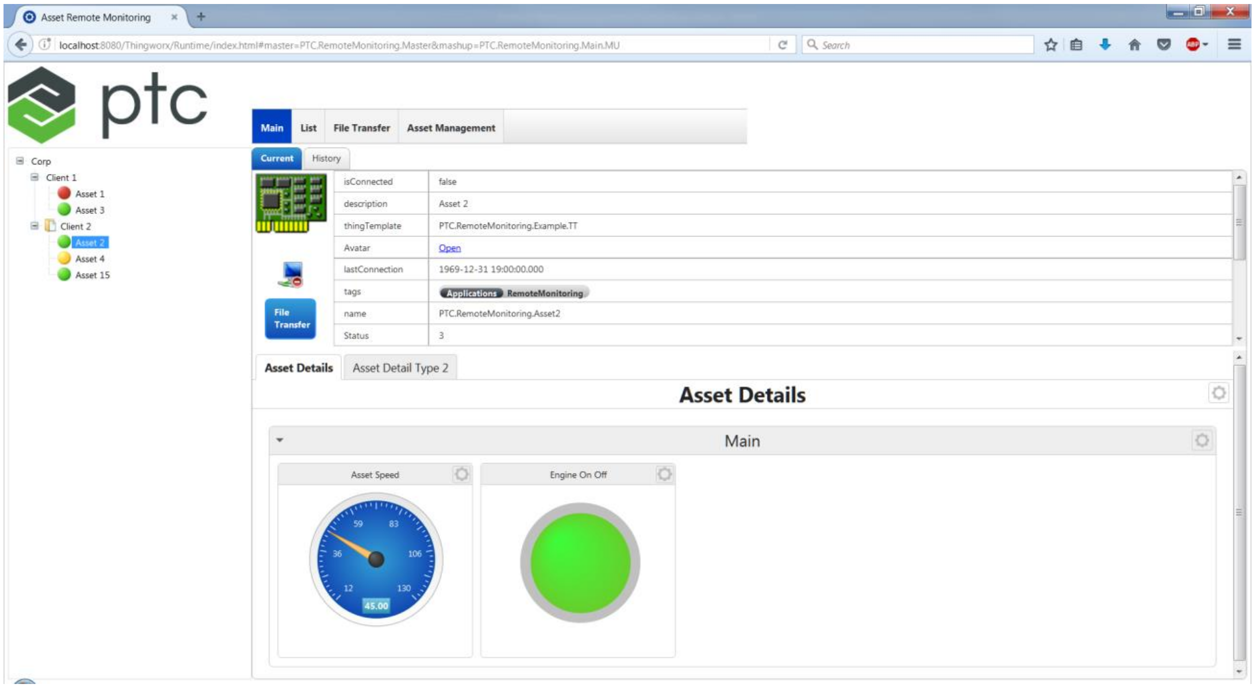
Status	Count
Critical	1
Warning	1
Normal	3

ID	Description	Status	StatusLabel
PTC.RemoteMonitoring.Asset1	Asset 1	1	Critical
PTC.RemoteMonitoring.Asset4	Asset 4	2	Warning
PTC.RemoteMonitoring.Asset3	Asset 3	3	Normal
PTC.RemoteMonitoring.Asset2	Asset 2	3	Normal
PTC.RemoteMonitoring.ExampleAsset15	Asset 15	3	Normal

- This mashup has a Master assigned in the Title property of the master we put “Asset Remote Monitoring” which now shows in the tab at the top.
- Logo is inserted in a left side bar with scaling to Height
- A menu is assigned which is secured with different Groups so that depending on who logs in, different options show up.
- A tree widget provides Navigation through all Asset levels.
    - Tree data includes a Status field to drive state based definitions
    - Tree data includes a Mashup name field to drive the actual mashup shown next to the tree
    - Selected Node data is passed in from Data Source Selected row to both the Mashup name of the contained mashup and to “Entity” which is a Mashup parameter
- Small Grid shows aggregate roll up of the assets that are assets under the selected node. Map shows assets under the selected node. Detail grid shows individual line items for all assets under the selected node. (ie. If client 1 is selected only two nodes will show and be aggregated)

- Map and Grid information come from the same service and a double click will navigate to the detail mashup using the ‘double clicked’ event, data source selected row and a Nav Widget. Again Mashup Name and Entity are passed in.

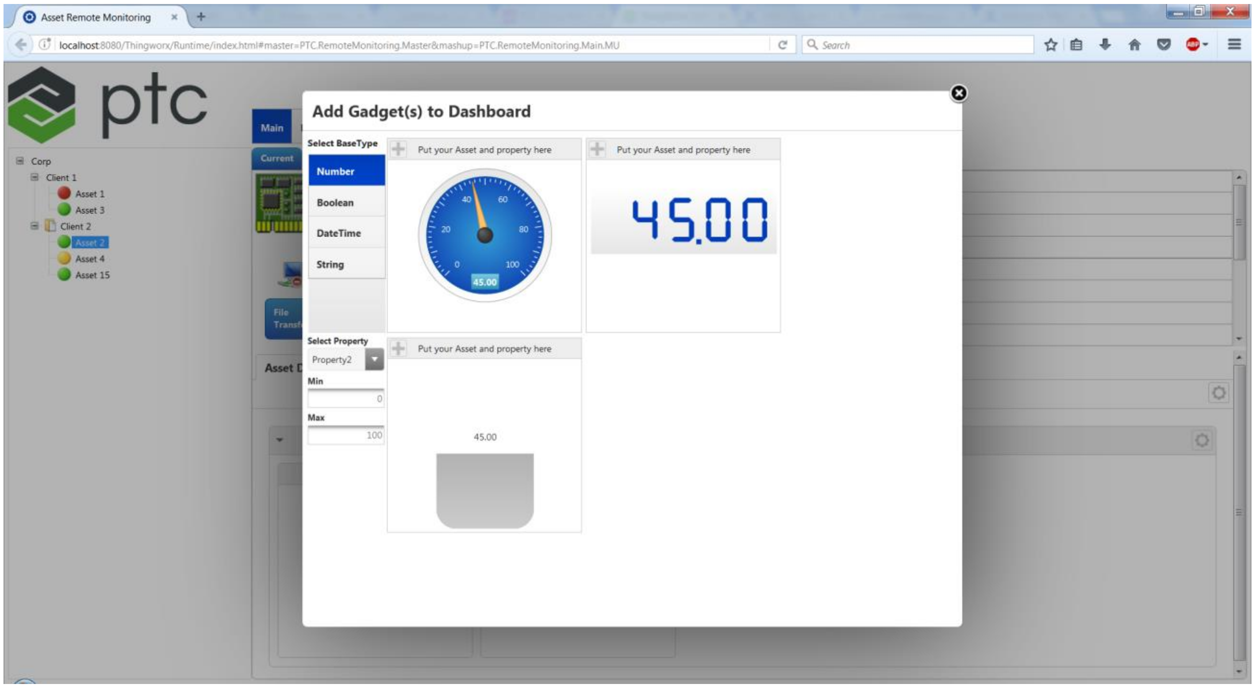
Asset Detail



isConnected	false
description	Asset 2
thingTemplate	PTC.RemoteMonitoring.Example.TT
Avatar	Open
lastConnection	1969-12-31 19:00:00.000
tags	Applications RemoteMonitoring
name	PTC.RemoteMonitoring.Asset2
Status	3

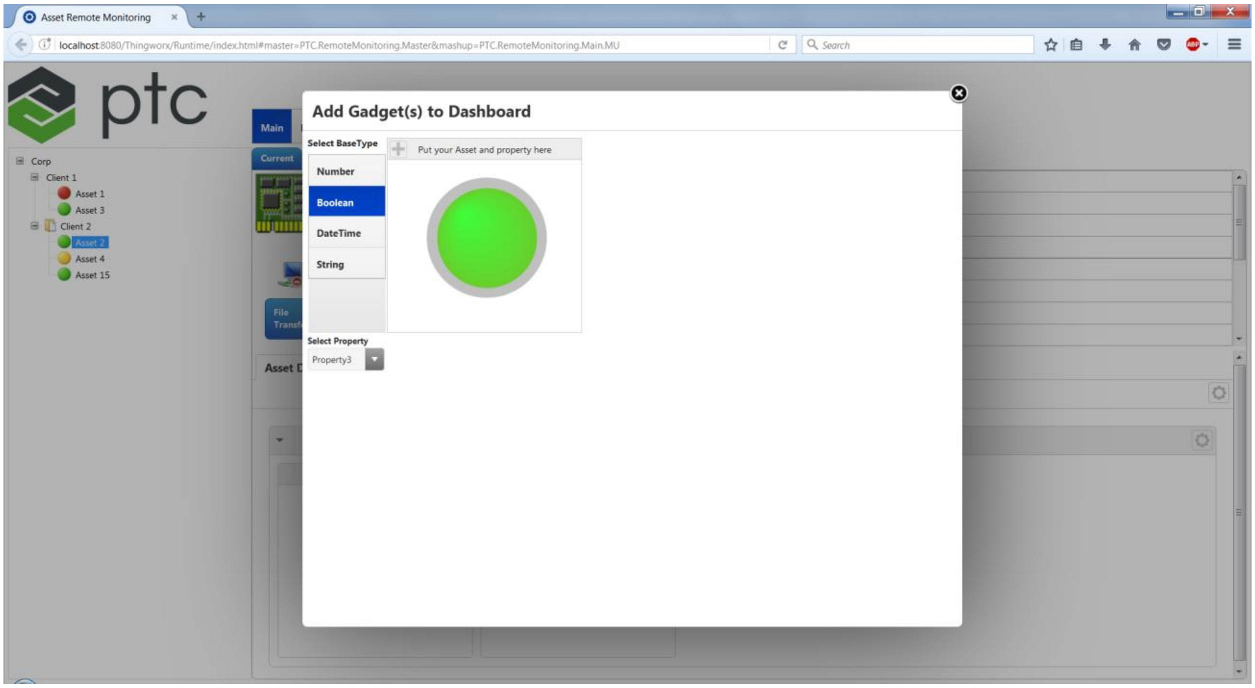
- Asset details are shown generically using a Property Display
- Asset avatar is showing
- Asset Remote Access is available
- Use File transfer to go to File Transfer for this Asset or any other asset
- An Asset Type Customizable Dashboard is available to bring in Specific Properties
    - Each asset can have their own specific Dashboard (all dashboards will still show in tabs) Configure this in Asset Management
    - Dashboards will dynamically know which Asset’s information needs to be loaded in Gadgets

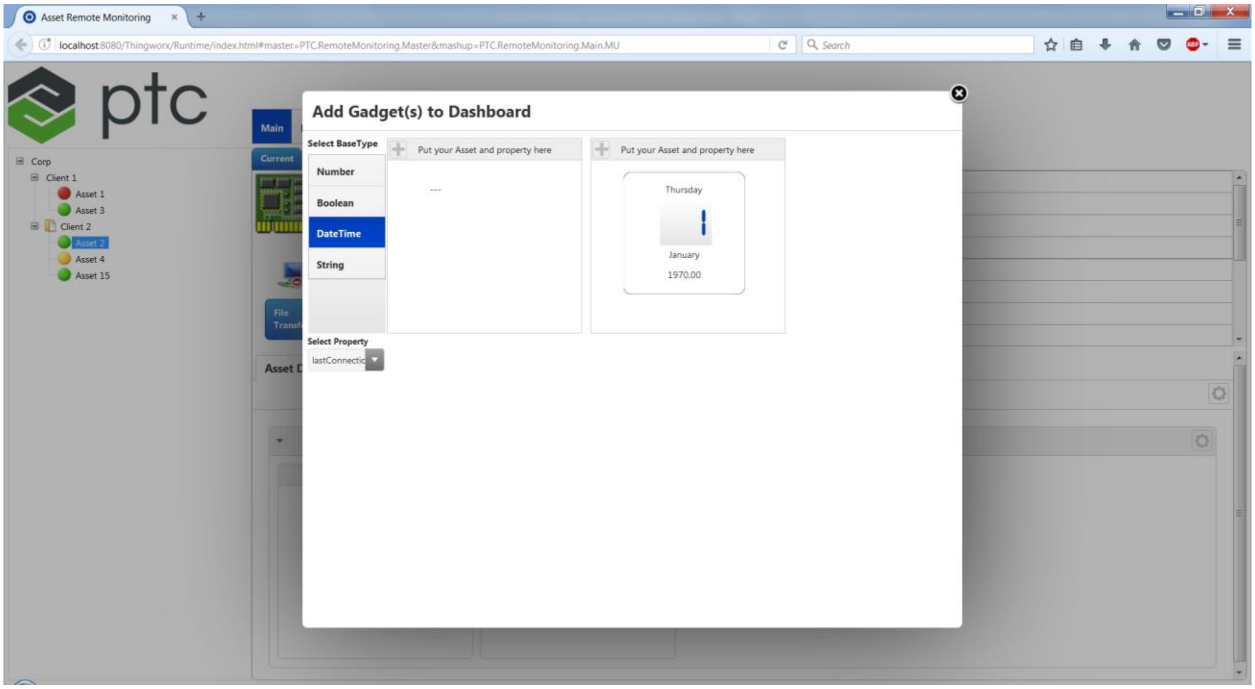
Gadget Library

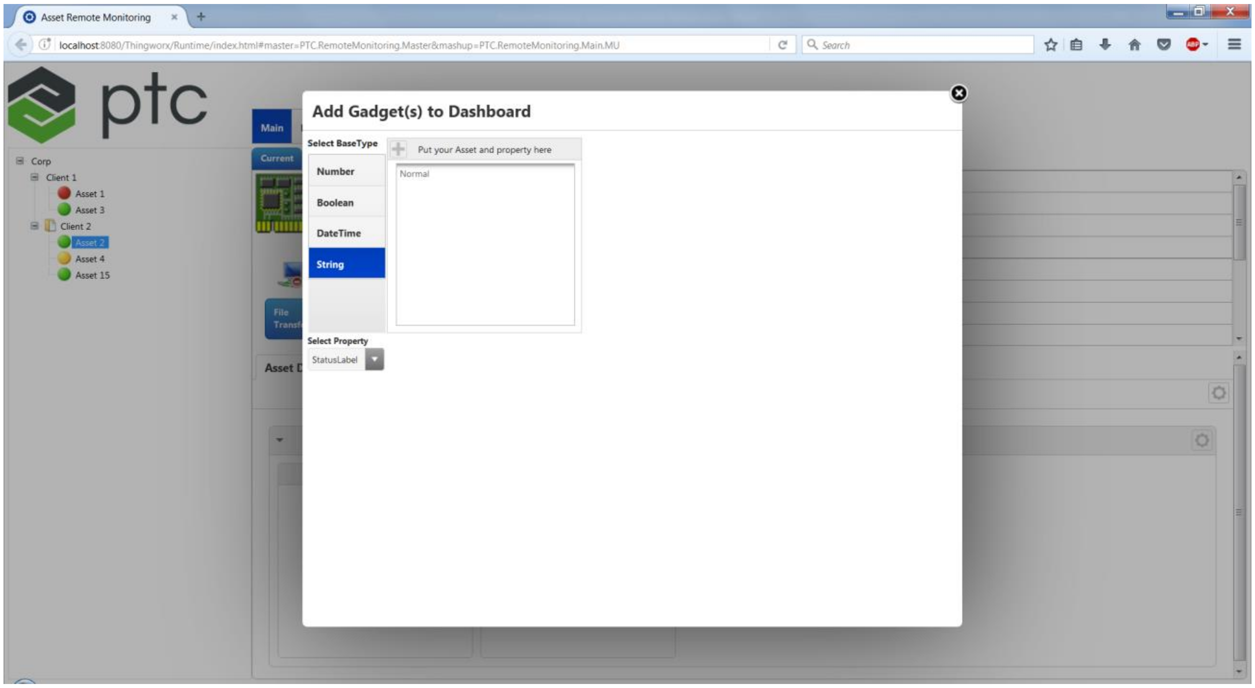

Asset Remote Monitoring
localhost:8080/Thingworx/Runtime/index.html#master=PTC.RemoteMonitoring.Master&mashup=PTC.RemoteMonitoring.Main.MU
Search
ptc
Main
Current
Corp
Client 1
Asset 1
Asset 3
Client 2
Asset 2
Asset 4
Asset 15
File Transf
Asset D
Add Gadget(s) to Dashboard
Select BaseType
Number
Boolean
DateTime
String
Select Property
Property2
Min
0
Max
100
Put your Asset and property here
Put your Asset and property here
Put your Asset and property here
45.00
45.00
45.00

- Gadget Library allows a User to pick Property BaseType and then a Property of that BaseType for selected Asset
- If the Property BaseType is a number, they can specify a min and max which is applied where applicable.
- After adding the Gadget it will be available on the Asset Dashboard for all assets that use that dashboard.

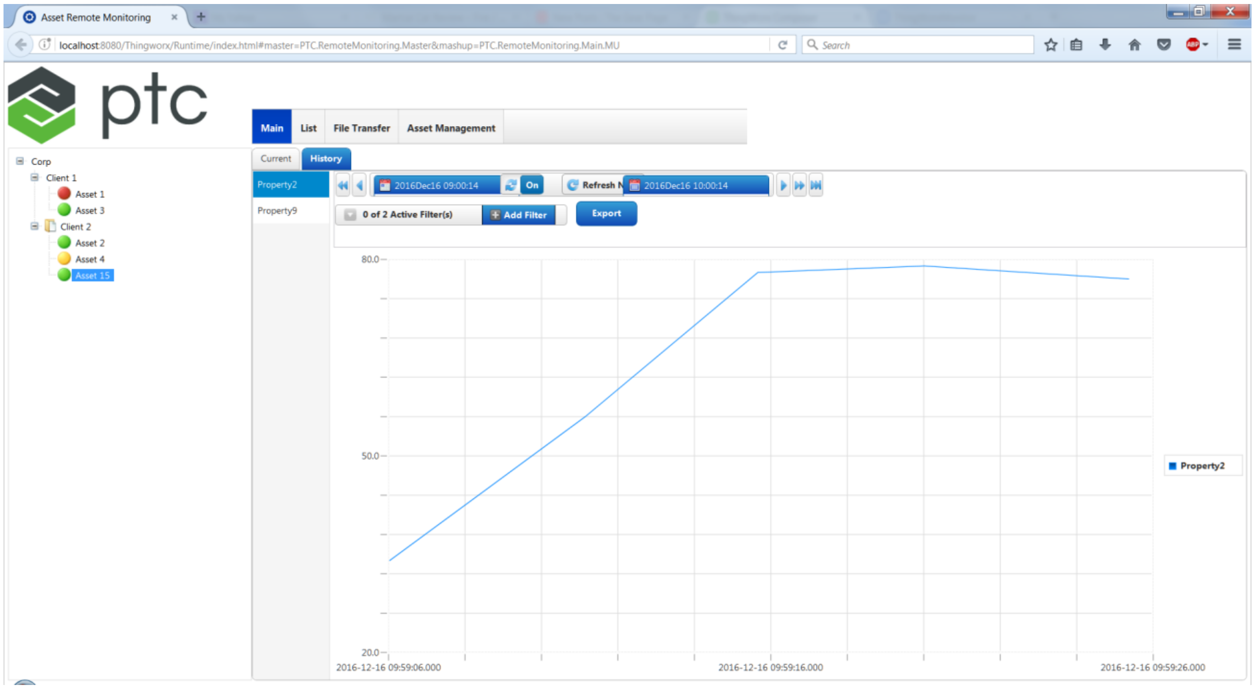
Additional Gadget Library screenshots





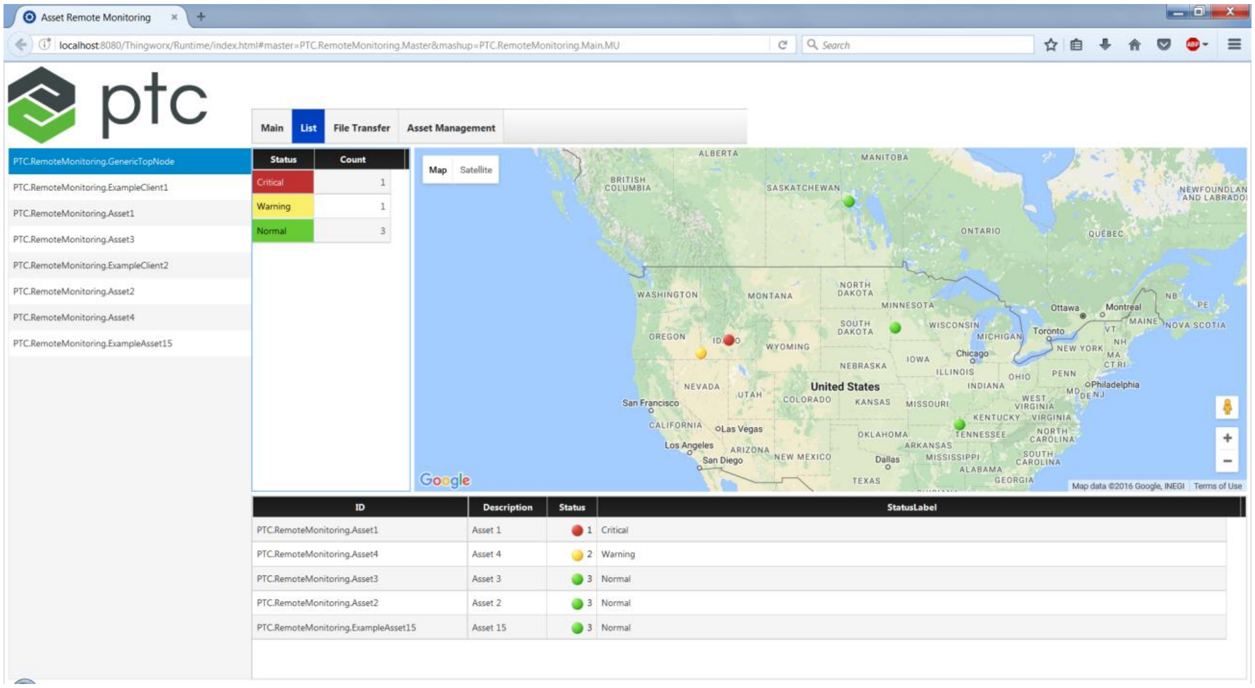


Asset Detail History



- History Tab shows all Numerical properties that are LOGGED
- You can Multi select the Properties and they will be displayed on the TimeSeries Chart

Alternate Main View

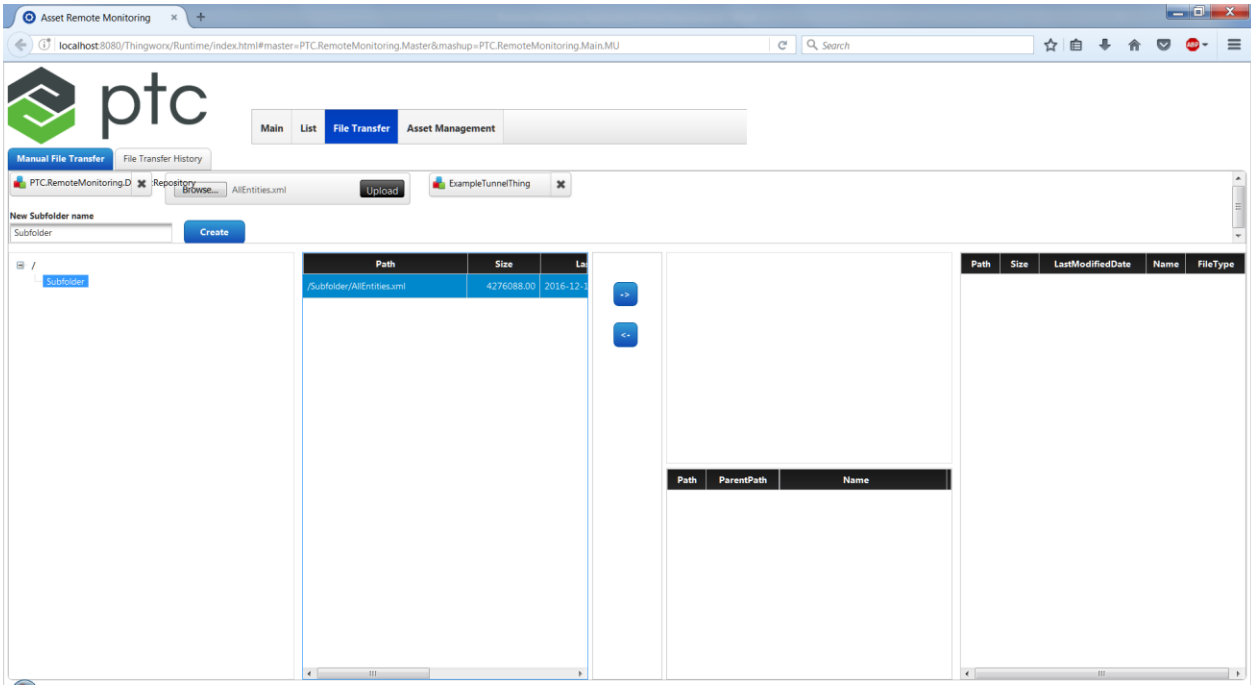

Asset Remote Monitoring
localhost:8080/Thingworx/Runtime/index.html#master=PTC.RemoteMonitoring.Master&mashup=PTC.RemoteMonitoring.Main.MU
Search
ptc
Main
List
File Transfer
Asset Management
PTC.RemoteMonitoring.GenericTopNode
PTC.RemoteMonitoring.ExampleClient1
PTC.RemoteMonitoring.Asset1
PTC.RemoteMonitoring.Asset3
PTC.RemoteMonitoring.ExampleClient2
PTC.RemoteMonitoring.Asset2
PTC.RemoteMonitoring.Asset4
PTC.RemoteMonitoring.ExampleAsset15
Map
Satellite
Google
Map data ©2016 Google, INEGI
Terms of Use

Status	Count
Critical	1
Warning	1
Normal	3

ID	Description	Status	StatusLabel
PTC.RemoteMonitoring.Asset1	Asset 1	● 1	Critical
PTC.RemoteMonitoring.Asset4	Asset 4	● 2	Warning
PTC.RemoteMonitoring.Asset3	Asset 3	● 3	Normal
PTC.RemoteMonitoring.Asset2	Asset 2	● 3	Normal
PTC.RemoteMonitoring.ExampleAsset15	Asset 15	● 3	Normal

- List is just a different way to display the assets

File Transfer screen



Path	Size	La
/Subfolder/AllEntities.xml	4276088.00	2016-12-1

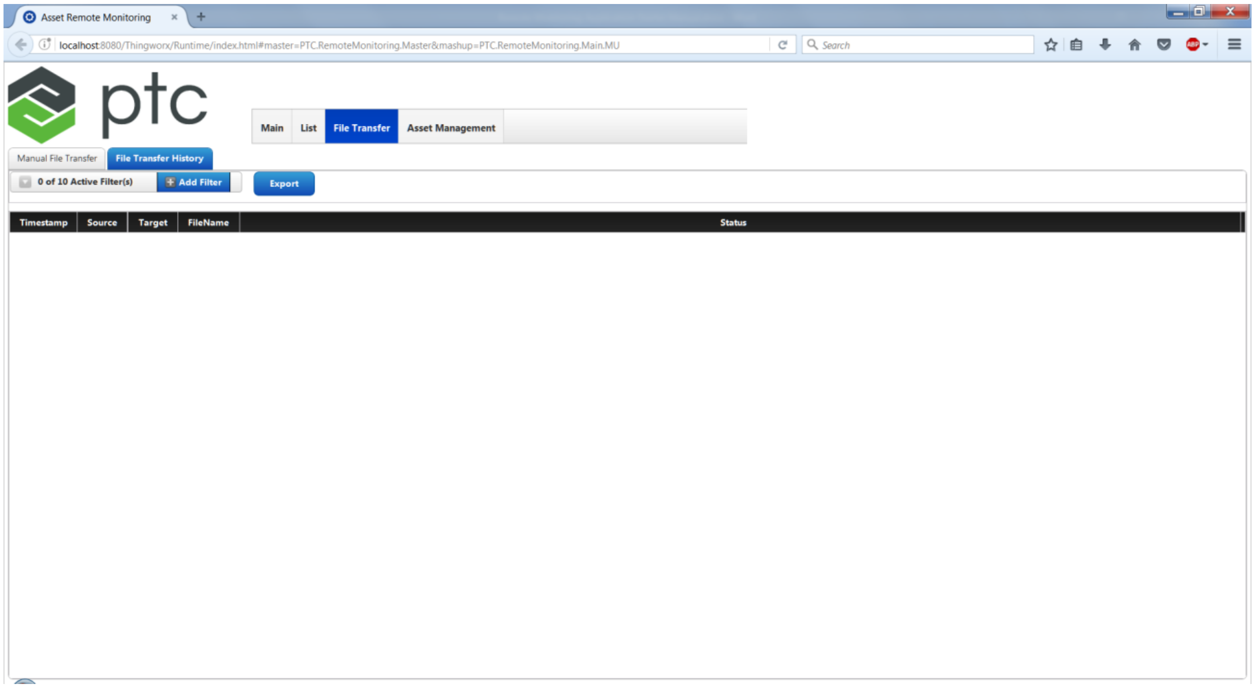
Path	ParentPath	Name

Path	Size	LastModifiedDate	Name	FileType

- File transfer is a ‘generic’ file transfer mashup
- Allows you to pick a File Repository on the platform
- Allows you to create subfolders

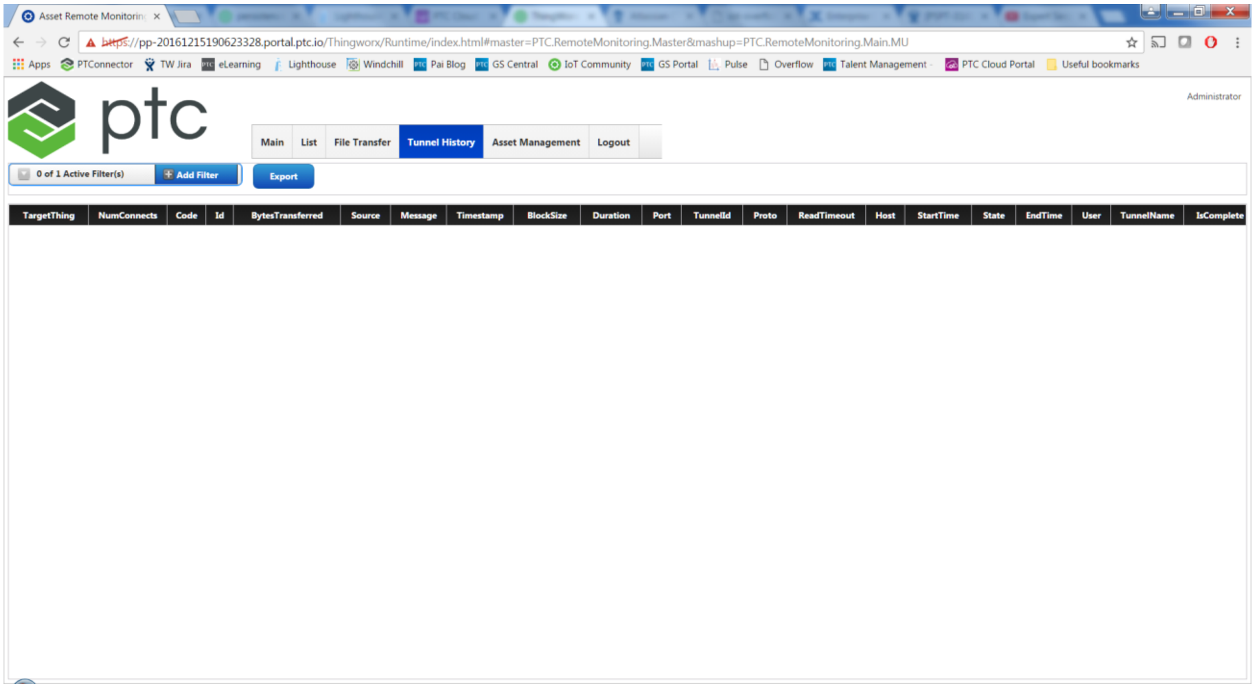
- Allows you to upload files
- Allows you to browse the folder structure and see files
- Allows you to pick an Agent that supports file transfer
- Allows you to browse the folder structure defined for the Agent
- Allows you to transfer files back and forth

File Transfer History screen



- File Transfer History is populated through the FileTransfer event into a Stream
- Grid allows sorting based on Column clicks
- Data Filter allows for filtering entries
- Export will export resulting data to csv client side

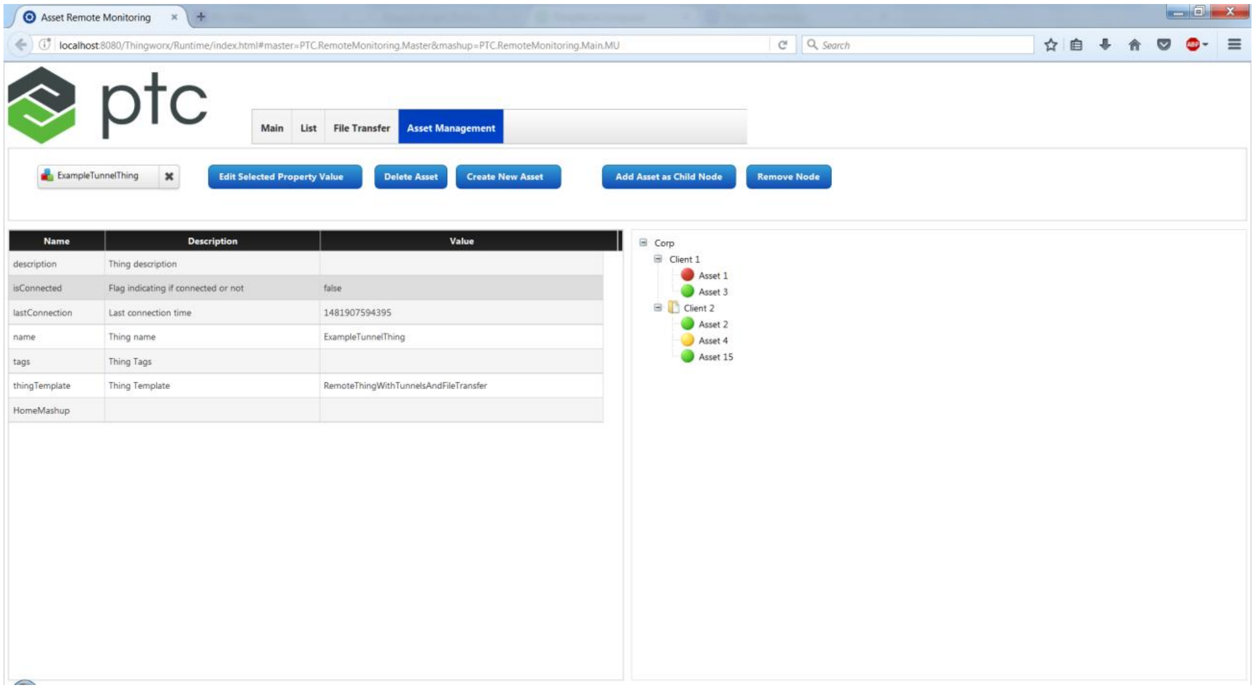
Tunnel History Screen


Asset Remote Monitoring
https://pp-20161215190623328.portal.ptc.io/Thingworx/Runtime/index.html#master=PTC.RemoteMonitoring.Master&mashup=PTC.RemoteMonitoring.Main.MU
Administrator
Main
List
File Transfer
Tunnel History
Asset Management
Logout
0 of 1 Active Filter(s)
Add Filter
Export

TargetThing	NumConnects	Code	Id	BytesTransferred	Source	Message	Timestamp	BlockSize	Duration	Port	TunnelId	Proto	ReadTimeout	Host	StartTime	State	EndTime	User	TunnelName	IsComplete

- Tunnel History is populated through the Tunnel events into a Stream
- Grid allows sorting based on Column clicks
- Data Filter allows for filtering entries
- Export will export resulting data to csv client side

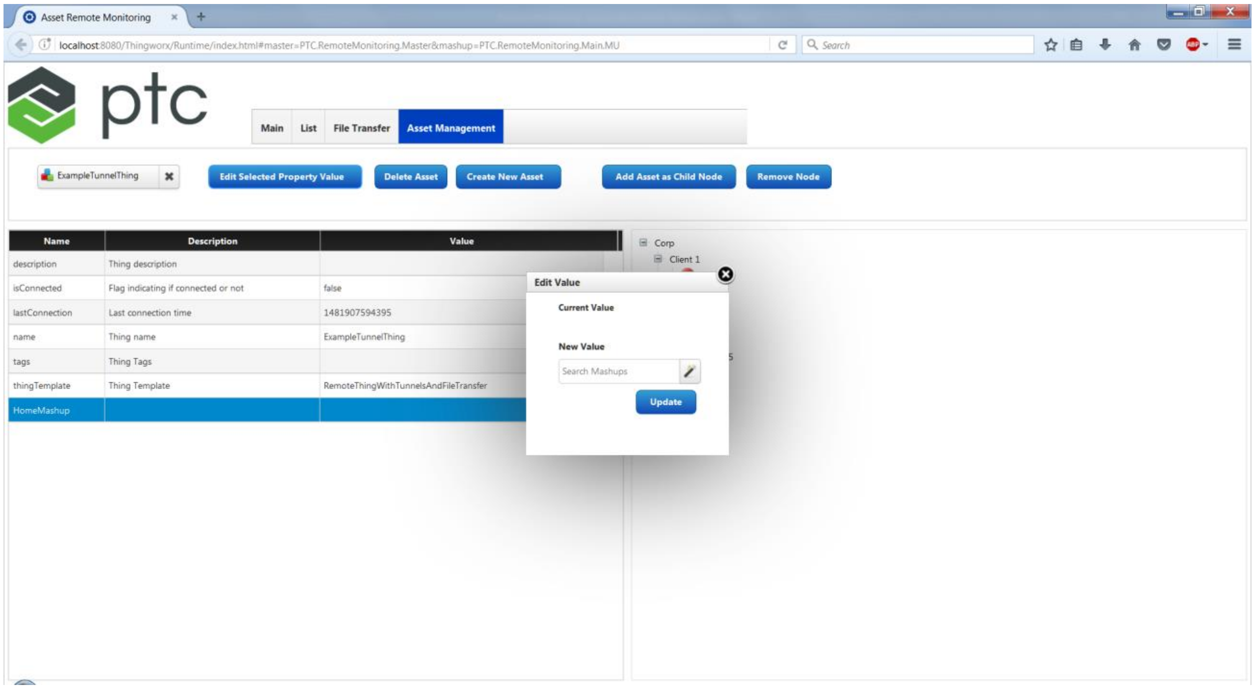
Asset Management screen



Name	Description	Value
description	Thing description	
isConnected	Flag indicating if connected or not	false
lastConnection	Last connection time	1481907594395
name	Thing name	ExampleTunnelThing
tags	Thing Tags	
thingTemplate	Thing Template	RemoteThingWithTunnelsAndFileTransfer
HomeMashup		

- Asset management is for Creating/Deleting and Configuring Assets
- Asset can be chosen
- Chosen Asset can be added/removed to Network
- Chosen Asset can be deleted

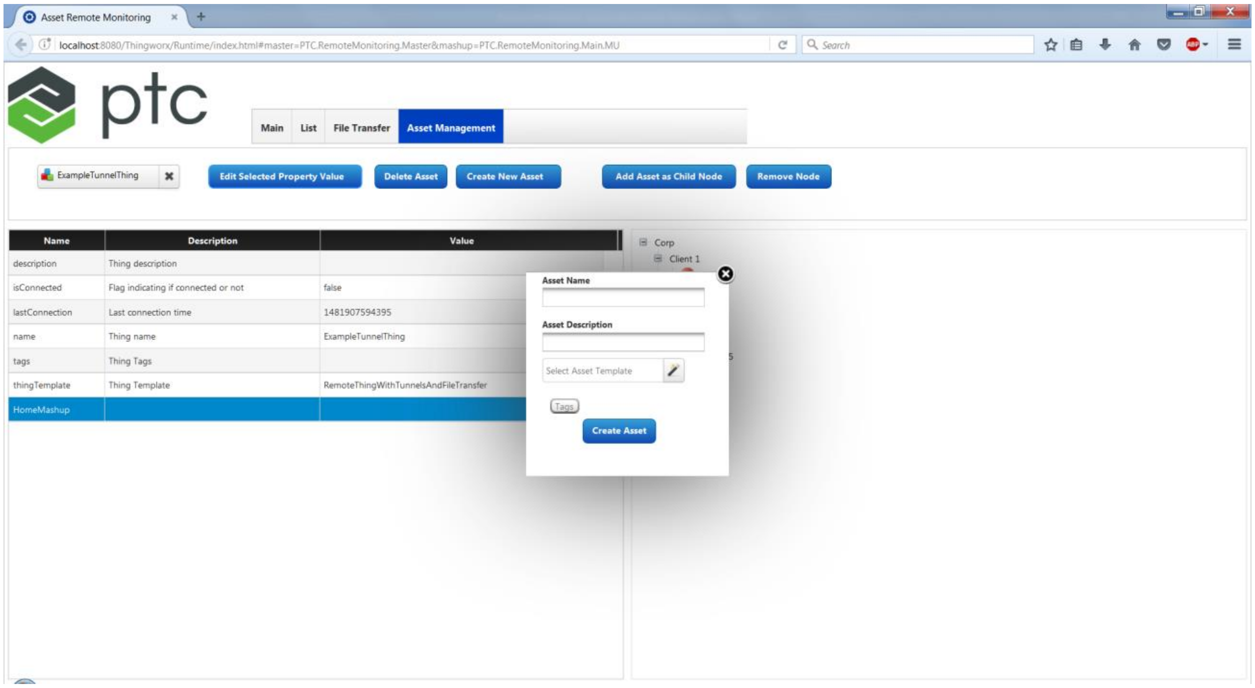
Universal Property Edit



Name	Description	Value
description	Thing description	
isConnected	Flag indicating if connected or not	false
lastConnection	Last connection time	1481907594395
name	Thing name	ExampleTunnelThing
tags	Thing Tags	
thingTemplate	Thing Template	RemoteThingWithTunnelsAndFileTransfer
HomeMashup		

- Any Property of any BaseType on the chosen Asset can be edited

Asset Creation



Name	Description	Value
description	Thing description	
isConnected	Flag indicating if connected or not	false
lastConnection	Last connection time	1481907594395
name	Thing name	ExampleTunnelThing
tags	Thing Tags	
thingTemplate	Thing Template	RemoteThingWithTunnelsAndFileTransfer
HomeMashup		

- New Asset can be created

Application Security Walkthrough

Security has been set up according to best practices.

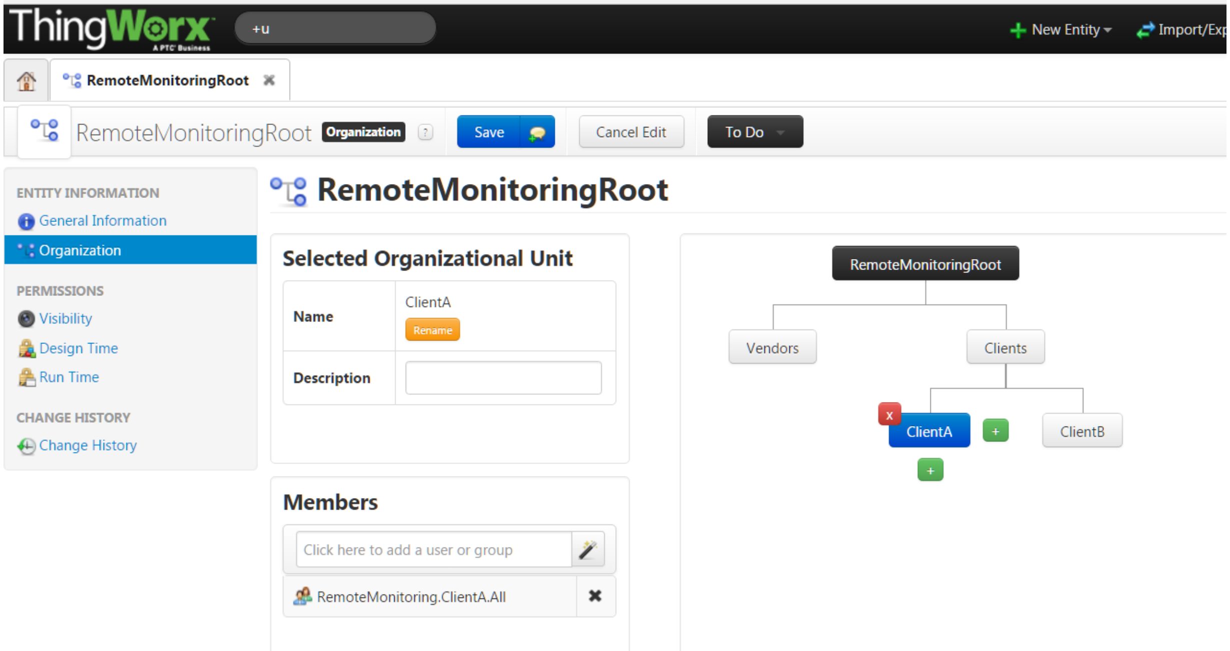
There are user groups for Visibility, user roles and Application Permissions. Users will be assigned to Role Groups, all role groups are their Visibility group and as needed, Role Groups are in Application Permission Groups. Also all OOTB services are secured with the System User.

Visibility

Visibility is what secures what a User can see and therefor interact with in the Model. This is split into:

- General Visibility – Everyone, every user see these. These are generally support application items etc.
- Specific Visibility – This client owns these assets and only this client can see them.

Following is the organization that is setup and note the Visibility Group assigned to Client A and Client B



RemoteMonitoringRoot

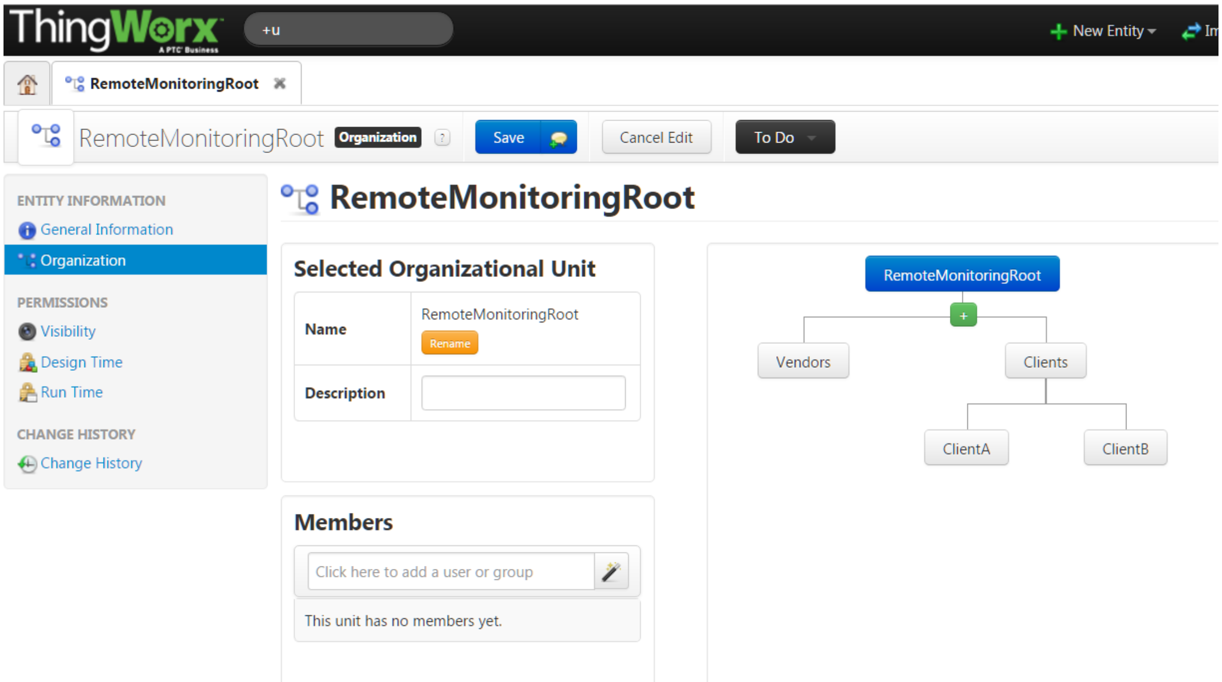
Selected Organizational Unit

Name	ClientA
Description	

Members

RemoteMonitoring.ClientA.All

Client A has RemoteMonitoringClientA.All assigned



RemoteMonitoringRoot

Selected Organizational Unit

Name	RemoteMonitoringRoot
Description	

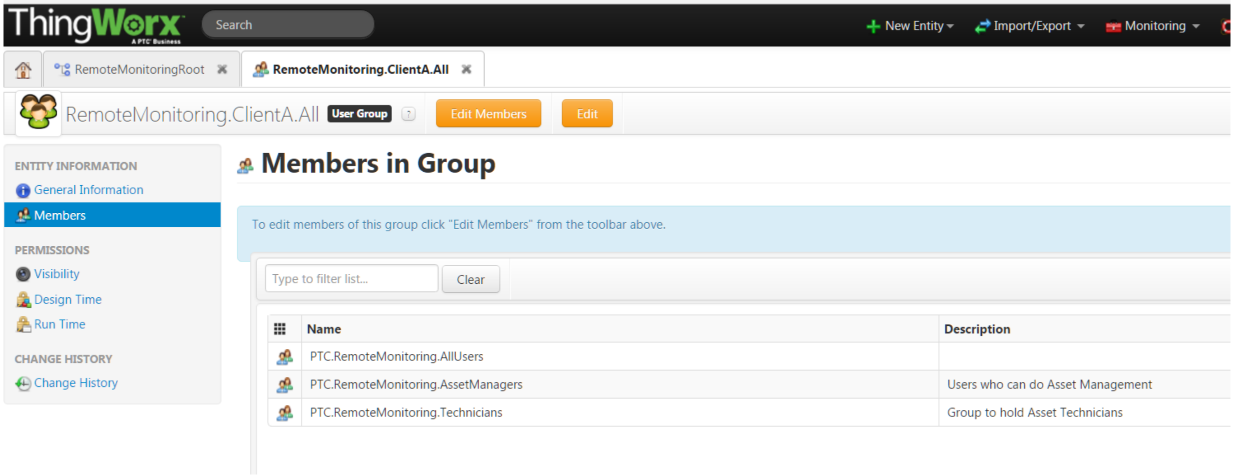
Members

This unit has no members yet.

Client B has RemoteMonitoringClientB.All assigned

Visibility Group

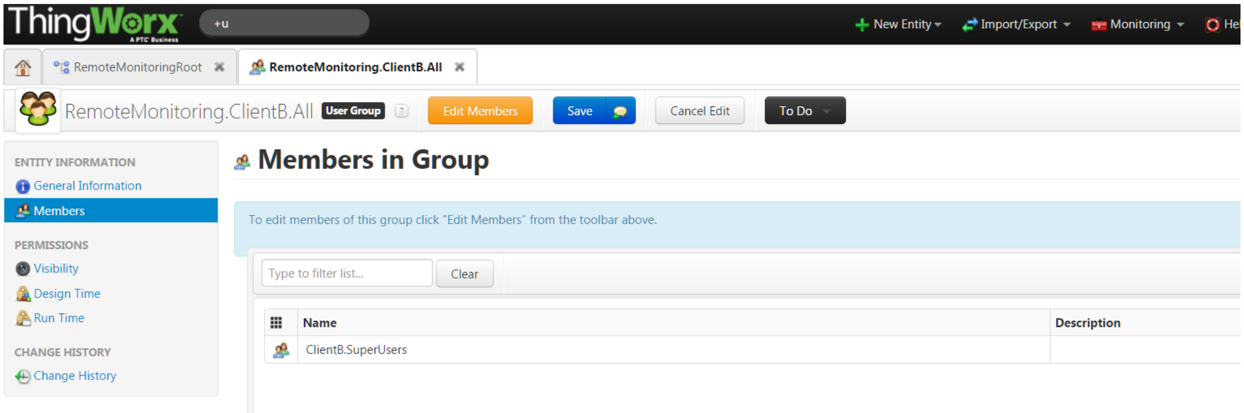
The Visibility Group contains all the User Role Groups



To edit members of this group click "Edit Members" from the toolbar above.

Name	Description
PTC.RemoteMonitoring.AllUsers	
PTC.RemoteMonitoring.AssetManagers	Users who can do Asset Management
PTC.RemoteMonitoring.Technicians	Group to hold Asset Technicians

Client A’s Role Groups in the aggregate visibility group (RemoteMonitoring.ClientA.All)



To edit members of this group click "Edit Members" from the toolbar above.

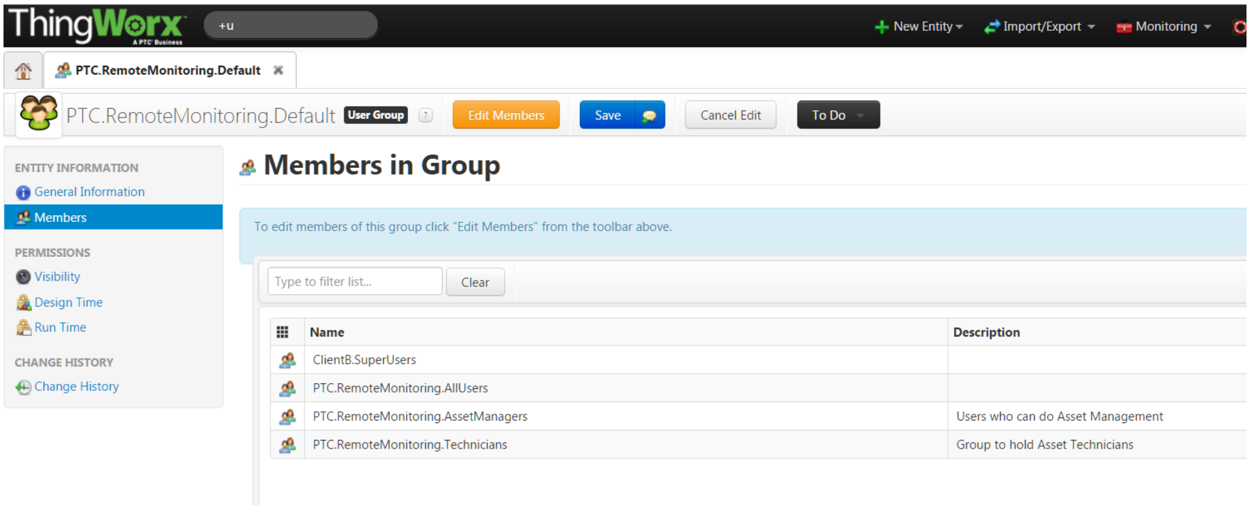
Name	Description
ClientB.SuperUsers	

Client B’s Role Groups in the aggregate visibility group (RemoteMonitoring.ClientB.All)

NOTE: Role Groups can be named according to the Client’s needs/standards.

Application Group

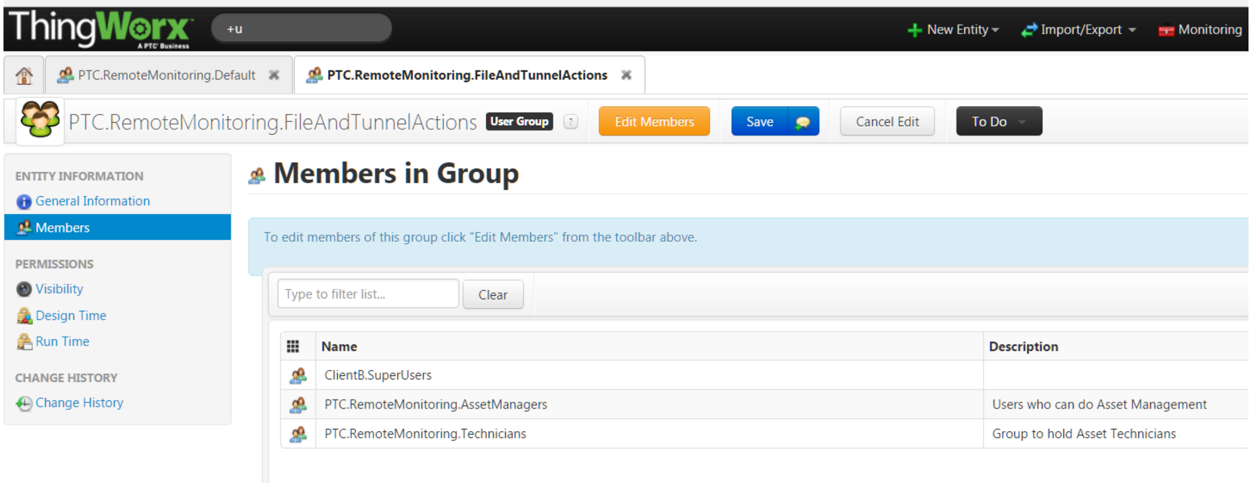
The Application Group contains the appropriate User Role Groups



Members in Group

To edit members of this group click "Edit Members" from the toolbar above.

Name	Description
ClientB.SuperUsers	
PTC.RemoteMonitoring.AllUsers	
PTC.RemoteMonitoring.AssetManagers	Users who can do Asset Management
PTC.RemoteMonitoring.Technicians	Group to hold Asset Technicians



Members in Group

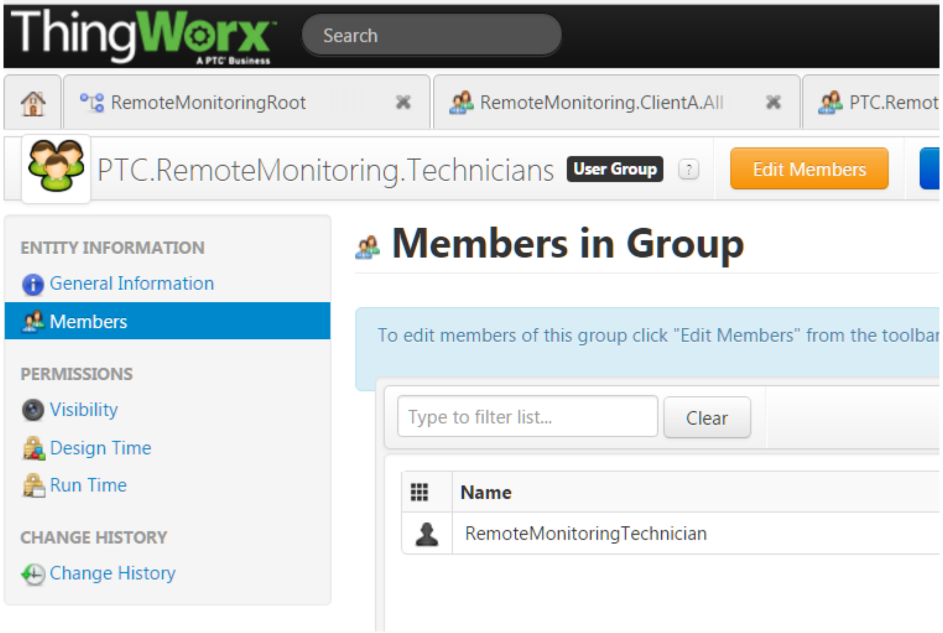
To edit members of this group click "Edit Members" from the toolbar above.

Name	Description
ClientB.SuperUsers	
PTC.RemoteMonitoring.AssetManagers	Users who can do Asset Management
PTC.RemoteMonitoring.Technicians	Group to hold Asset Technicians

NOTE: Role groups assigned across clients because these groups define what users can DO. Visibility permissions will define which entities they can apply these permissions on.

Users

Users are then in their Role Groups

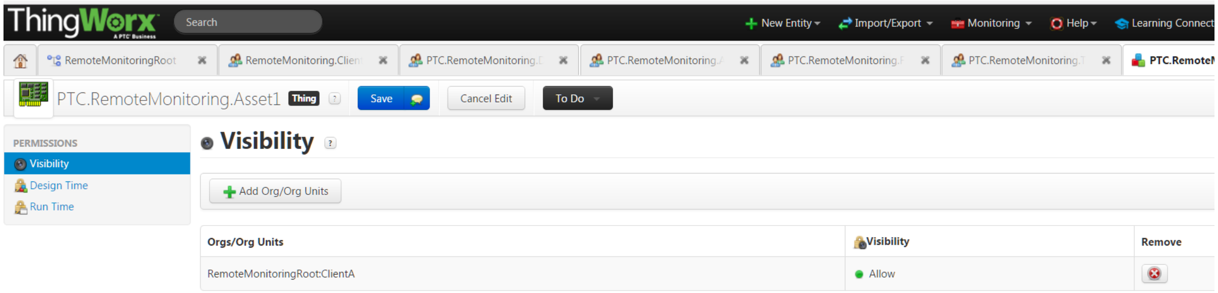


To edit members of this group click "Edit Members" from the toolbar

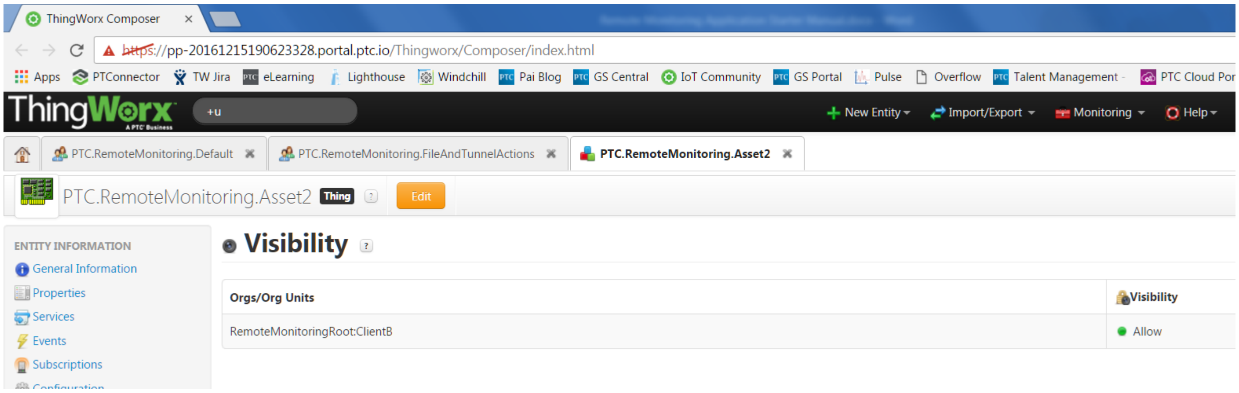
Name
RemoteMonitoringTechnician

Visibility and Permissions are then assigned for the Visiblity Group and Application Groups in the actual model. In Addition the System user receives Runtime Service Execute, to allow wrapped services to execute.

Visibility assignment

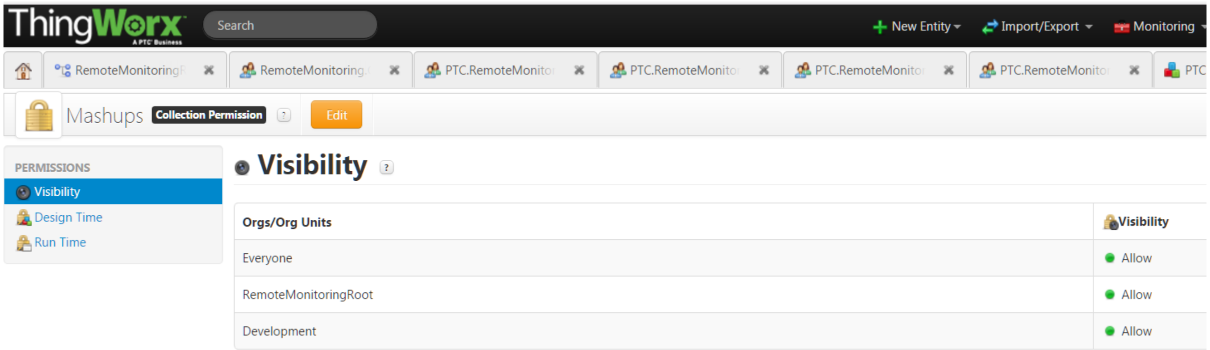


Orgs/Org Units	Visibility	Remove
RemoteMonitoringRoot:ClientA	Allow	☒



Orgs/Org Units	Visibility
RemoteMonitoringRoot:ClientB	Allow

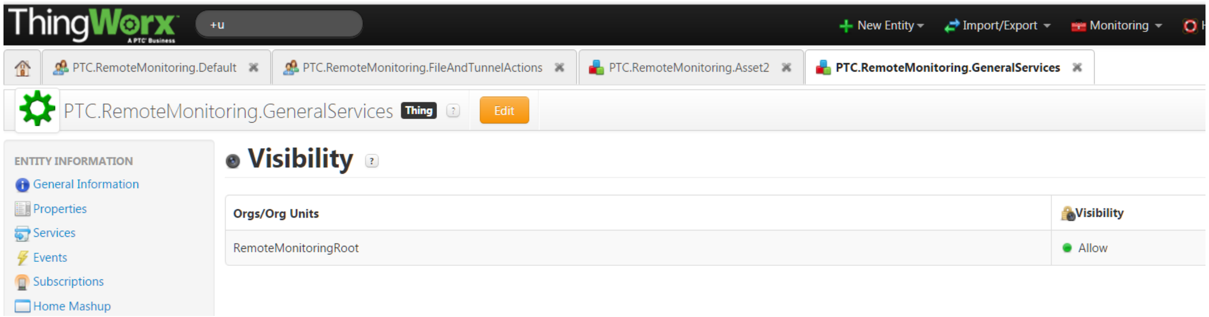
Specific Asset Visibility



Visibility

Orgs/Org Units	Visibility
Everyone	● Allow
RemoteMonitoringRoot	● Allow
Development	● Allow

All Users in the Organization have Visibility to all Mashups

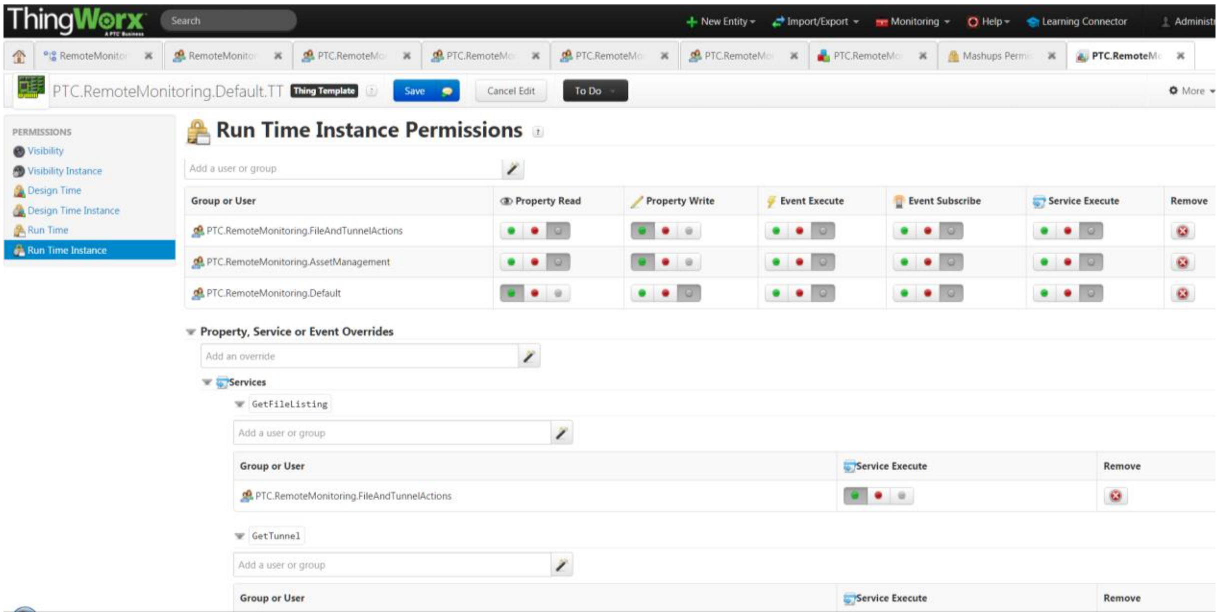

ThingWorx
New Entity
Import/Export
Monitoring
PTC.RemoteMonitoring.Default
PTC.RemoteMonitoring.FileAndTunnelActions
PTC.RemoteMonitoring.Asset2
PTC.RemoteMonitoring.GeneralServices
PTC.RemoteMonitoring.GeneralServices
Thing
Edit
ENTITY INFORMATION
General Information
Properties
Services
Events
Subscriptions
Home Mashup

Visibility

Orgs/Org Units	Visibility
RemoteMonitoringRoot	● Allow

All Users in the Organization have Visibility to this Thing which has Application Supporting Services

Permission Assignments



Run Time Instance Permissions

Group or User	Property Read	Property Write	Event Execute	Event Subscribe	Service Execute	Remove
PTC.RemoteMonitoring.FileAndTunnelActions	● ● ○	● ● ○	● ● ○	● ● ○	● ● ○	☒
PTC.RemoteMonitoring.AssetManagement	● ● ○	● ● ○	● ● ○	● ● ○	● ● ○	☒
PTC.RemoteMonitoring.Default	● ● ○	● ● ○	● ● ○	● ● ○	● ● ○	☒

Property, Service or Event Overrides

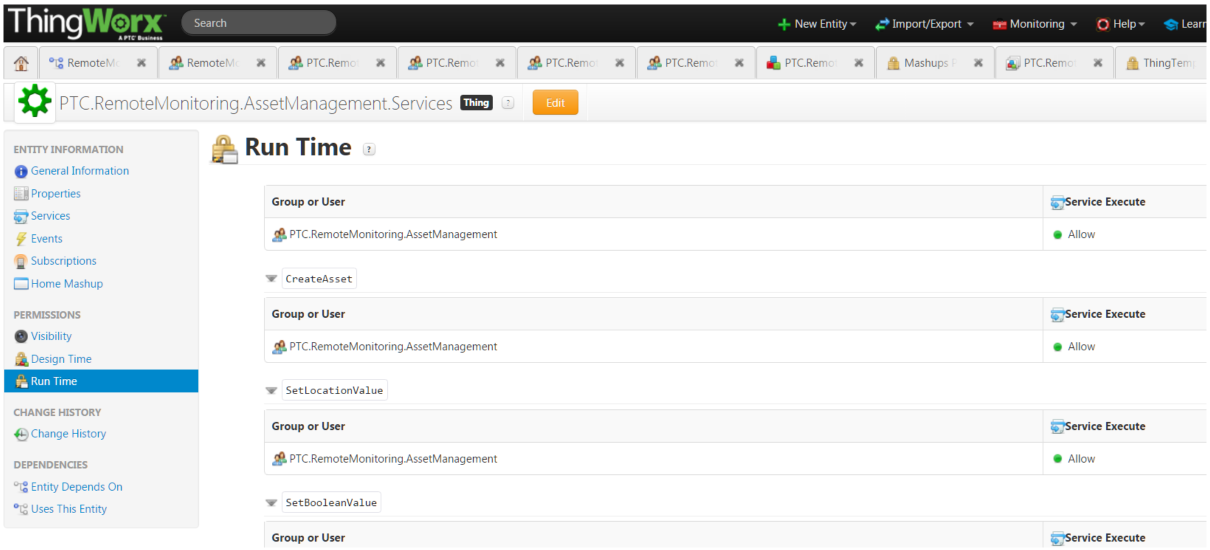
Services

`GetFileListing`

Group or User	Service Execute	Remove
PTC.RemoteMonitoring.FileAndTunnelActions	● ● ○	☒

`GetTunnel`

Group or User	Service Execute	Remove



Run Time

Group or User	Service Execute
PTC.RemoteMonitoring.AssetManagement	Allow

`CreateAsset`

Group or User	Service Execute
PTC.RemoteMonitoring.AssetManagement	Allow

`SetLocationValue`

Group or User	Service Execute
PTC.RemoteMonitoring.AssetManagement	Allow

`SetBooleanValue`

Group or User	Service Execute

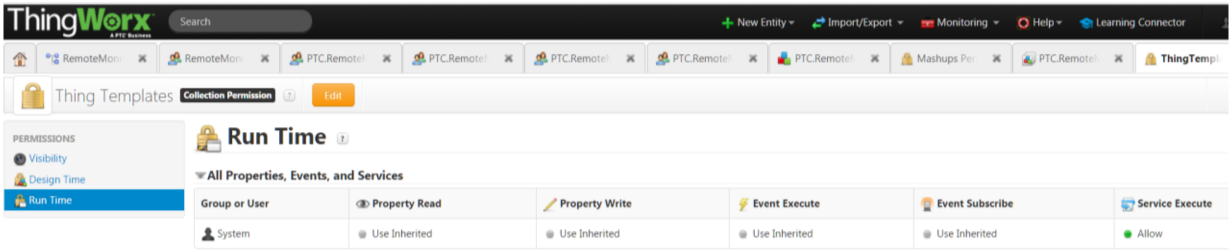
Permissions are assigned to the Application Groups,which will contain the specific Client Role Groups, so all Clients can leverage these groups.

For Properties Read/Write an all Read Write is given, although you could do specific properties, you can also override deny specific properties.

For Services each individual Service is specifically permitted.

NOTE: Permissions stack, so in this case File/Tunnel and Management are also IN Default and receive Property Read through that.

System User Assignment



Run Time

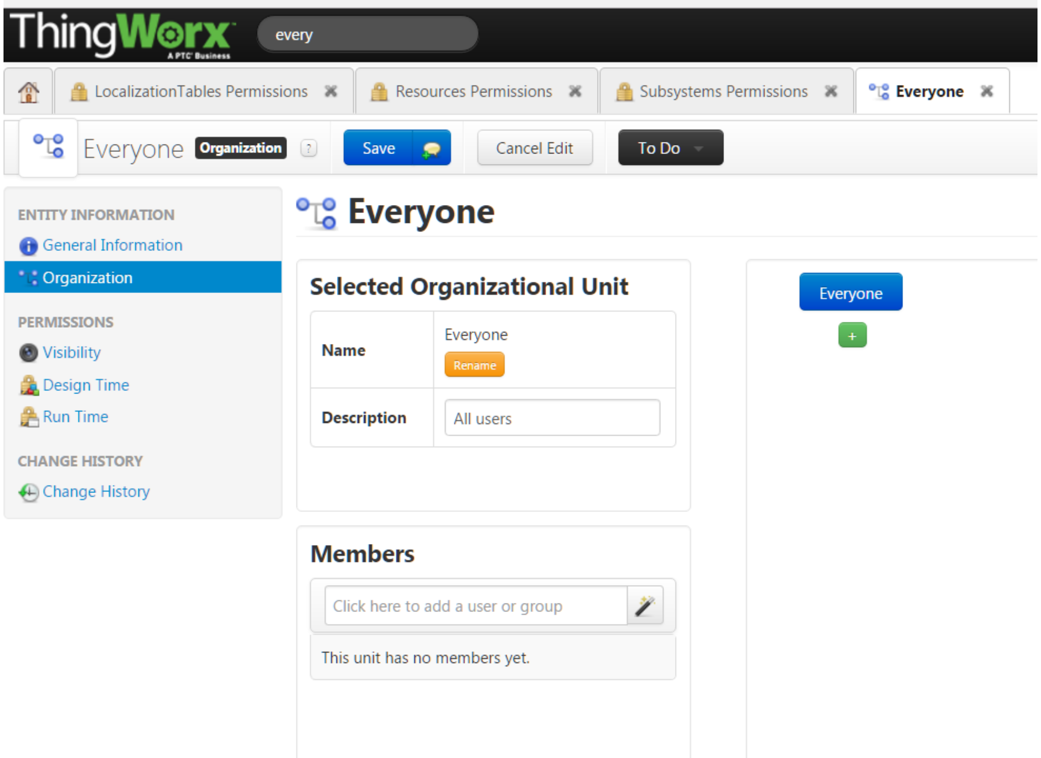
All Properties, Events, and Services

Group or User	Property Read	Property Write	Event Execute	Event Subscribe	Service Execute
System	Use Inherited	Use Inherited	Use Inherited	Use Inherited	Allow

System user is assigned over Entity Types to have Service Execute for all services.

This for example allows a Service like CreateEntity (seen in PTC.RemoteMonitoring.AssetManagement.Services) to utilize the service CreateThing found in Resources/EntityServices

Everyone Organization



Everyone

Selected Organizational Unit

Name	Everyone
Description	All users

Members

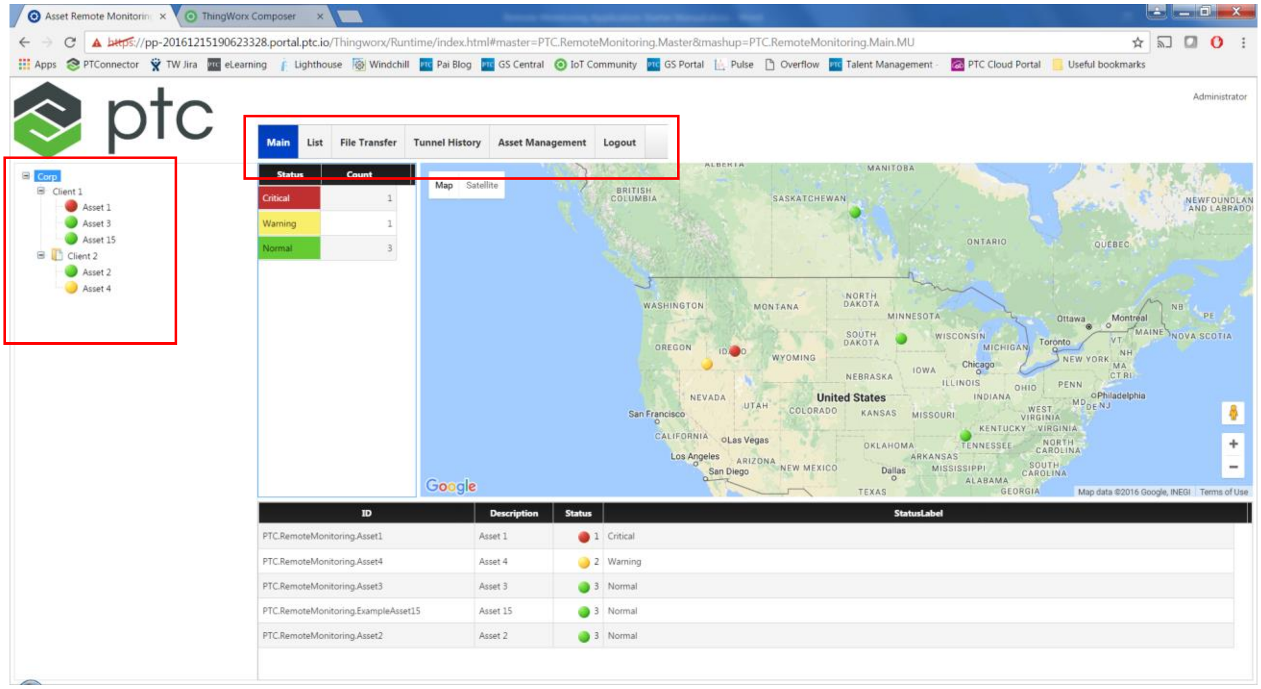
Click here to add a user or group

This unit has no members yet.

Everyone

NOTE: For visibility to be Enabled, remove Users from the Everyone Organization.

Administrator View

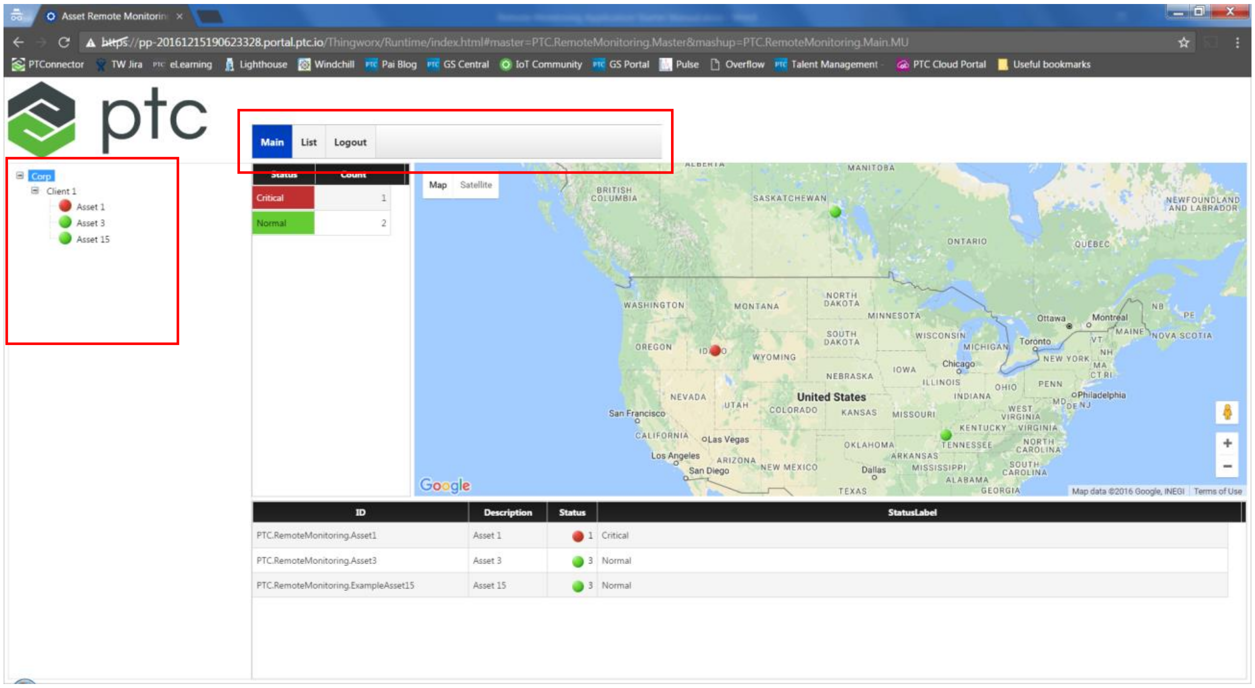

Asset Remote Monitorin
ThingWorx Composer
https://pp-20161215190623328.portal.ptc.io/Thingworx/Runtime/index.html#master=PTC.RemoteMonitoring.Master&mashup=PTC.RemoteMonitoring.Main.MU
Administrator
Main
List
File Transfer
Tunnel History
Asset Management
Logout
Map
Satellite
Google
Map data ©2016 Google, INEGI
Terms of Use

- Corp
  - Client 1
    - Asset 1
    - Asset 3
    - Asset 15
  - Client 2
    - Asset 2
    - Asset 4

Status	Count
Critical	1
Warning	1
Normal	3

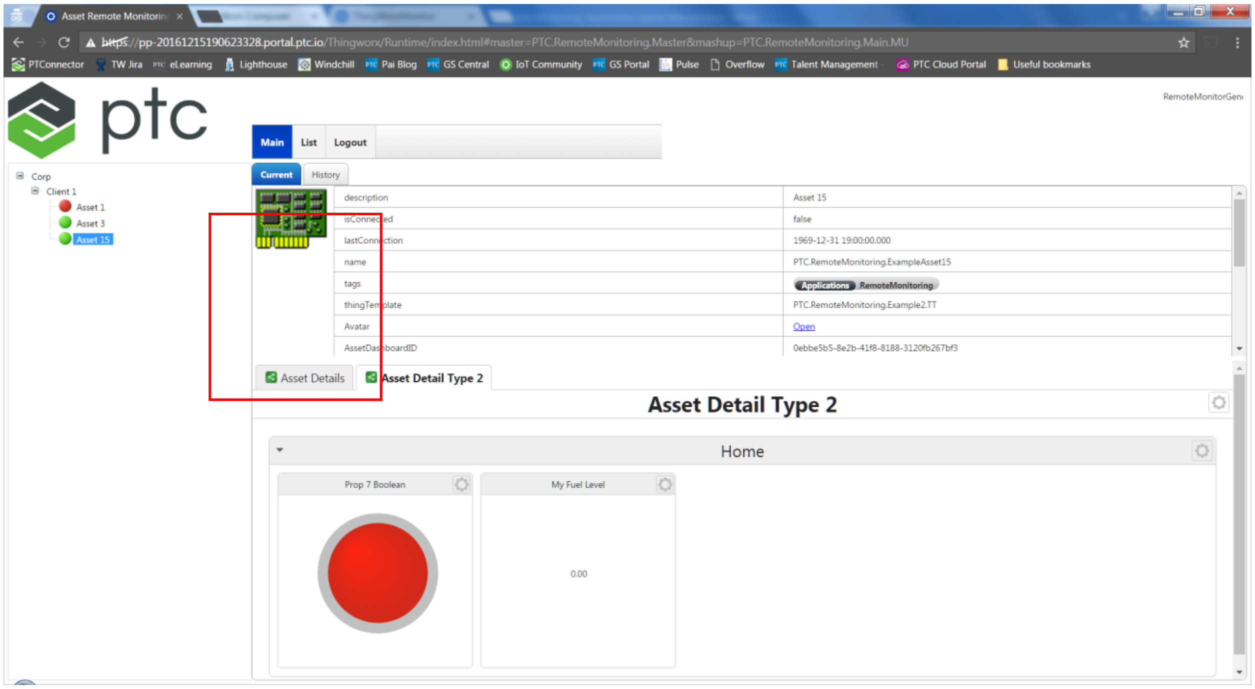
ID	Description	Status	StatusLabel
PTC.RemoteMonitoring.Asset1	Asset 1	● 1	Critical
PTC.RemoteMonitoring.Asset4	Asset 4	● 2	Warning
PTC.RemoteMonitoring.Asset3	Asset 3	● 3	Normal
PTC.RemoteMonitoring.ExampleAsset15	Asset 15	● 3	Normal
PTC.RemoteMonitoring.Asset2	Asset 2	● 3	Normal

General User View


Asset Remote Monitorin
https://pp-20161215190623328.portal.ptc.io/Thingworx/Runtime/index.html#master=PTC.RemoteMonitoring.Master&mashup=PTC.RemoteMonitoring.Main.MU
Main
List
Logout
Corp
Client 1
Asset 1
Asset 3
Asset 15
Map
Satellite
Google
Map data ©2016 Google, INEGI
Terms of Use

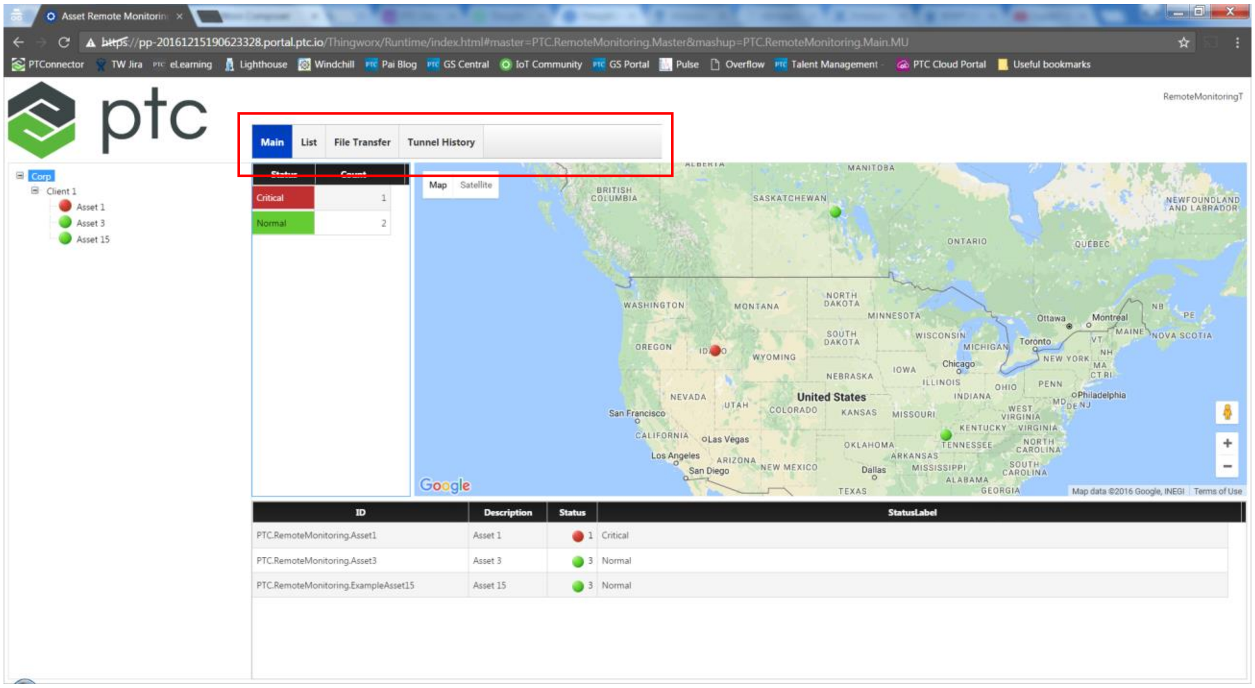
Status	Count
Critical	1
Normal	2

ID	Description	Status	StatusLabel
PTC.RemoteMonitoring.Asset1	Asset 1	🔴 1	Critical
PTC.RemoteMonitoring.Asset3	Asset 3	🟢 3	Normal
PTC.RemoteMonitoring.ExampleAsset15	Asset 15	🟢 3	Normal



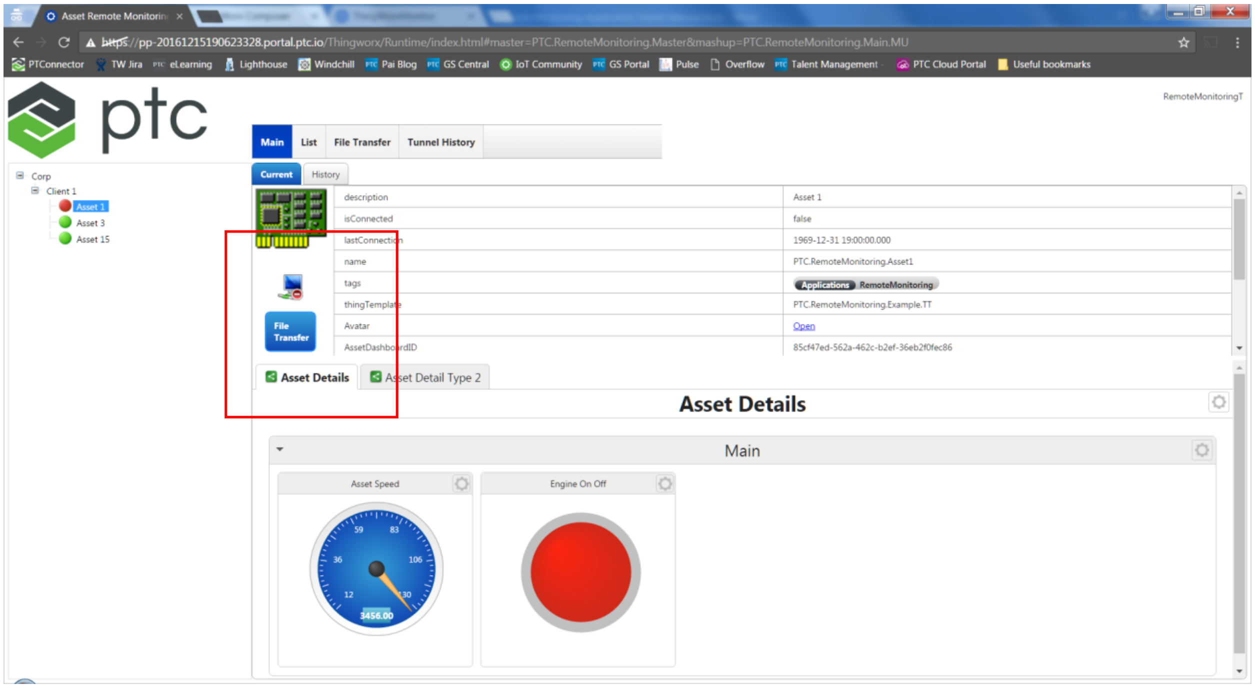
description	Asset 15
isConnected	false
lastConnection	1969-12-31 19:00:00.000
name	PTC.RemoteMonitoring.ExampleAsset15
tags	Applications RemoteMonitoring
thingTemplate	PTC.RemoteMonitoring.Example2.TT
Avatar	Open
AssetDashboardID	0ebbe5b5-8e2b-41f8-8188-3120fb267bf3

Technician View


Asset Remote Monitorin
https://pp-20161215190623328.portal.ptc.io/Thingworx/Runtime/index.html#master=PTC.RemoteMonitoring.Master&mashup=PTC.RemoteMonitoring.Main.MU
RemoteMonitoringT
Main
List
File Transfer
Tunnel History
Corp
Client 1
Asset 1
Asset 3
Asset 15
Map
Satellite

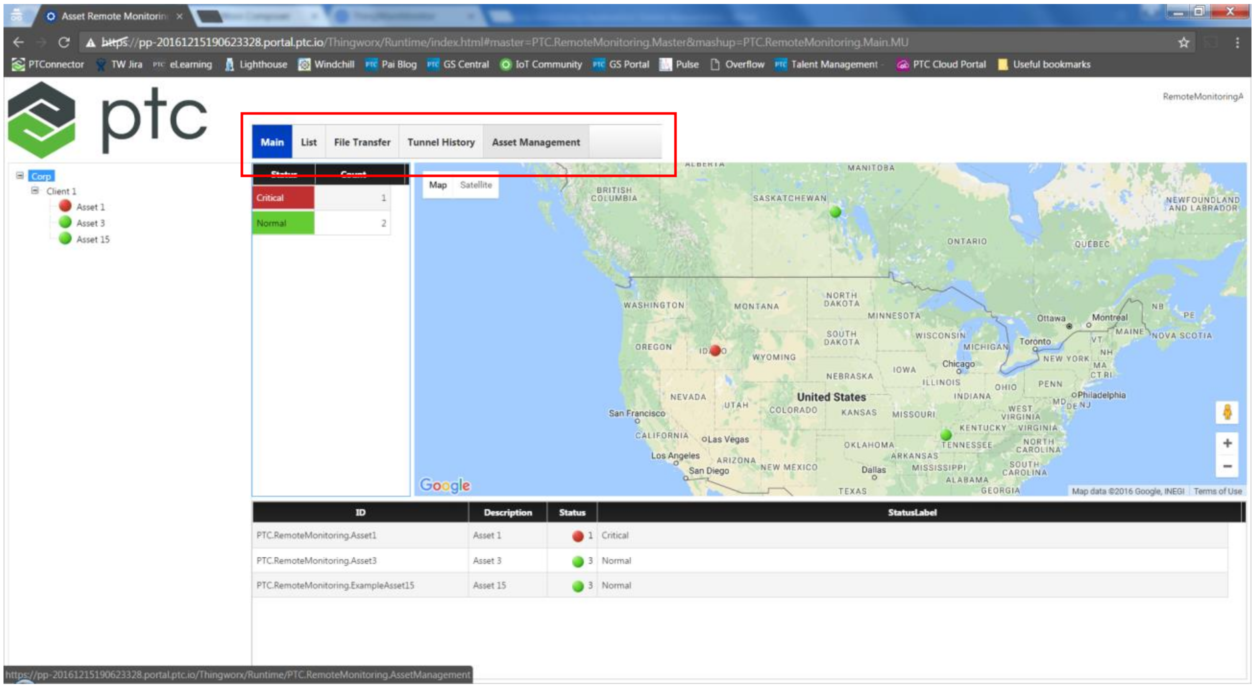
Status	Count
Critical	1
Normal	2

ID	Description	Status	StatusLabel
PTC.RemoteMonitoring.Asset1	Asset 1	● 1	Critical
PTC.RemoteMonitoring.Asset3	Asset 3	● 3	Normal
PTC.RemoteMonitoring.ExampleAsset15	Asset 15	● 3	Normal



description	Asset 1
isConnected	false
lastConnection	1969-12-31 19:00:00.000
name	PTC.RemoteMonitoring.Asset1
tags	Applications RemoteMonitoring
thingTemplate	PTC.RemoteMonitoring.Example.TT
Avatar	Open
AssetDashboardID	85cf47ed-562a-462c-b2ef-36eb2f0fec86

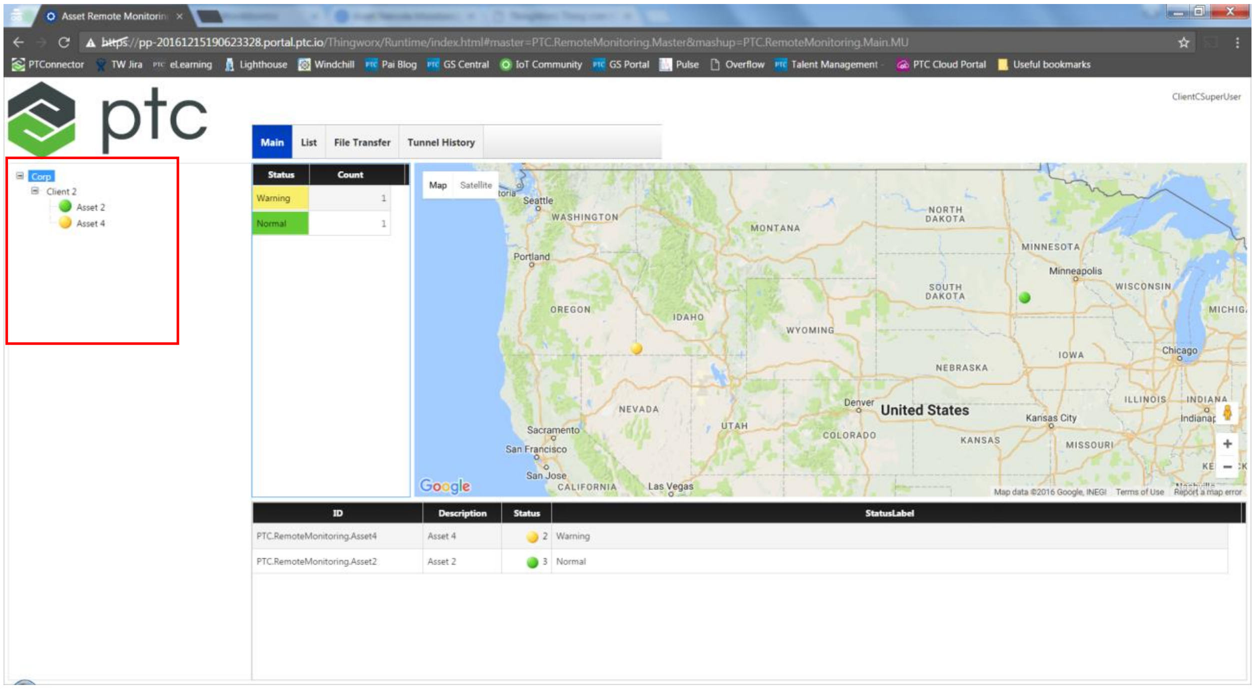
Asset Manager View


Asset Remote Monitorin
https://pp-20161215190623328.portal.ptc.io/Thingworx/Runtime/index.html#master=PTC.RemoteMonitoring.Master&mashup=PTC.RemoteMonitoring.Main.MU
RemoteMonitoringA
Main
List
File Transfer
Tunnel History
Asset Management
Corp
Client 1
Asset 1
Asset 3
Asset 15
Map
Satellite
https://pp-20161215190623328.portal.ptc.io/Thingworx/Runtime/PTC.RemoteMonitoring.AssetManagement

Status	Count
Critical	1
Normal	2

ID	Description	Status	StatusLabel
PTC.RemoteMonitoring.Asset1	Asset 1	🔴 1	Critical
PTC.RemoteMonitoring.Asset3	Asset 3	🟢 3	Normal
PTC.RemoteMonitoring.ExampleAsset15	Asset 15	🟢 3	Normal

Client B View


Asset Remote Monitorin
https://pp-20161215190623328.portal.ptc.io/Thingworx/Runtime/index.html#master=PTC.RemoteMonitoring.Master&mashup=PTC.RemoteMonitoring.Main.MU
ClientCSuperUser
Main
List
File Transfer
Tunnel History
Corp
Client 2
Asset 2
Asset 4
Map
Satellite

Status	Count
Warning	1
Normal	1

ID	Description	Status	StatusLabel
PTC.RemoteMonitoring.Asset4	Asset 4	🟡 2	Warning
PTC.RemoteMonitoring.Asset2	Asset 2	🟢 3	Normal

Application Model Design Walkthrough

The Model will not be explained in full, but certain parts will be highlighted

ThingShapes

Two ThingShapes are in use

- PTC.RemoteMonitoring.Default.TS

  This is a default shape that must be included in all entities that are in the Network of the application.

- PTC.RemoteMonitoring.Session.TS

  This is a ThingShape that is incorporated in the UserManagement Configuration as a Session ThingShape and provides several Properties that become Session Parameters to be used Client Side and Server Side

ThingTemplates

There is one main ThingTemplate for Assets from which all actual Asset Templates should be derived from which is PTC.RemoteMonitoring.Default.TT. Base permissions are set in this ThingTemplate. Any Thing that implements this or a deriveved ThingTemplate will be considered an actual Asset vs. Corp or Client.

There is also PTC.RemoteMonitoring.FileRepo.TT which was created so as to help with permissioning.

Things

There are several categories of Things (including ValueStreams and Streams)

- Corp
  - PTC.RemoteMonitoring.GenericTopNode

  Generic entity to take up the top node in the hierarchical network.

- Clients
  - PTC.RemoteMonitoring.ExampleClient1
  - PTC.RemoteMonitoring.ExampleClient2

  Clients of the Corp who own assets, also created to be placed in the hierarchical network.

- Assets
  - PTC.RemoteMonitoring.Asset1
  - PTC.RemoteMonitoring.Asset2
  - PTC.RemoteMonitoring.Asset3
  - PTC.RemoteMonitoring.Asset4
  - PTC.RemoteMonitoring.ExampleAsset15

  Actual assets of interest, these are Remote type Things that receive their information from an Agent on a physical asset. They support File Transfer and Tunneling as well.

- Support
  - PTC.RemoteMonitoring.GeneralServices
  - PTC.RemoteMonitoring.DashboardServices
  - PTC.RemoteMonitoring.AssetManagement.Services

These Things hold general services that are used in the Application. They are split according to function so they can be secured for specific Application Groups if necessary. See AssetManagementServices as an example.

- FileRepository
    - PTC.RemoteMonitoring.DefaultRepository

Generic starting repository

- ValueStream
    - PTC.RemoteMonitoring.Default.VS

ValueStream has been assigned to the PTC.RemoteMonitoring.Default.TT ThingTemplate and logs all the Properties that are set to Logged and drives the Asset Detail History screen.

- Stream
    - PTC.RemoteMonitoring.TunnelHistory.Stream
    - PTC.RemoteMonitoring.FileTransferHistory.Stream

These hold the Tunnel and FileTransfer historical events respectively and drive the History screens.

Networks

There is one network in the model PTC.RemoteMonitoring.Network which represents the Corp/Asset hierarchy and is used to drive the retrieval of the information for the Asset Tree and Asset Status payloads.

Model Tags

Full application is tagged with Applications:RemoteMonitoring

Menus

There are two menus

- PTC.RemoteMonitoring.Application.Menu

This is the main application menu that is seen in the Master. It has specific User Groups assigned to each Menu Entry to show/hide the item for the appropriate Users

- PTC.RemoteMonitoring.GadgetLib.Menu

The Gadget Library menu is used in the Gadget Library mashup when adding gadgets.

NOTE: For both of these you will see a permission to execute GetEffectiveMenu which is needed for the Menu widget to properly render the menu.

Visualizations

Visualizations are described later, please note that several styles and state definitions were created to support the visualizations.

Notable Modeling Techniques

Inhertitance

Model is set up to leverage ThingShapes and ThingTemplates

Leveraging a Network to easily associate Entities and Assets

Using HomeMashup of Things

Using a Property to indicate associated Dashboard for Assets

Modeling Choices

Using both a ThingTemplate and ThingShape requirement for inclusion in the application to allow for Remote type and non-Remote type things/assets to appear in the application. Corp vs. Asset1 for example.

Using both a Number and String for Status to help with Sorting as well as friendly label display

No localization was used, just to save time. Better would've been to have all labels as tokens.

Naming Conventions

As much as possible consistent naming conventions are used.

Scripting Techniques

GetNetworkConnections vs. GetSubNetworkConnections

In PTC.RemoteMonitoring.GeneralServices it is used in GetAssetsForNavigation and GetAssetsByStatus, the first service is used to populate the Navigation Tree and retrieves ALL Nodes. The second service is used to populate the Map and Grids and receives a Selected Node input filter to only retrieve Selected Node and its Children.

GetAssetAggregatedByStatus

This service actually just retrieves the Aggregated values from a Session Parameter that was filled in by GetAssetsByStatus. Since this service already has all the information to do the aggregation, it would have been a waste to have to run that service once more to do the aggregation. As such the retrieval of the information is done first and returned as a result to the mashup, but also the aggregation is done and placed in Session, this way the same user can be logged in from many machines and see the relevant aggregation information based on their choices in that mashup.

DeriveFields

Thingworx provides a library of InfoTable functions, one of the most powerful ones being DeriveFields. DeriveFields can generate additional columns to your InfoTable and fill that with values that can be derived from ... nearly anything! Hard coded, based on a Service you call, based on a Property Value, based on other values within the InfoTable you are adding the column to.

Just remember for this Service (as well as Aggregate), no spaces between different column definitions and use a , (comma) as separator.

Here are some extracts from GetAssetByStatus in PTC.RemoteMonitoring.GeneralServices:

```
 //Calling another function using DeriveFields
//Note that the value thingTemplate is the actual value in the row of column thingTemplate!
var params = {
  types: "STRING" /* STRING */,
  t: AllItems /* INFOTABLE */,
  columns: "BaseTemplate" /* STRING */,
    expressions:
"Things['PTC.RemoteMonitoring.GeneralServices'].RetrieveBaseTemplate({ThingTemplateName:
thingTemplate})" /* STRING */
};

// result: INFOTABLE
var AllItemsWithBase = Resources["InfoTableFunctions"].DeriveFields(params);

//Getting values from other Properties
//to in this case is the value of the row in the column to
//Note the use of , and no spaces
//NOTE: You can make this even more generic with something like Things[to][propName]
var params = {
    types: "NUMBER,STRING,STRING,LOCATION" /* STRING */,
    t: AllAssets /* INFOTABLE */,
    columns: "Status,StatusLabel,Description,AssetLocation" /* STRING */,
    expressions:
"Things[to].Status,Things[to].StatusLabel,Things[to].description,Things[to].AssetLocation" /*
STRING */
};
// result: INFOTABLE
```

var AllAssetsWithStatus = Resources["InfoTableFunctions"].DeriveFields(params);

Generic Property Value Retrieval

Used in PTC.RemoteMonitoring.DashboardServices each of the ….Value services like GetAssetBooleanPropertyValue helps retrieve values based on the input of the Asset name and Property name. There had to be this many services to properly return values of each proper BaseType.

Code is the same for all of them: result = Things[AssetName][PropertyName]

This is used for the Gadgets so we can use them as generic as possible.

Generically Setting Property Values

Similar to being able to retrieve Property Values generically, you can set them as well.

This is used for Asset Configuration and the services can be found in: with the services each being called Set …. Value. Again all it needs is the Asset and Property name, now the difference is not the ouput, but the input BaseType.

Code is same for all of them: result = Things[AssetName][PropertyName] = Value;

Recursive Functions

Full fledged JavaScript is allowed and so you can define Functions inside Services and use them recursively. See the example in: PTC.RemoteMonitoring.GeneralServices in a service called RetrieveBaseTemplate.

Creating new Assets

Thingworx comes with EntityServices which will allow you to create any Entities as required. To create Things some special steps are required. After a CreateThing call, you must Enable and Restart the Thing

Service can be found in: PTC.RemoteMonitoring.AssetManagement.Services in the CreateAsset Service.

Use of CurrentUser and retrieving User Groups

In PTC.RemoteMonitoring.GeneralServices there is a Service FileTransferAndTunnelPermission that is used to determine if the User is in a particular User Group. This drives visibility on a Mashup.

As a best practice when you need to determine something based on the context of the User who is logged in, always use: GetCurrentUser server side, vs. an Input parameter which can be abused.

Also you can see the use of the GetGroups service.

Managing the Corporate Entity/Asset network

In PTC.RemoteMonitoring.AssetManagement.Services there are also services that show how you can easily manage a Network using Services. As a general rule almost anything that you do in Composer (outside of Mashup building) can be done with Services, or if you think beyond that picture, REST API calls.

NOTE: At the very beginning I entertained the thought of having a Dynamic Network assignment, but felt that there wasn't a real use case for that, it is simple enough though to take any services that use a Network (currently hardcoded name) and make that 'dynamic' using Input parameters or even Session Parameters.

Application Mashup Design Walkthrough

Master

PTC.RemoteMonitoring.Master provides the frame around the application and holds the Logo and Menu.

Providing a Browser tab Title

Within the Master the Title property is set and that overrides the default title on the browser tab so now it reads: Asset Remote Monitoring

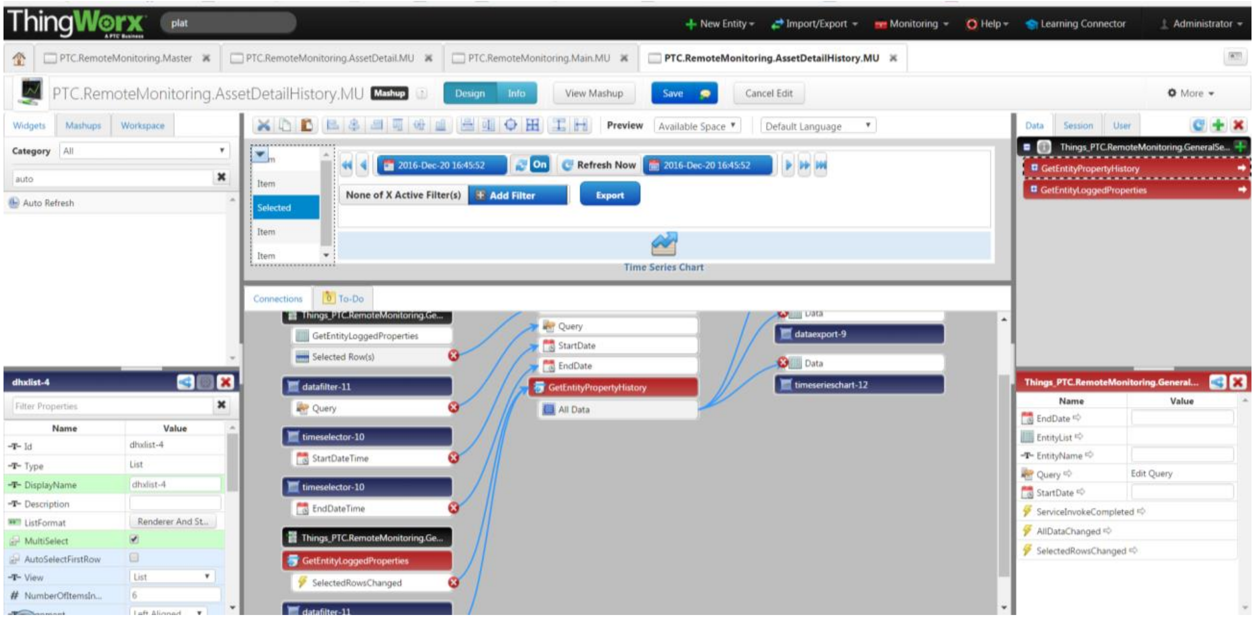
Asset Tree

In using a network it is very easy to get that information into the Tree widget in PTC.RemoteMonitoring.Main.MU. With DeriveFields we've added some extra information so that we can drive the State Based Definition with a Status field, but we also drive the Contained Mashup to show a Summary mashup or Detail mashup based on the Tree Entity selection.

Dynamic Asset History in TimeSeries Chart

Thingworx TimeSeries chart are great because they can AutoConfigure also Grids can handle varying InfoTable returns when you use 'Show All Columns'.

In this screen we provide the Selected Row that indicates the Asset and from there we pull the Numerical Properties that are Logged into a List. Then based on the List selection we pull the History of those Properties and display them in the TimeSeries Chart. In addition the Data Filter and Data Export were added, see: PTC.RemoteMonitoring.AssetDetailHistory.MU


ThingWorx
PTC.RemoteMonitoring.Master
PTC.RemoteMonitoring.AssetDetail.MU
PTC.RemoteMonitoring.Main.MU
PTC.RemoteMonitoring.AssetDetailHistory.MU
Design
Info
View Mashup
Save
Cancel Edit
Time Series Chart
Connections
GetEntityPropertyHistory
GetEntityLoggedProperties
dhxlist-4
dataexport-9
timeserieschart-12

Contained Mashups

All over the Application there is the use of Contained Mashups, this makes Mashups reusable in other places, it also makes the Contained Mashup space dynamically configurable as an example see the PTC.RemoteMonitoring.Main.MU, but also the PTC.RemoteMonitoring.AssetDetail.MU where we dynamically set the Dashboard to show based on the Asset's Dashboard ID property.

PTC.RemoteMonitoring.AssetDetail.MU itself is a slightly 'bad' or inconsistent example where there is a mix and match of actual Mashup and Contained Mashup for the Property History. To be consistent, there would've been a main Asset Container Mashup and then two contained Mashups one of the Asset Detail and the other with the Asset History.

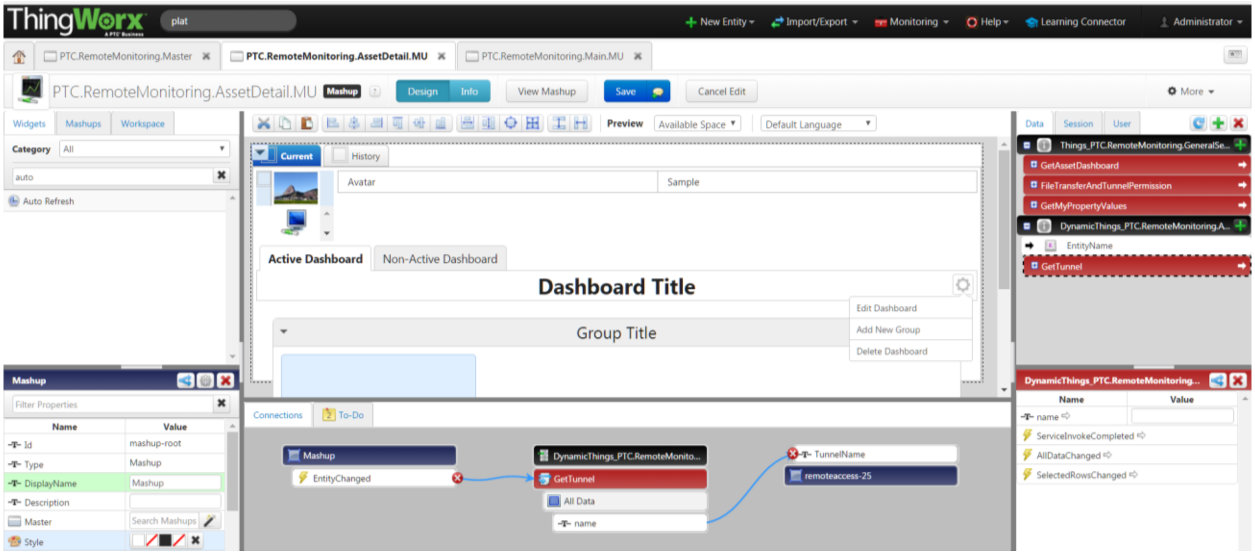
User Context

Where necessary there is User Context in the UI, like the Menu which has its Menu Items defined with User Groups. Also in the Asset Detail it determines if you have FileTransfer/Tunnel permissions and together with a Validation Widget it show/hides those options.

Selection Context

Selection context is used everywhere in the Application and always starts from 'Selected Row' of a Data Source. This is then passed either through Mashup parameters, Service Input parameters or Session Parameters. Actions are then invoked through 'Selected Row Changed' of the Data Source or 'Double Click' of the Widget and sometimes pass into a Navigation Widget to create a Navigation or into another Data Service to retrieve contextual information. Sometimes Refresh request are being passed to force a Service to run in a different Mashup.

Here is an example where the Tunnel is assigned to the Remote Access widget dynamically in PTC.RemoteMonitoring.AssetDetail.MU:

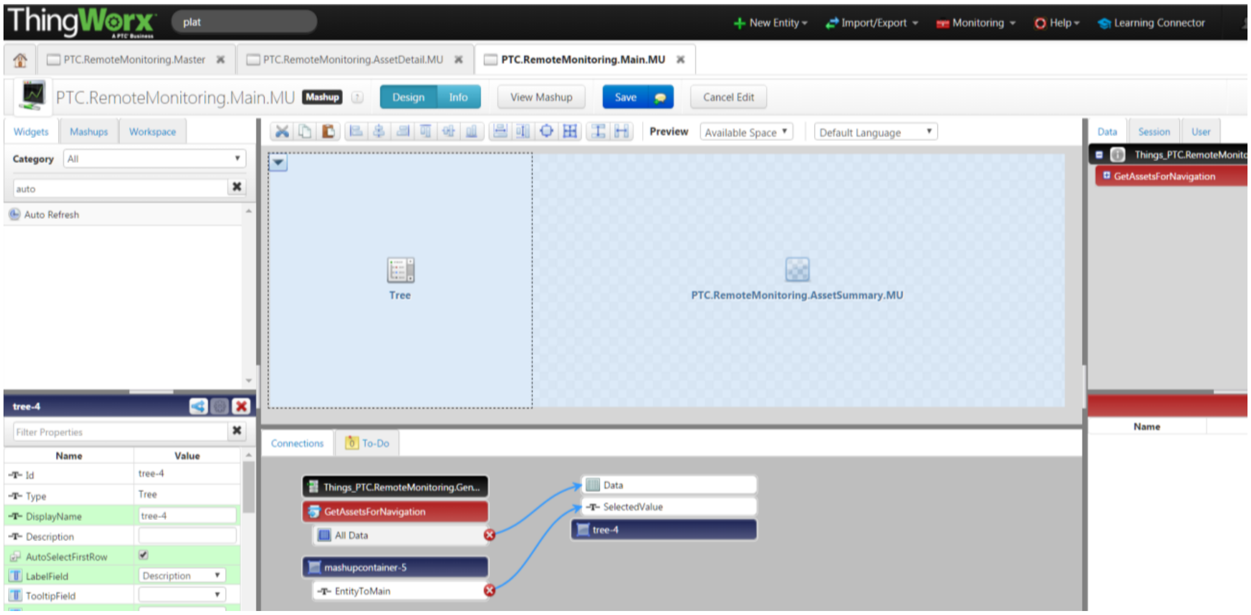


Combining Dynamic Contained Mashups and Mashup Parameters

One very powerful technique that combines Contained dynamic mashups and the ability to pass Mashup Parameters is also used.

A tutorial entry can be found [here]() on the Thingworx Community.

Here is a place where it is used in PTC.RemoteMonitoring.Main.MU:



Application Dashboard Design Walkthrough

Dashboards were added to the Asset Detail to provide the User to add specific Properties to monitor on their Assets. Dashboards allow a User not only to re-arrange the elements they've chose, but indeed allow users to pick and remove visualizations.

A very dynamic setup was created to allow Users to pick a variety of Visualizations based upon BaseType and tie them to any Property Value of choice.

Once a Gadget (visualization) is added it will use a Session Parameter to dynamically set the Asset context, but it will be locked into the Property context.

Dashboard

Dashboards are actually widgets and so the bottom section of PTC.RemoteMonitoring.AssetDetail.MU has been dedicated to the Dashboard Widget.

A parameter is passed in from the Asset Property to set the Default selected Dashboard.

A Gadget Library must be defined to allow the Addition of Gadgets.

Gadget Library

One of the biggest challenges with Gadgets is that they have to be pre-defined. So a User is limited to use the visualizations you provide. Gadgets are easily added though.

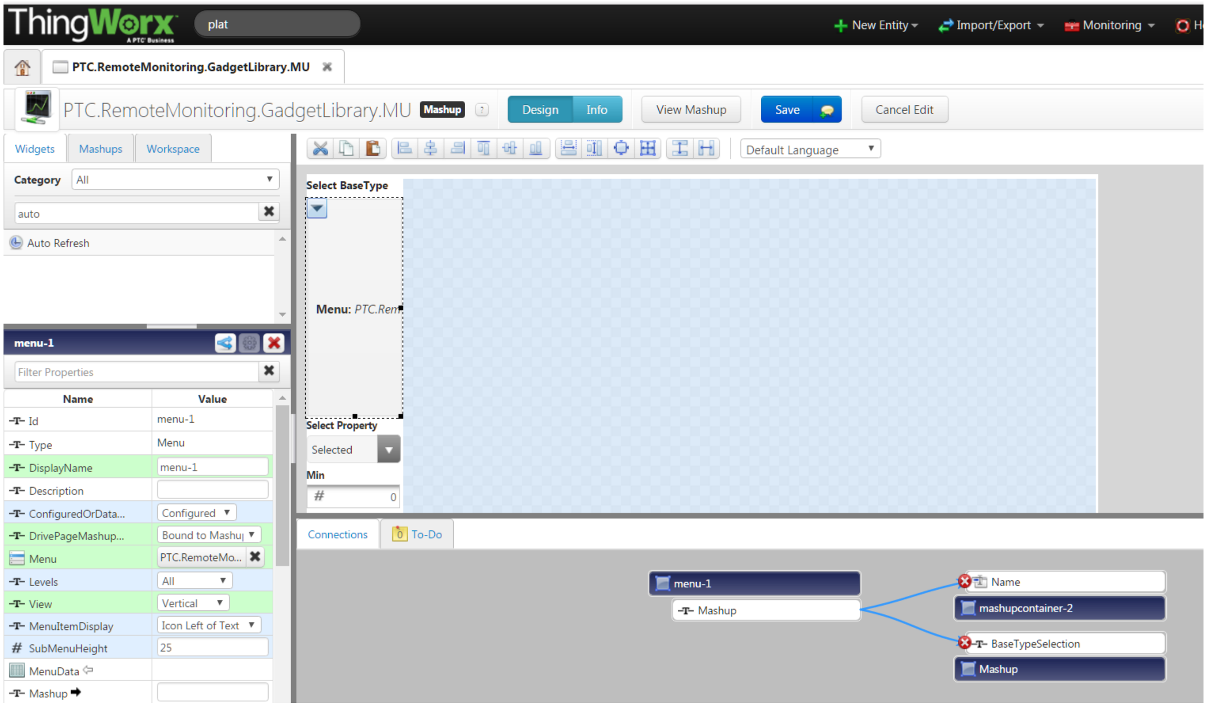
PTC.RemoteMonitoring.GadgetLibrary.MU is the Gadget Library Mashup.

Remember that your Gadget library is a full fledge mashup and all common techniques apply.

Menu to drive Gadget grouping (vs mashups)

The Menu is hardcoded for the base types, but the menu widget is used in Data Driven mode to drive a value in the Contained Mashup. Mashups were named slightly differently to be easily distinguished and do not follow Best Practice naming convention.

The BaseType also drives into a Service that retrieves the appropriate Properties



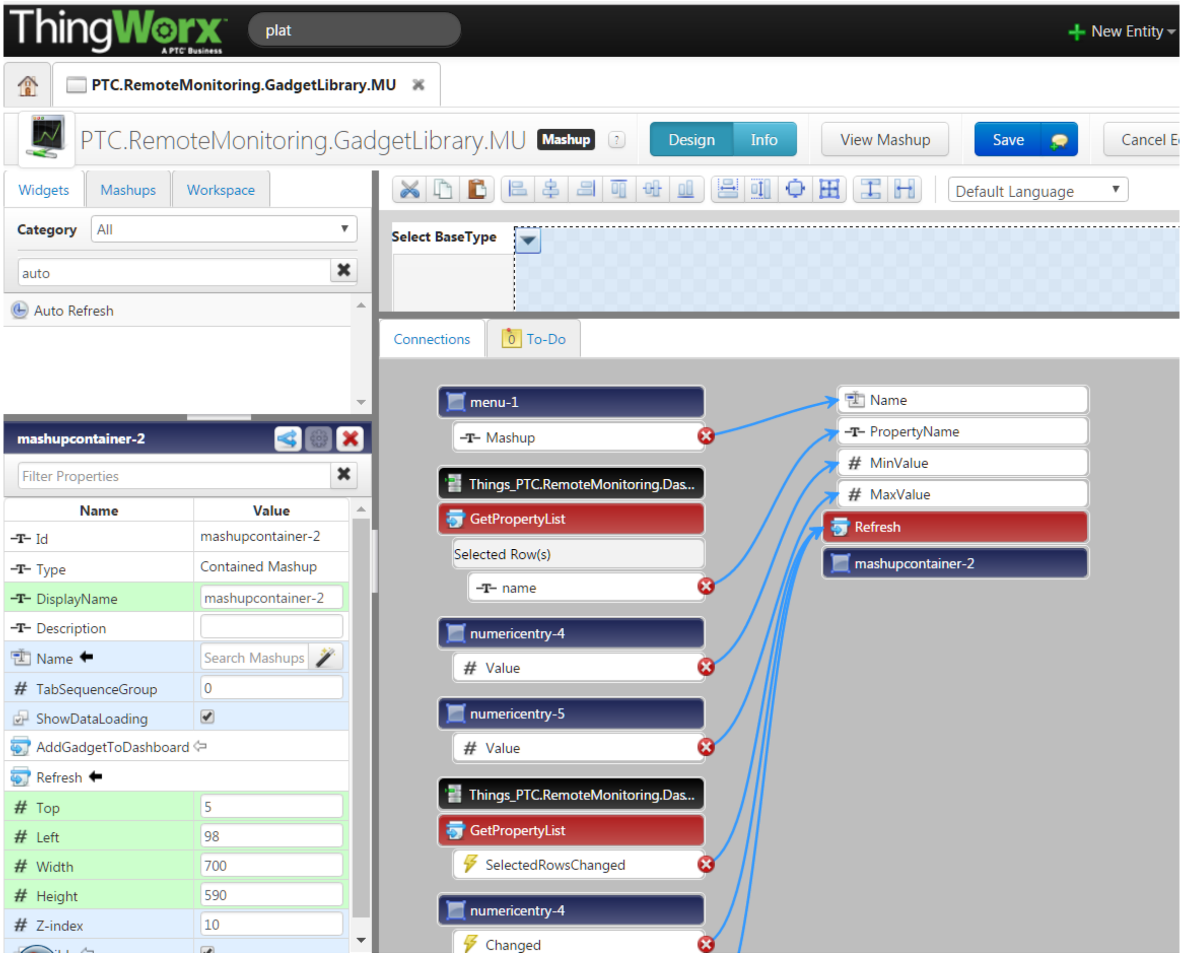
menu-1

Name	Value
Id	menu-1
Type	Menu
DisplayName	menu-1
Description	
ConfiguredOrData...	Configured
DrivePageMashup...	Bound to Mashu
Menu	PTC.RemoteMo...
Levels	All
View	Vertical
MenuItemDisplay	Icon Left of Text
SubMenuHeight	25
MenuData	
Mashup	

Mashup parameters and Refresh Request to Drive Gadget content

Mashup parameters are used to drive the Gadget content as well as Events driving the Refresh request.

Note: A Validation Widget makes two Numeric Entries Visible to provide a Min and Max value for Numeric Gadgets. You can bind additional Unused Mashup parameters (Used in the Numeric Gadget Mashup but not in the Text Gadget Mashup) and this will not break the Mashup.

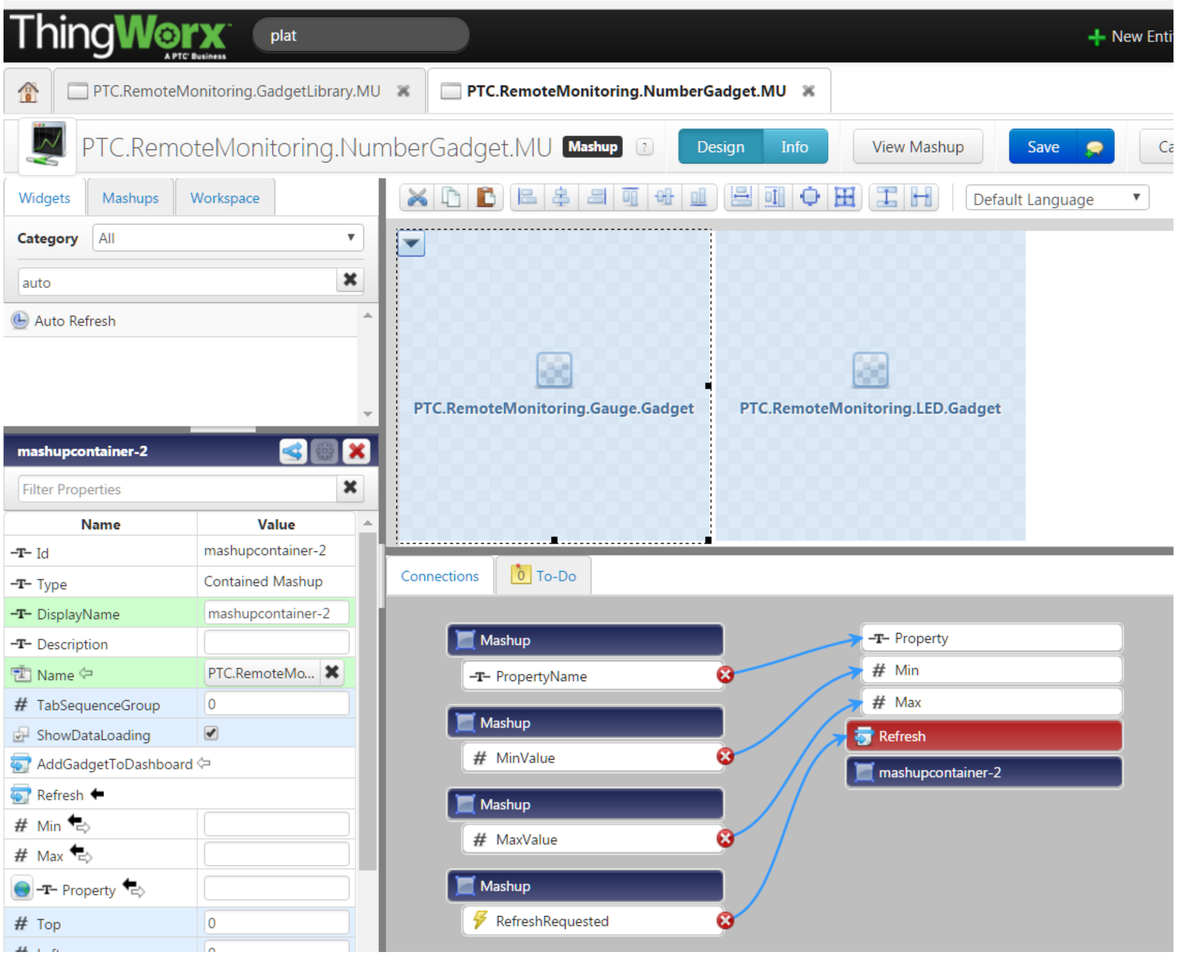


Name	Value
Id	mashupcontainer-2
Type	Contained Mashup
DisplayName	mashupcontainer-2
Description	
Name	Search Mashups
TabSequenceGroup	0
ShowDataLoading	☑
AddGadgetToDashboard	
Refresh	
Top	5
Left	98
Width	700
Height	590
Z-index	10

Multiple Gadgets in a Contained Mashup in a Contained Mashup

Gadgets can be added from any level of a Mashup. In this case the Gadgets are 'two levels' deep in a Contained Mashup on the Gadget Library Mashup. That also means Parameters are passed from Parent mashup into Child mashup into GrandChild mashup, into a Gadget.

See: PTC.RemoteMonitoring.NumberGadget.MU as an example.



Name	Value
Id	mashupcontainer-2
Type	Contained Mashup
DisplayName	mashupcontainer-2
Description	
Name	PTC.RemoteMo...
TabSequenceGroup	0
ShowDataLoading	☑
AddGadgetToDashboard	
Refresh	
Min	
Max	
Property	
Top	0

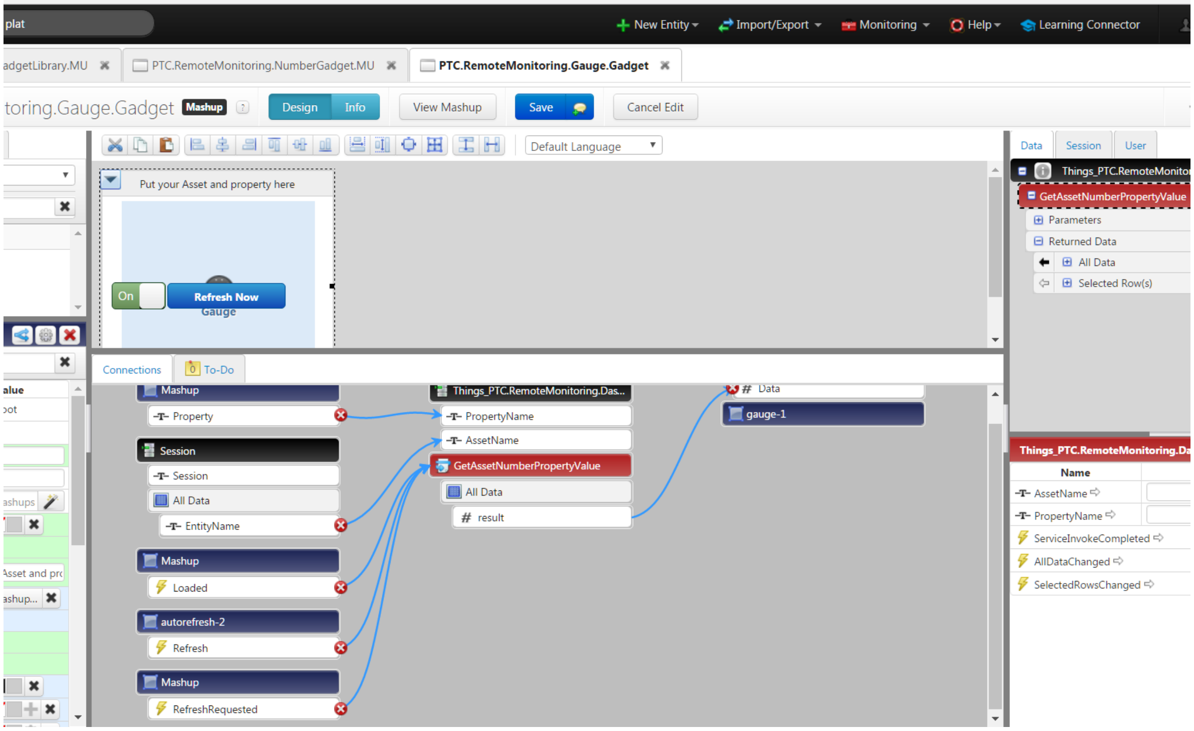
Gadgets

All the Gadgets work the same, a Service that takes an input from the Session which is the Asset and then through Mashup Parameters the Property Name and for Numbers the Min and Max.

Additionally there is a Refresh request being passed in to show up to date values when viewing the gadget library before adding it to the Dashboard.

Each Gadget has an AutoRefresh widget to show updated values while displaying in the Dashboard.

See PTC.RemoteMonitoring.Gauge.Gadget as an example.



Visiblity and Permissions for Dashboards

Dashboards require specific Visibility and Permissions settings. You can find those here in the documentation.