2. VANTIQ Business Application
Temperature Control Project

This tutorial guides a developer to define a Vantiq application that monitors temperature in cold Storage Room and launch actions in case of critical situations.

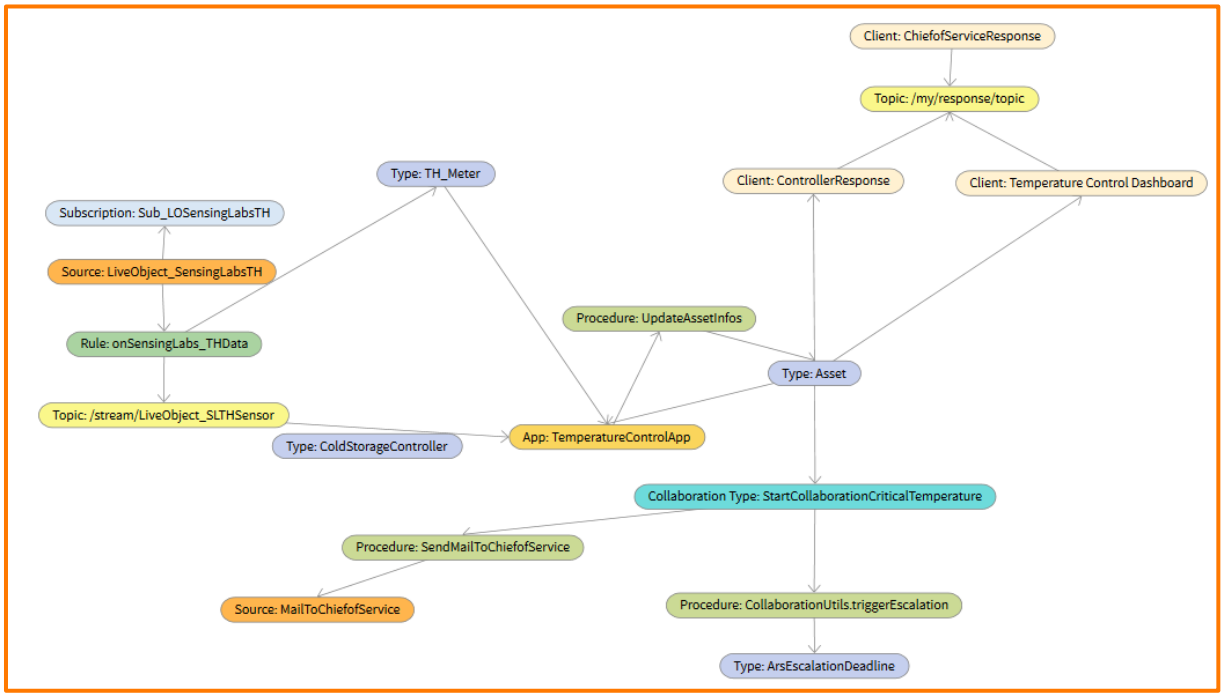
The Data Simulator project can be used to simulate the data of the sensors.

It is assumed that the developer has a working knowledge of the VANTIQ IDE. It is recommended that a new developer completes the lessons in the Introductory Tutorial before starting this tutorial.

Summary Table

1. Introduction

Project Modelo Graphical View of the project



2. Create a new Project

Use the **Projects** button, select **Create New Project** and title the project "TemperatureControl".

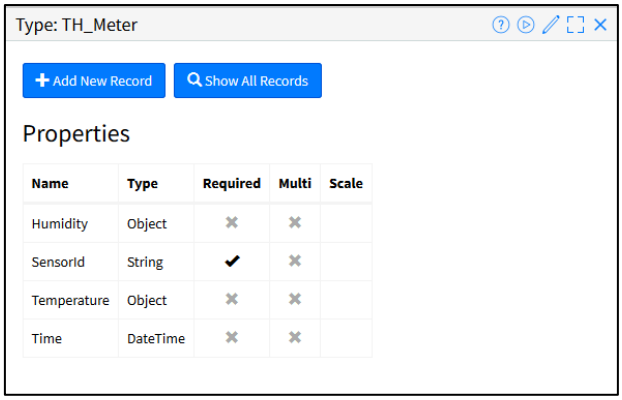
3. Create Data Types

A. Create a new data Type TH_Meter

This datatype is used to store the sensors data received from Live Objects.

Use the **Add** button to select **Type…**

Use the **New Type** button to create the “TH_Meter” datatype, and add the following properties.

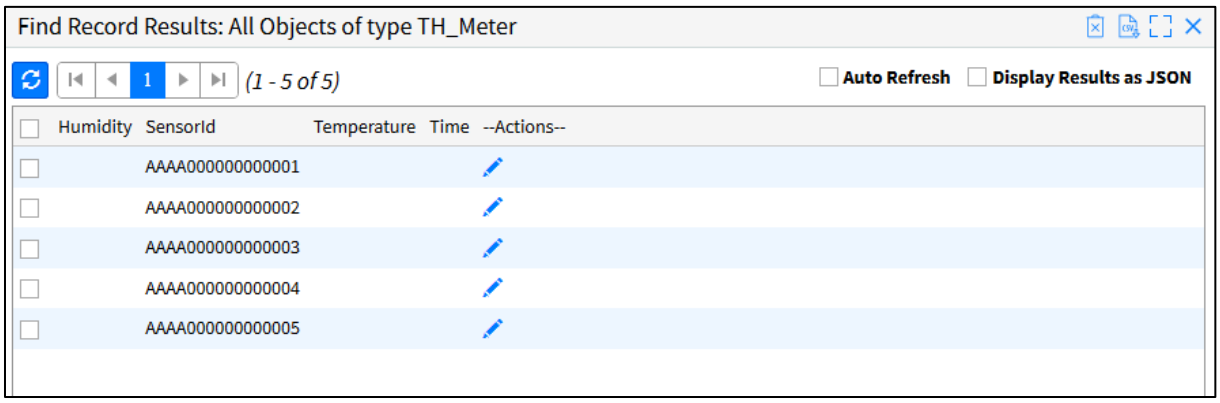


Name	Type	Required	Multi	Scale
Humidity	Object	✗	✗	
SensorId	String	✓	✗	
Temperature	Object	✗	✗	
Time	DateTime	✗	✗	

B. Record 5 sensors TH_Meter entries

Use the **Add New Record button** to record 5 sensors, with the following SensorId:

AAAA000000000001
AAAA000000000002
AAAA000000000003
AAAA000000000004
AAAA000000000005



Humidity	SensorId	Temperature	Time	--Actions--
	AAAA000000000001			
	AAAA000000000002			
	AAAA000000000003			
	AAAA000000000004			
	AAAA000000000005			

C. Create a new DataType Asset

Use the **Add** button to select **Type…**

Use the **New Type** button to create the “Asset” datatype and add the following properties.

Type: Asset

Properties

Name	Type	Required	Multi	Scale
Humidity	Real	✖	✖	
IndoorLocation	GeoJSON	✖	✖	
Name	String	✔	✖	
SensorData	Object	✖	✖	
SensorId	String	✖	✖	
Status	String	✖	✖	
Temperature	Real	✖	✖	

The Asset DataType will define the information on the Room where stock is stored:

- Name of the Room
- Location of the room
- Information on the Sensor installed in the room (Temperature,..)
- Status of the Room (critical, risky, normal)

D. Record 5 Rooms entries

Use the **Add New Record button** to record 5 assets, as defined below.

Humidity	IndoorLocation	Name	SensorData	SensorId	Status	Temperature	--Actions--
0	Lon 15, Lat 15	Room 1		AAAA000000000001		0	
0	Lon 20, Lat 25	Room 2		AAAA000000000002		0	
0	Lon 14, Lat 28	Room 3		AAAA000000000003		0	
0	Lon 8, Lat 22	Room 4		AAAA000000000004		0	
0	Lon 15, Lat 35	Room 5		AAAA000000000005		0	

Rooms are defined with different parameters:
- the room name
- the indoor location of the room (Point Location Type) . This corresponds to the location on the room in the floor plan.
- the SensorId of the sensor installed in the room.
- the SensorData that will be added when data are received

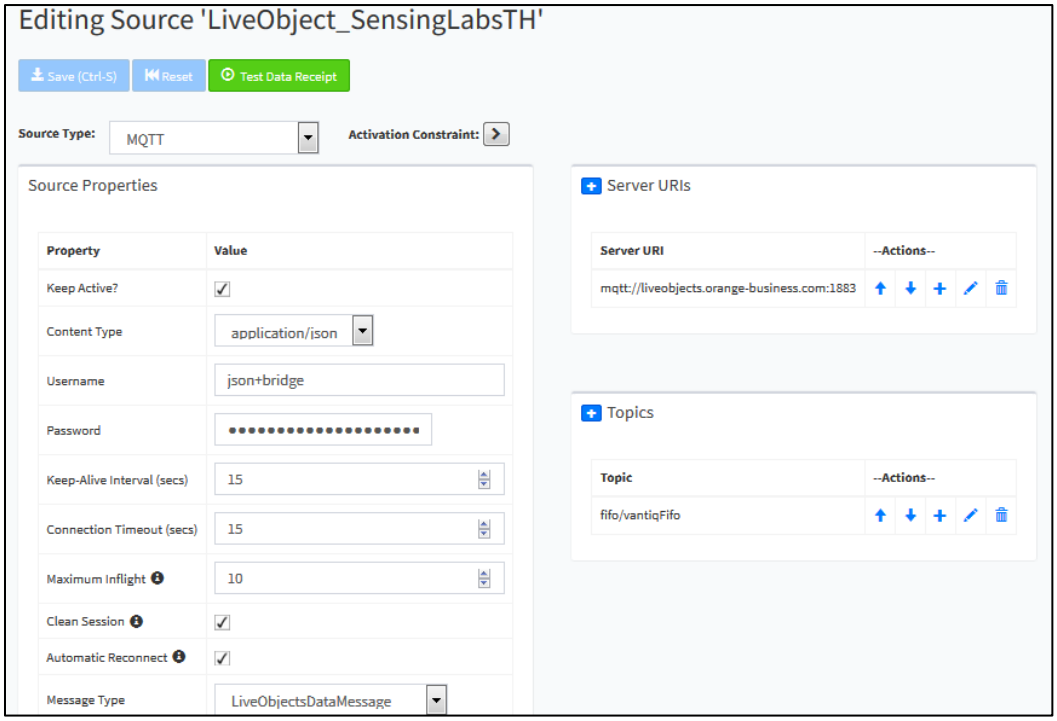
4. Create the Source to get data from Live Objects

The aim is to create a MQTT source connected to your Live Objects account to get the data of the SensingLabs Temperature and Humidity sensors.

The MQTT Fifo needs to be previously defined on Live Objects:
On Live Objects, in Configuration/Bus Messages, add a Fifo called “vantiqFifo”, with associated Routing Key ~event.v1.data.new.# . (It pushes on the Fifo all the messages received on Live Objects). Ensure that the Live Objects API-KEY that you are using has rights on this Fifo.

Use the **Add** button to select **Source…**
Use the **New Source** button to create the “LiveObject_SensingLabsTH” source, and add the following properties.

Content-Type : application/json
Username :json+bridge
Password : your liveObject API Key
Serveur URI : mqtt://liveobjects.orange-business.com:1883



Add a Rule to filter the data on devices of interest

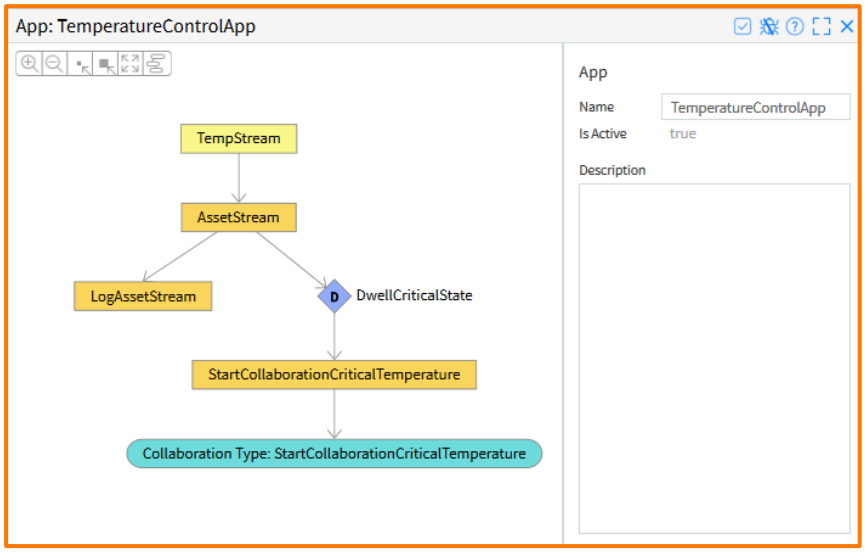
The MQTT topic forward data coming from all devices from your Live Objects account.
Data of interests are data from the Sensor devices recorded in TH_Meter Type.
The Rule will filter the data and push them to an internal topic.

Use the **Add** button to select **Rule…**
Use the **New Rule** button to create the "onSensingLabs_THData" rule, and define the following code :

```
RULE onSensingLabs_THData

WHEN MESSAGE ARRIVES FROM LiveObject_SensingLabsTH as LiveObjectDataReading

log.info("onSensingLabs_THData1")
var newSLSensor = {}


newSLSensor.Time = LiveObjectDataReading.timestamp
newSLSensor.SensorId = substring(LiveObjectDataReading.streamId, 9,25)
newSLSensor.Temperature = LiveObjectDataReading.value.temperature
newSLSensor.Humidity = LiveObjectDataReading.value.humidity

log.info("newSLSensor.Time" +stringify(newSLSensor.Time) )
log.info("newSLSensor.SensorId" +stringify(newSLSensor.SensorId) )
log.info("newSLSensor.Temperature" +stringify(newSLSensor.Temperature) )

select * from TH_Meter as thmeter
{
   log.debug("thmeter.SensorId "+stringify(thmeter.SensorId))
         log.debug("newSLSensor.SensorId "+stringify(newSLSensor.SensorId))
   if (thmeter.SensorId == newSLSensor.SensorId)
   {
      log.debug("match")
      upsert TH_Meter(newSLSensor)
         log.debug("onSensingLabs_THData : Publish " +stringify(newSLSensor))
                  publish newSLSensor to TOPIC "/stream/LiveObject_SLTHSensor"
   }
}
```

5. Create the Control Application

The application will monitor the temperature of the room and trigger collaboration in case of critical situation detection.

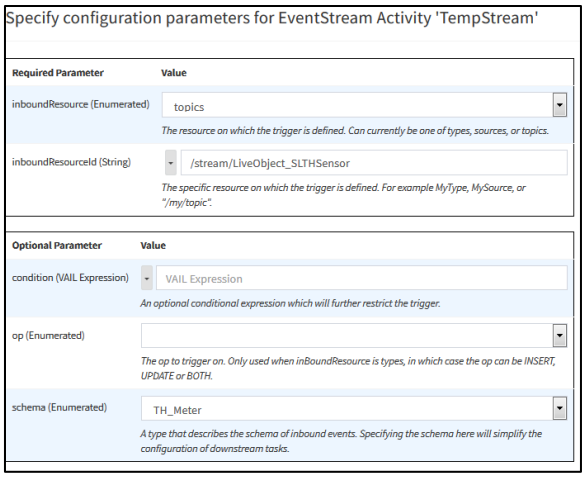


A. Create the App

Use the **Add** button to select **App.**
Use the **New App** button to create the “TemperatureControlApp” app.

B. Create the TempStream

Click on the **'EventStream' Activity** box, name it TempStream and configure it to get the Temp data from the "/stream/LiveObject_SLTHSensor defined previously.



C. Create a procedure that transforms TH_Meter data in Asset data

(In the project) Use the **Add** button to select **Procedure**

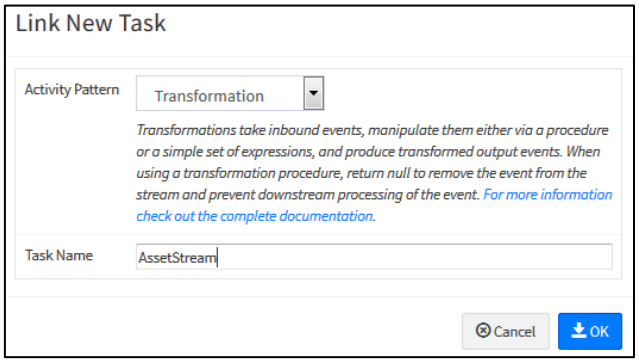
Use the **New Procedure** button to create the "UpdateAssetInfos" procedure, and define the following code :

```
PROCEDURE UpdateAssetInfos( thmeter)

log.info( "START UPDATE ASSET PROCEDURE for thmeter.SensorId:"+thmeter.SensorId)

var AssetInfo = {}
AssetInfo.SensorId = thmeter.SensorId
AssetInfo.SensorData = thmeter
AssetInfo.Temperature = thmeter.Temperature.value
AssetInfo.Humidity = thmeter.Humidity.value

if (thmeter.Temperature.value>-15) {
   AssetInfo.Status = "CRITIQUE"
} else if (thmeter.Temperature.value>-19) {
   AssetInfo.Status="A RISQUE"
} else {
   AssetInfo.Status="NORMAL"
}

select * from Asset as myasset
{
            if (myasset.SensorId == AssetInfo.SensorId)
   {
      AssetInfo.Name = myasset.Name
                     upsert Asset (AssetInfo)

            }
}

return AssetInfo
```

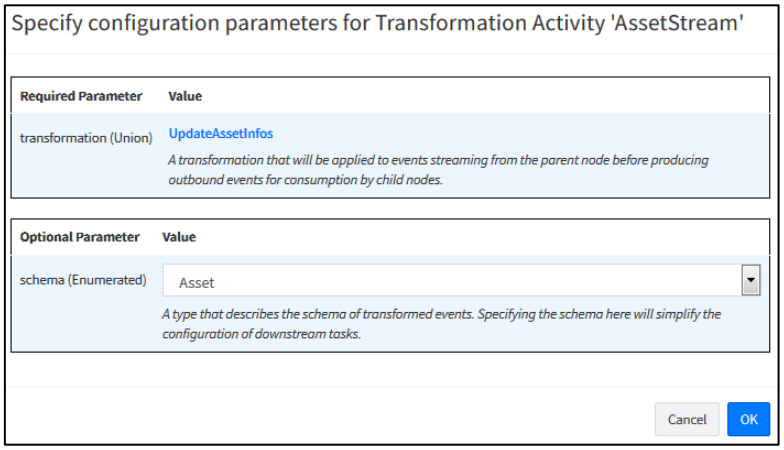
D. Create the AssetStream

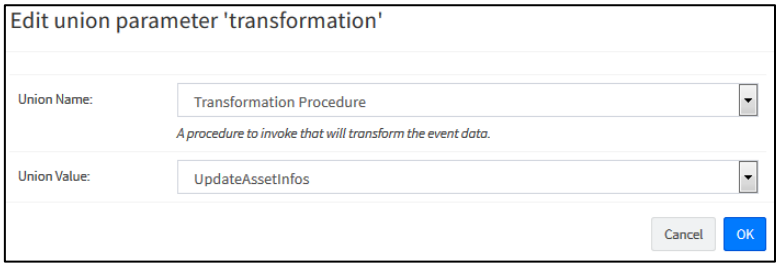
The Asset Stream is an enriched stream of TempStream that has additional information, information of the assets (the Rooms).

In the Application, click right on TempStream and Link a New Task, Select Transformation as Activity Pattern and name it AssetStream.



Configure the Transformation Activity to call UpdateAssetInfos procedure

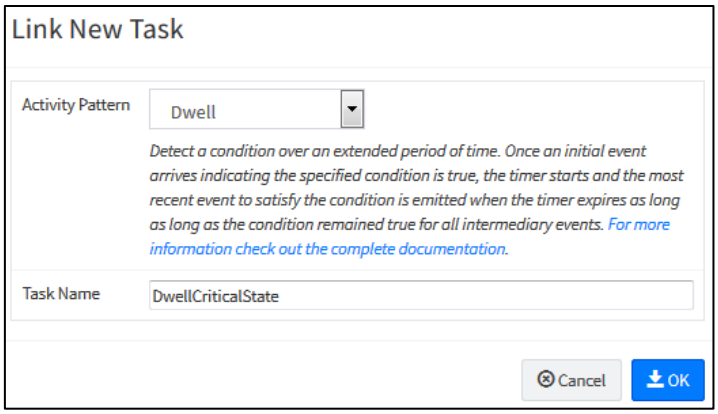




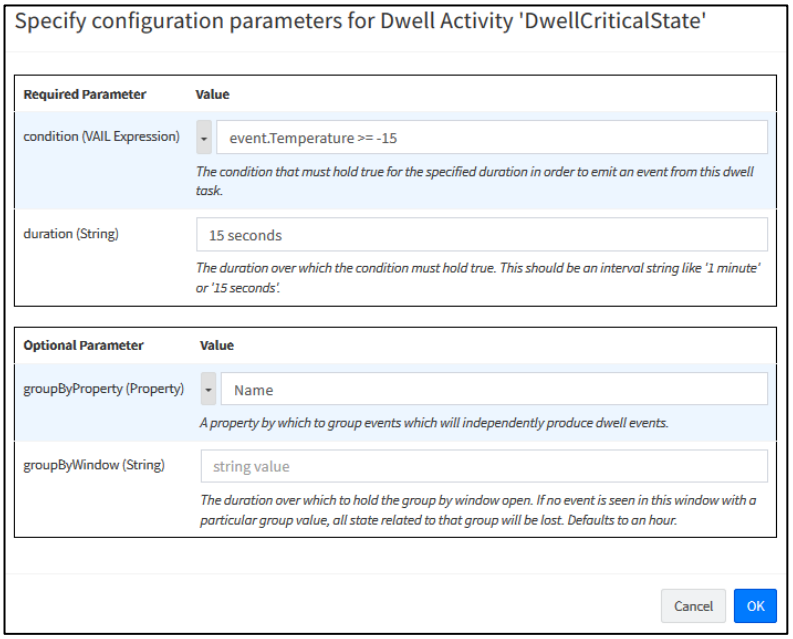
E. Add a Dwell Activity to detect a critical situation

A critical situation is detected if the temperature of a Room stays above -15°C during more than 15 seconds.

In the Application, click right on AssetStream and Link a New Task, Select Dwell as Activity Pattern and name it DwellCriticalState.

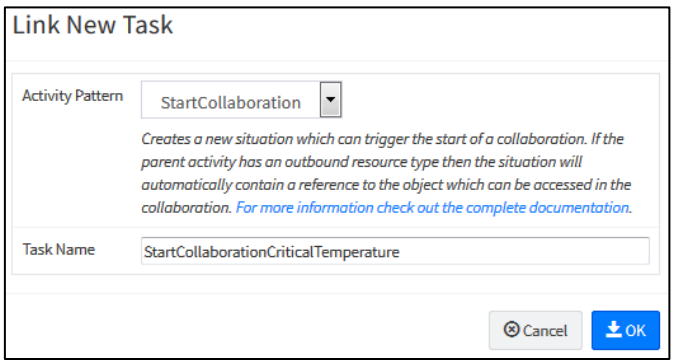


Select the DwellCriticalState, edit its configuration as followed:



F. Add a StartCollaboration Task

In the Application, click right on DwellCriticalState and Link a New Task, Select StartCollaboration and name it StartCollaborationCriticalSituation.

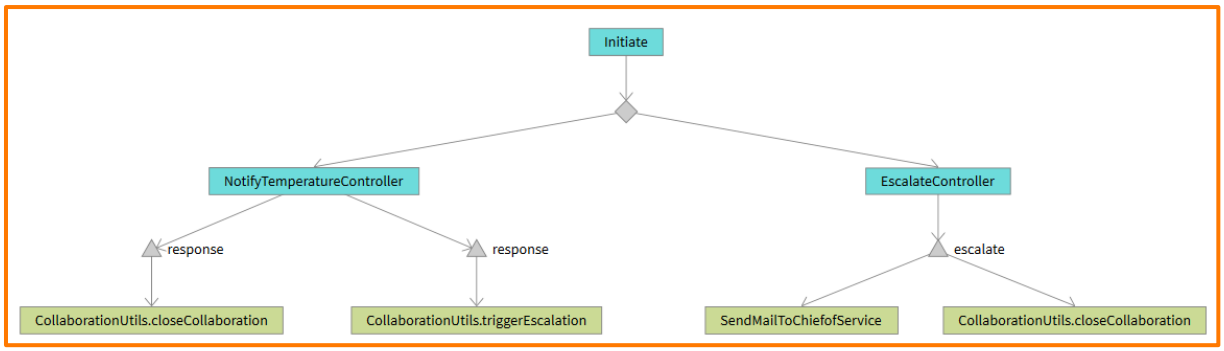


Edit configurations of StartCollaborationCriticalSituation, and Select “Create New Collaboration Type”, set the optional parameters as below , set OK and **save the App**. A collaboration called StartCollaborationCriticalSituation is created.

Optional Parameter	Value
entityConstraint (VAIL Expression)	Name == event.Name *A boolean expression that identifies a unique instance of the entityType. For instance, if the entityType has a natural key on a property called sensorId, an entityConstraint might be 'sensorId == event.id'.*
entityType (Enumerated)	Asset *The type of the entity that should be bound to this situation and any downstream collaboration entity roles.*

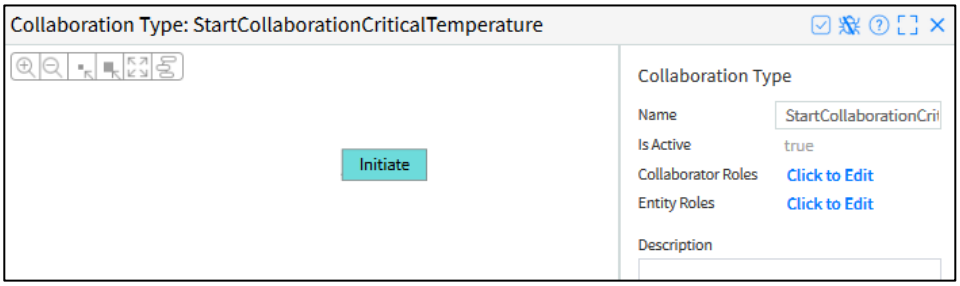
6. Define the Collaboration

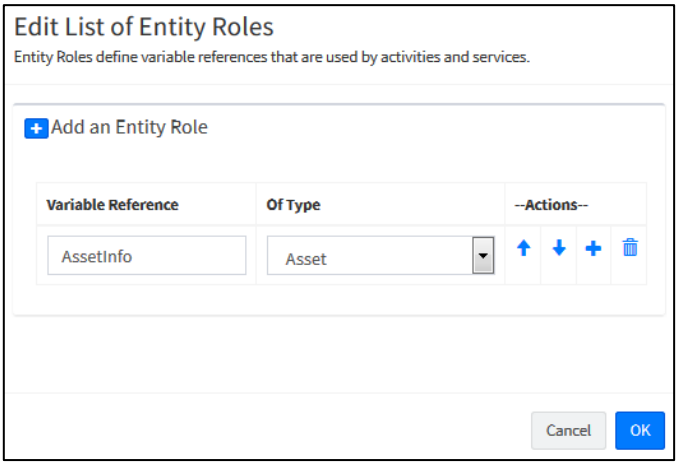
Collaboration Graph



A. Edit Collaboration configuration

In the configuration, edit Entity Roles.
Add an Entity Role called AssetInfo of type Asset.

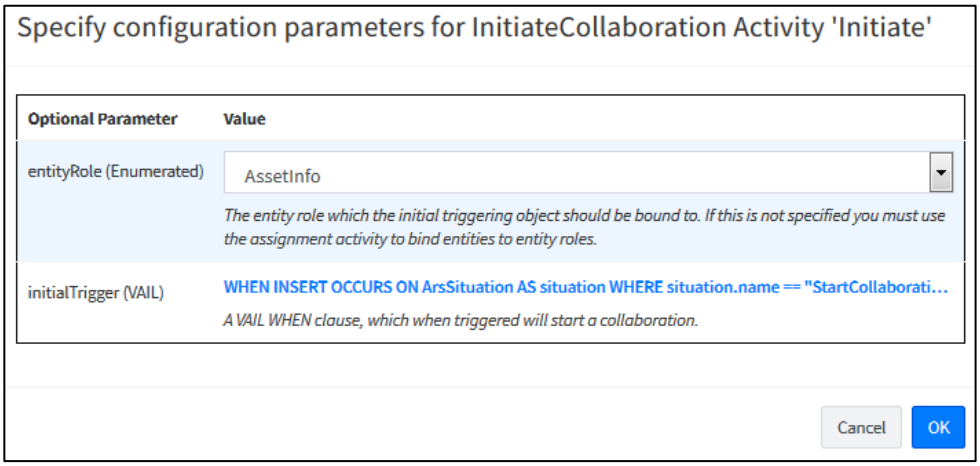




No collaborators roles are defined.

B. Initiate Collaboration configuration

Select the Initiate box, and edit the Configuration.
Select AssetInfo as entityRole.
The initialTrigger is already defined.



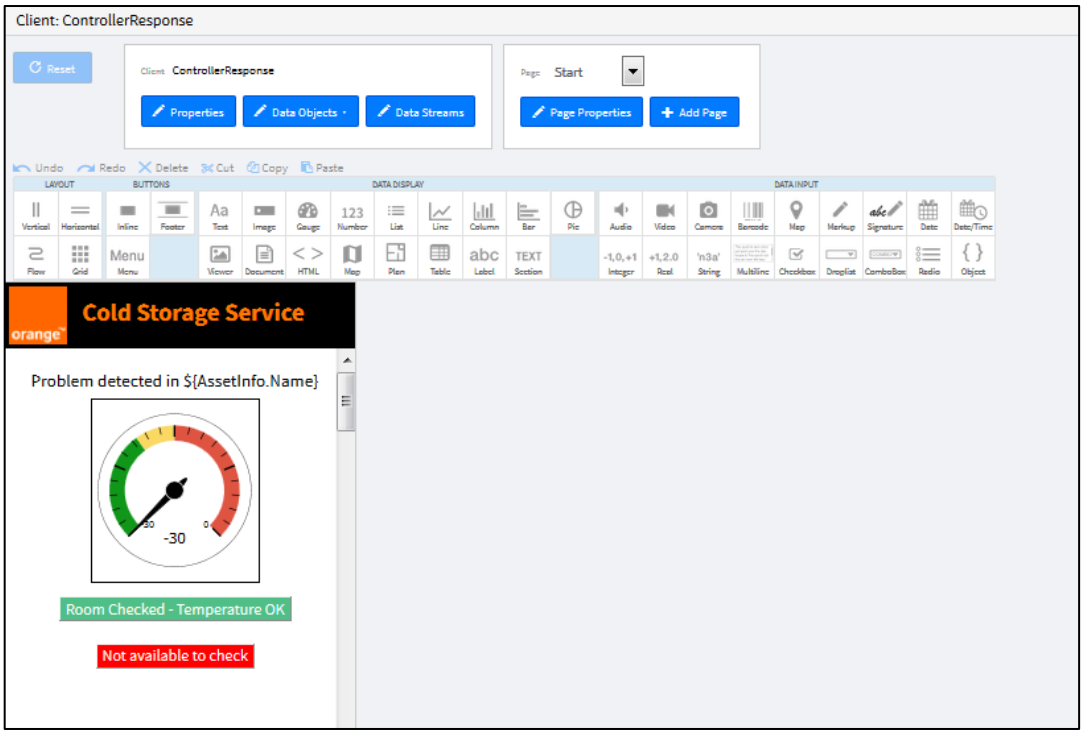
C. Notify a Controller

Once the critical situation has been detected, the first follow-up task is to notify a controller that may be able to correct the situation. The notification will be sent to a controller using the **Vantiq Mobile Application**. (Available in both iOS and Android versions).
In order to notify the controller, we will create a Vantiq Client that contains 2 responses buttons for the controller (“OK, I can fix the problem”/”NOK”).

D. Create the Mobile App to receive and respond to the notification

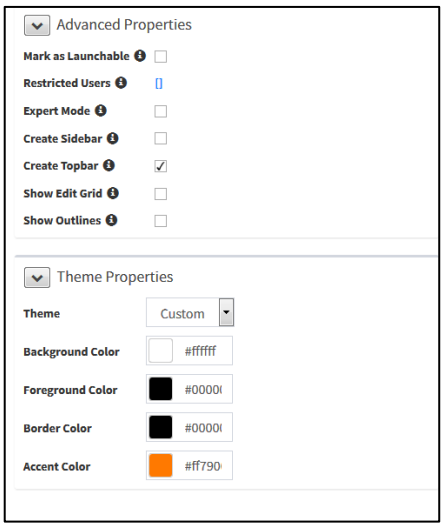
Use the **Add** button to select **Client.**
Use the **New Client** button to create the “NotifyController” client, and choose “Design for mobile apps” Layout Type.

Use the OK button to display the Client Builder.



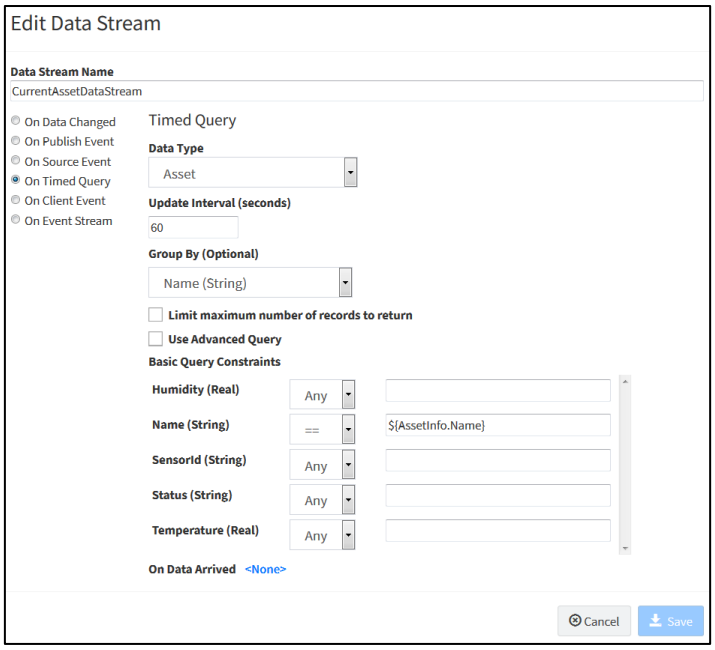
Configure Properties:

In properties, Advanced Properties, mark TopBar, and set the following Theme Properties:



Configure DataStream:

Add in Data Streams a Timed Query event **CurrentAssetDataStream** , that will get the information (temperature, localization) of the room where the critical situation has been detected.



Create the Topbar of the App

TopBar

On the client Builder Canvas, select the TopBar, and choose Black as Bakground Color.
Drag and drop **a Text** from the widget palette , place it in the TopBar.
Change the text to “Cold Storage Service”, and the Font Color to “Custom, #ff7900.

Add the Orange Logo

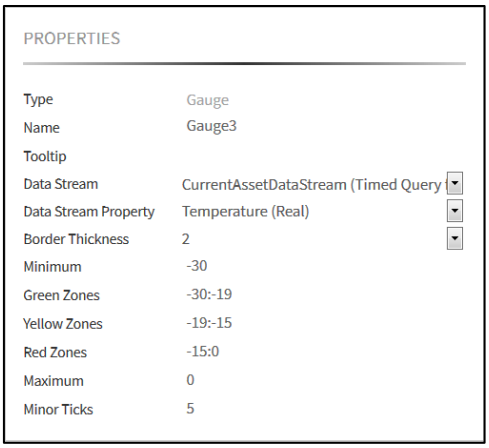
Add first the Orange Logo in the document of the workspace . In Show->Advanced-> Documents, add the OrangeLogo image with the following link public/images/OrangeLogo.png.

In the Topbar, drag and drop an **Image** on the left, and set the following link: https://dev.vantiq.com/ui/docs/NS/*”your workspace Name”*/images/OrangeLogo.png

Create the Start Page

Below the topbar, drag and drop a **Text** , and set “Problem detected in ${AssetInfo.Name}” ${AssetInfo.Name} refers to the Room Name where a critical situation has been detected.

Drag and drop a **Gauge** and set its properties as follows:



Drag and drop **two Inline buttons** from the widget palette to the Client Builder canvas. Click on the top Inline button to display its properties. Change the following property values:

- Button Label: Room Checked - Temperature OK
- Value: 0
- Button Label Font Size: 18
- Button Label Color: White
- Button Label Background Color : Custom #50be87

Click on the bottom Inline button to display its properties. Change the following property values:

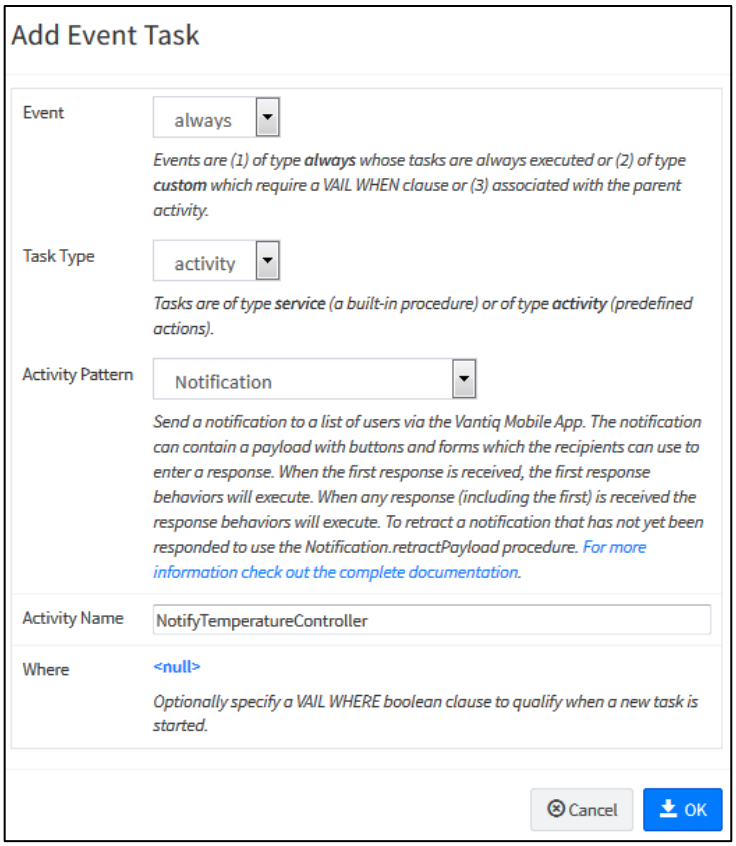
- Button Label: Not available to check
- Value: 1
- Button Label Font Size: 18
- Button Label Color: White
- Button Label Background Color : Custom #ff0000

Then use the **Save** button to save the *ControllerResponse* Client.

E. Define the Notification in the Collaboration

Click right on Initiate box in the Collaboration Type graph, and select **Add Event**. In the resulting Add Event Task dialog:

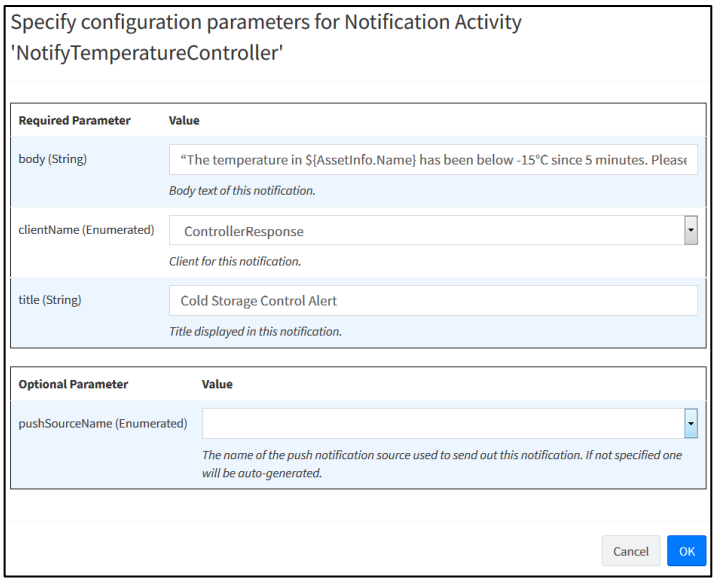
- the Event is always since we want the task to always execute,
- the Task Type is activity,
- the Activity Pattern is Notification,
- the Activity Name is **NotifyTemperatureController**,
- the Where clause is not specified (leave it <null>)



Use the **OK** button to save the Task parameters.

Click the NotifyTemperatureController rectangle to display its parameters in the right hand section.
The Configuration Parameters are:

- body: “The temperature in ${AssetInfo.Name} has been below -15°C since 5 minutes. Please check”. The body parameter is text that is sent in the notification to the VANTIQ mobile app. Note that we’ve added a reference to the room name, which is preceded by *${* and followed by *}*. When the notification is generated, the reference will be substituted with the room name in which the critical situation was detected.
- clientName: select *ControllerResponse* from the pull-down menu.
- title: “Cold Storage Control Alert”. The title parameter is text that is sent in the notification to the VANTIQ mobile app.
- pushSourceName: no value is necessary here.



Use the **OK** button to save the Configuration Parameters.

The Runtime Params corresponds to the user, the controller we want to send the notification. For the purpose of the tutorial, the notification is sent to your Vantiq Mobile App.
Set the users as an **expression**, and fill it with your Vantiq Account ID: Value = **["xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"]**

Use the **OK** button to save the users parameter then the **OK** button again to save the Runtime Parameter. Use the **Save** button in the top left corner of the IDE to save the project.

F. Add an Escalation Task

Once the controller has been notified, another follow-up task is to be defined, an escalation task in case the controller does not respond to the notification.
The escalation task will trigger, after a certain time, the sending of an email to the Chief of the service.

- **Source and procedure definition for mail sending**

The first step is to define a source for the mail server information.

In the project, Use the **Add** button to select **Source…**
Use the **New Source** button to create the "MailToChiefofService" source and define the following properties. This example uses a gmail server.
Source Type : EMAIL

Source Properties:
Server: smtp.gmail.com
Server Port: 465
Username: your gmail user account
Password: your gmail password

Click Save.

In the project, Use the **Add** button to select **Procedure…**
Use the **New Procedure** button to create the "SendMailToChiefofService" procedure, and define the following code :

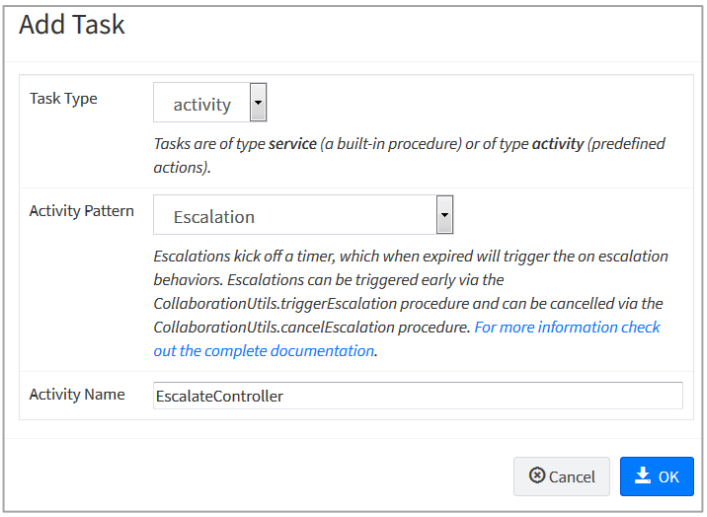
```
PROCEDURE SendMailToChiefofService(NoRoom String)

var mailbody = " Ms. Chief of Service,

A damaging problem on the cold storage service occured in "+NoRoom+ ".
Temperature has gone too high for too long time.
Controllers were not able to correct the situation.
Storage stock in this room is out of order. "

PUBLISH {text:mailbody} TO SOURCE MailToChiefofService
USING {
   from: "the mail address that emits the email",
   to: ["the mail address you want to send the email"],
   subject: " Service Cold Storage Rooms - Damaging Problem occured in "+NoRoom+". Stock is
out of order"
}
```

- **Add the Escalation task in the Collaboration**

To add an Escalation task associated with the triggering of the collaboration instance, right-click on the small diamond shape (which represents the *always* event) under the *Initiate* rectangle in the Collaboration Type graph, then select the **Add Task** menu item.

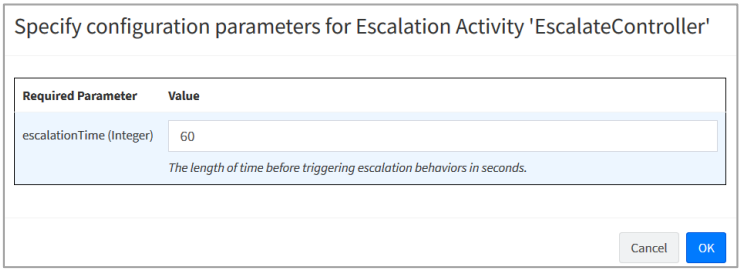
In the resulting Add Task dialog:

- the Task Type is activity,
- the Activity Pattern is Escalation,
- the Activity Name "EscalateController"

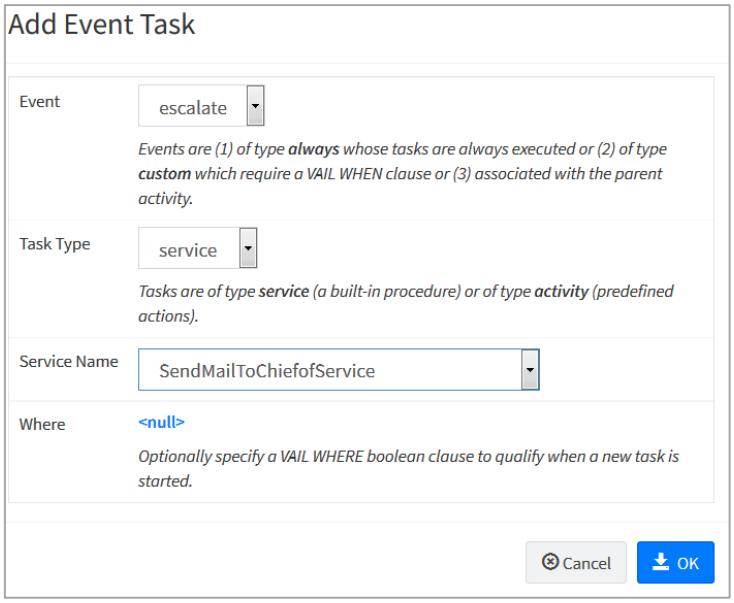


Click OK to Save.

Edit the Configuration of the EscalateController Escalation Activity and set the escalation Time to 60.

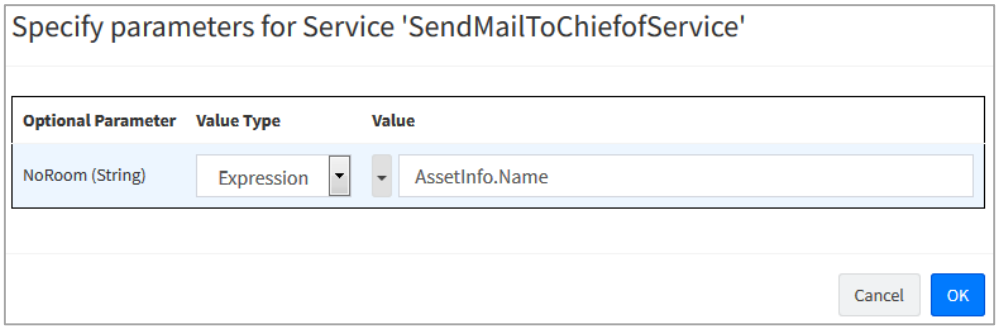


Click right on the EscalateController and Add Event to send Mail to Chief of Service.



Click OK to Save.

Select the SendMailToChiefofService box, and configure the Service Params: set the NoRoom which is the input parameter to the SendMailToChiefofService Procedure.

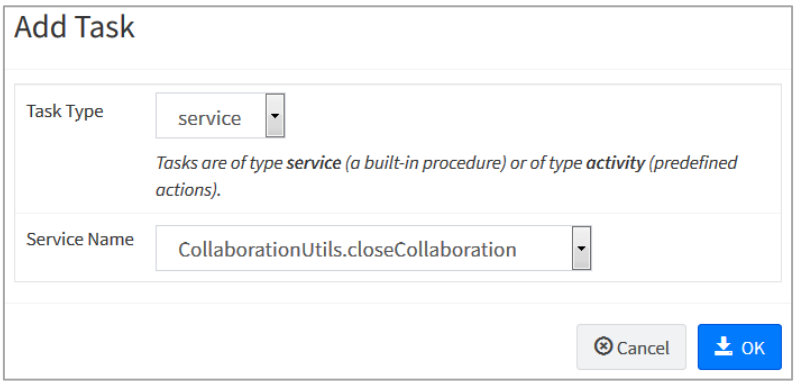


G. Close the Collaboration

The collaboration will be closed either when the Controller acknowledged the critical situation and correct it, or when the Chief of Service receives the mail that informs him the stock is out of order.

<u>Close Collaboration after Escalation:</u>
To close the collaboration, we'll add a VANTIQ built-in service, *CollaborationUtils.closeCollaboration*, to the *EscalateController* task.

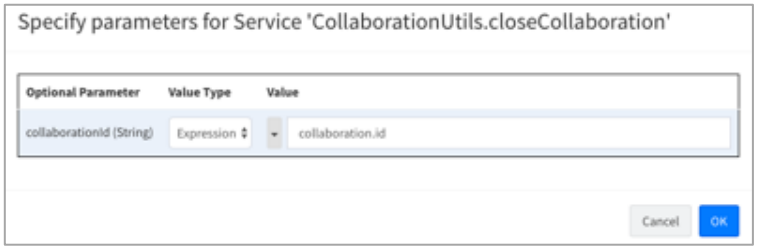
Right click on the triangle Escalate and **Add Task.**



Click the CollaborationUtils.closeCollaboration rectangle in the Collaboration Type graph to display its parameters in the right hand section. Change these parameters:

Change the Service Parameters by clicking the list next to the Service Params parameter name. The Services Parameter for this tutorial is:

- collaborationId: the collaboration system needs to know which collaboration to close. We use a built-in variable reference to specify the collaborationID. Select the **Expression** choice, then use the pull-down menu to select collaboration.id. These choices instruct the collaboration instance to use the ID associated with the current collaboration:



Use the **OK** button to save the Service Parameter.

Add Notification Response (Close Collaboration or Trigger Escalation)

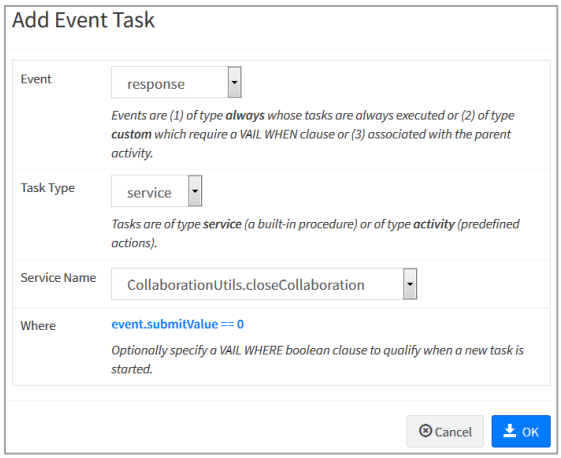
In the NotifyController Application, the controller can press the first button "**Room Checked - Temperature OK"** (Value= 0) , or the second one **"Not Available to Ckeck"**(Value = 1).

In the case the controller press the first button, the collaboration can be closed. To close the collaboration, right click on the NotifyController rectangle in the Collaboration Type graph, then select **Add Event** menu item. In the resulting Add Event Task dialog,

- the Event is response,
- the Task Type is service,
- the Service Name is CollaborationUtils.closeCollaboration,
- Since we want to close the collaboration only in the first case, we must add a Where clause. Click the <null> link. In the resulting Edit VAIL parameter dialog, we must specify a VAIL Boolean expression to indicate when the CollaborationUtils.closeCollaboration task is run. Recall that the value associated with Room Checked - Temperature OK is 0 (zero), so the Boolean expression must be event.submitValue == 0.

Use the Insert Reference Variable pull-down menu for hints on which variables make sense in the context of the response event:

Once the Where clause has been specified, the Add Event Task dialog looks like:



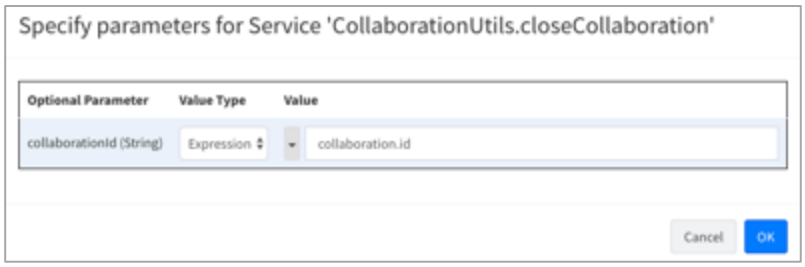
Use the **OK** button to save the Task parameters.

Click the CollaborationUtils.closeCollaboration rectangle in the Collaboration Type graph to display its parameters in the right hand section. Change these parameters:

Change the Service Parameters by clicking the list next to the **Service Params** parameter name. The Service Parameter for this tutorial is:

- collaborationId: collaborationId: the collaboration system needs to know which collaboration to close. As before, we use a built-in variable reference to specify the collaborationID. Select the **Expression** choice, then use the pull-down menu to select collaboration.id.

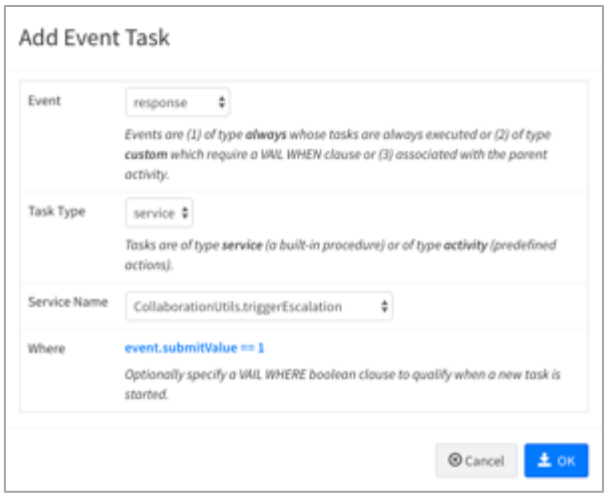
The parameters dialog should then look like this:



In the second case - **Not Available to Check** case, we immediately want to trigger the escalation that will notify the chief of service EscalateController. To trigger the escalation, right-click on the NotifyController rectangle in the Collaboration Type graph, then select the **Add Event** menu item. In the resulting Add Event Task dialog,

- the Event is response,
- the Task Type is service,
- the Service Name is CollaborationUtils.triggerEscalation,
- since we want to trigger the escalation only in the **Not Available to Check** case, we must add a Where clause. Click the <null> link. In the resulting Edit VAIL parameter dialog, we must specify a VAIL Boolean expression to indicate when the CollaborationUtils.cancelEscalation task is run. Recall that the value associated with **Not Available to Check** is 1 (one), so the Boolean expression must be event.submitValue == 1. Use the Insert Reference Variable pull-down menu for hints on which variables make sense in the context of the response event:

Once the Where clause has been specified, the Add Event Task dialog looks like:



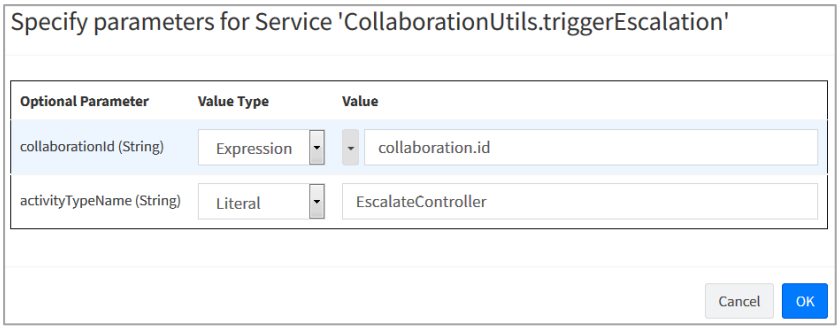
Use the **OK** button to save the Task parameters.

Click the CollaborationUtils.triggerEscalation rectangle in the Collaboration Type graph to display its parameters in the right hand section. Change these parameters:

Change the Service Parameters by clicking the list next to the **Service Params** parameter name. The Service Parameters for this tutorial are:

- collaborationId: the collaboration system needs to know which collaboration to close. As before, we use a built-in variable reference to specify the collaborationID. Select the **Expression** choice, then use the pull-down menu to select collaboration.id.
- activityTypeName: the name of the escalation task to trigger. For this tutorial, the value is EscalateController.

The parameters dialog should then look like this:

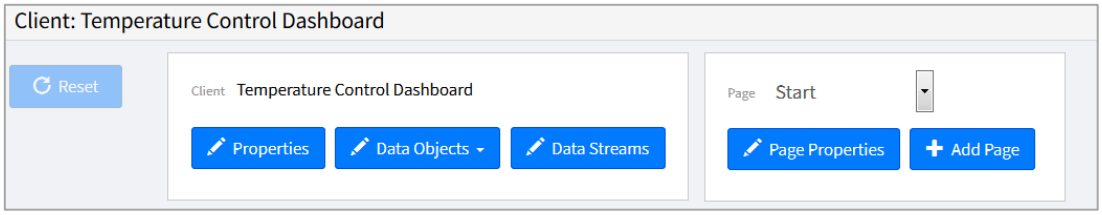


Use the **OK** button to save the Service Parameters.

7. Create a web Application to monitor the rooms temperature

Use the **Add** button, select **Client,** add a new one and name it LiveObjectsDataGenerator**.** Chose Design for browser Layout, and click OK.

A. Client Configuration

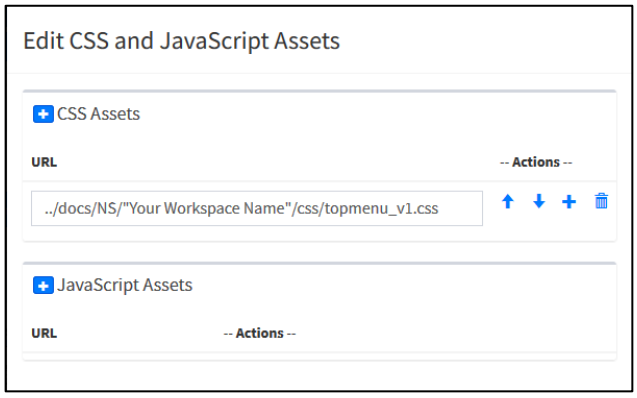


<u>Properties</u>

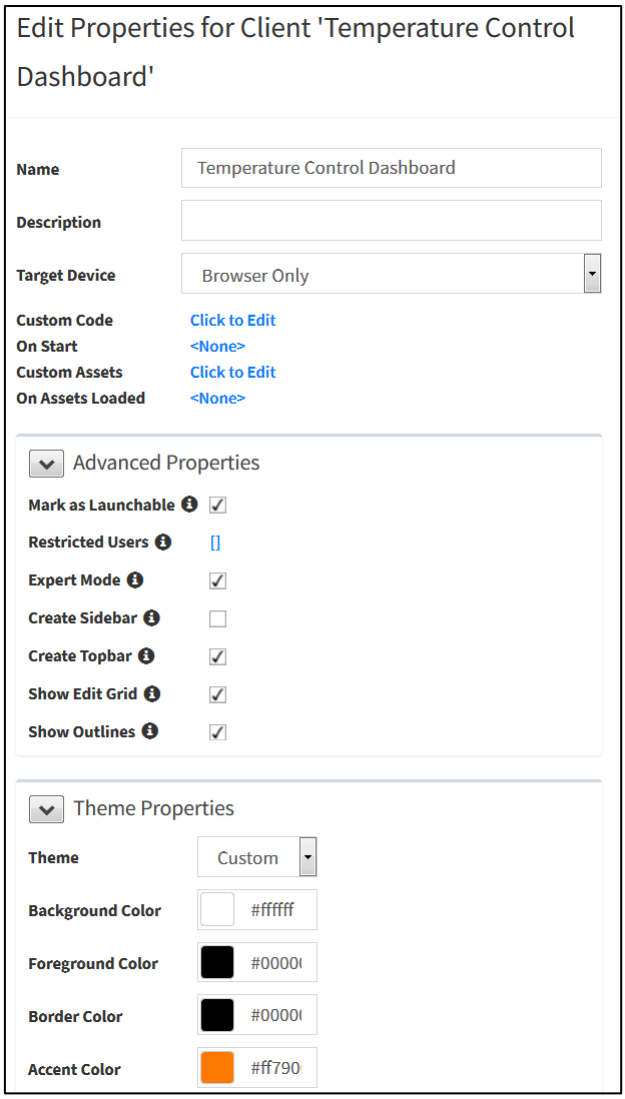
Add **Custom Code** to define the Orange header.

```
$('.navbarIcon').css("background-image", "url(https://dev.vantiq.com/ui/docs/NS/"your Namespace Name"/images/OrangeLogo.png)");
$(".navbarIcon").css("width", "60px");
$(".navbarIcon").css("height", "60px");
$('.navbarIcon').css("background-repeat", "no-repeat");

$("#rtcTitle" ).remove();
$(".main-header .logo").css({'height':'60px'}); // Hauteur du bandeau du haut
$(".navbar").css("background-color", "#000000"); // Couleur de la bare de navigation
$(".logo").css("background-color", "#000000"); // Couleur du fond sous le logo + hauteur du bandeau
$(".skin-blue .main-header .navbar .nav > li > a").css("color", "#444444");  // Theme de la bare du haut, couleur de police du nom du user et
Namespace VANTIQ, au format CSS,
}
```

Add **Custom Asset** to add a specific .css if wanted .
The .css needs to be previously added in the documents of the workspace (Show -> Advanced -> Documents) with the following link public/css/topmenu_V1.css.



Set **Mark as Launchable** and **Expert Mode, Create Topbar** in Advanced Properties**, and** Theme Properties as defined below.



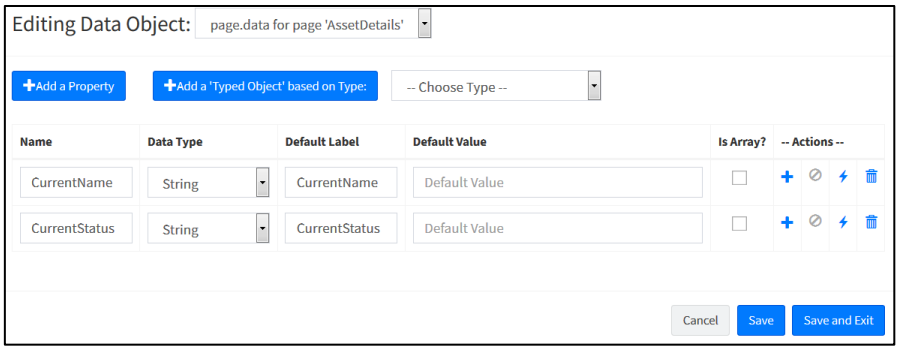
Add Page -> Add AssetDetails Page

The web application is composed of 2 pages, Start Page and AssetDetails Page.
On the Start Page, the user will have an overview of all the Rooms. He will be able to access to a room details on AssetDetails Page.
To add the AssetDetails Page, click on Add Page and Name it AssetDetails Page.

Data Objects

Add a Property in **Data Objects** / **page.data** for Page AssetDetails

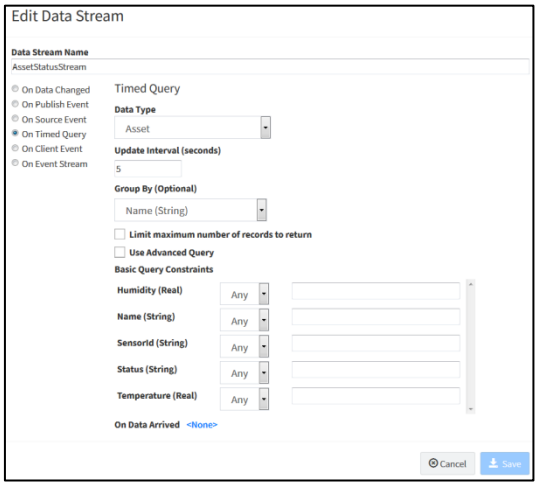
- a string CurrentName
- an integer CurrentStatus



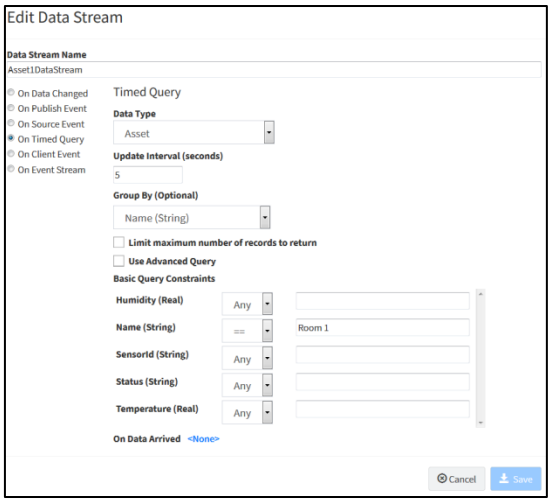
Data Streams

Add in Data Streams

- an **on Timed Query** event **AssetStatusStream** to get the information from Asset DataType (Query every 5 seconds)



- an **on Timed Query** event **CurrentAssetStream** to get the information from Asset DataType (Query every 5 seconds). This will be used for the AssetDetails page.
- an **on Timed Query** events **Asset1DataStream** to get the information of the Room 1 . (filtering on the Name) in Asset DataType.



- an **on Timed Query** events **Asset2DataStream** to get the information of the Room 2 . (filtering on the Name) in Asset DataType.
- an **on Timed Query** events **Asset3DataStream** to get the information of the Room 3 . (filtering on the Name) in Asset DataType.
- an **on Timed Query** events **Asset4DataStream** to get the information of the Room 4 . (filtering on the Name) in Asset DataType.
- an **on Timed Query** events **Asset5DataStream** to get the information of the Room 5 . (filtering on the Name) in Asset DataType.

Page Properties (Start Page Properties)

Edit **Properties** on **Page Start** and set the following code for On Client Start and On Start , to refresh data in Data Stream.

```
ds = client.getDataStreamByName("AssetStatusStream");
ds.restart();
ds = client.getDataStreamByName("Asset1DataStream");
ds.restart();
ds = client.getDataStreamByName("Asset2DataStream");
ds.restart();
ds = client.getDataStreamByName("Asset3DataStream");
ds.restart();
ds = client.getDataStreamByName("Asset4DataStream");
ds.restart();
ds = client.getDataStreamByName("Asset5DataStream");
ds.restart();
```

Page Properties (AssetDetails Page Properties)

Select **AssetDetails Page** and Edit the **PageProperties**.
Add the following code that enables to adapt the CurrentAssetStream DataStream to the selected Room .

```
this.data.CurrentName = parameters.Name;
this.data.CurrentStatus = parameters.Status;
// adjust the query
var ds = client.getDataStreamByName("CurrentAssetStream");
var p = new TimedQueryParameters();
p.whereClause = { Name: parameters.Name};
client.modifyTimedQuery(ds,p);
```

B. Create the TopBar

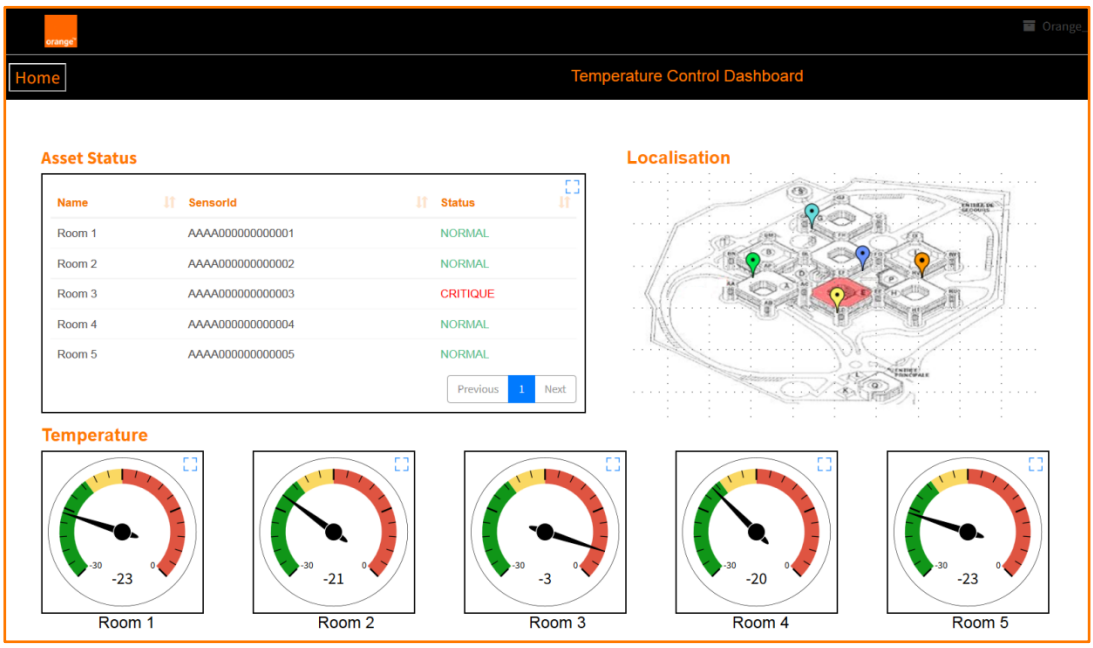
Select the Topbar and choose Black as Background Color.

Add **Static Text** box , with Text "Temperature Control Dashboard", Font Size 20, Font Color Custom #ff7900.

Add an Inline Button on the left,

- ButtonLabel : Home
- Button Label Font Size :22
- Button Label Color : Custom #ff7900
- Button Background Color : Black

C. Create Start Page with widget selection



Asset Status

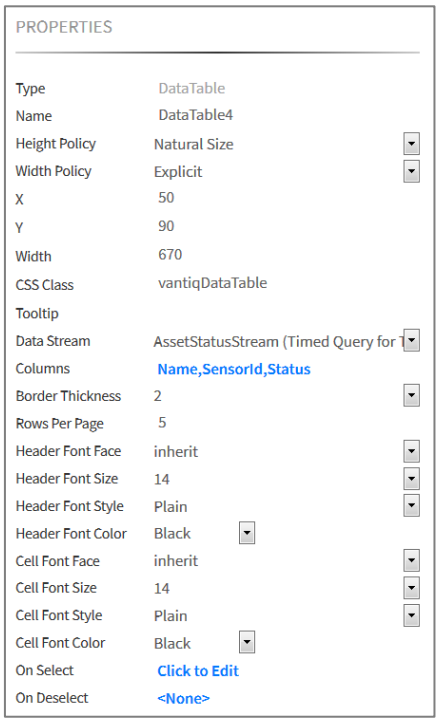
Add **Label** box , with Text "Asset Status", Font Size 22, Font Weight Bold , Font Color Custom #ff7900.

Add a **Data Table** to see the current Status of the different assets,

with DataStream set to "AssetStatusStream"

with Columns : Name, SensorId, Status

on Select : "client.goToPage("AssetDetails", extra);"



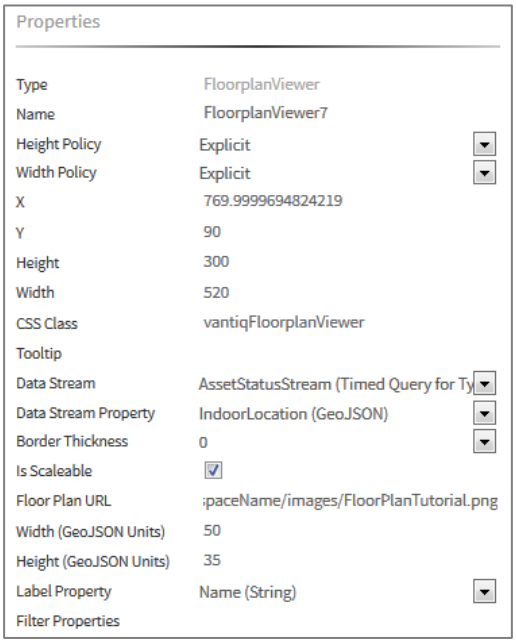
Localisation

Add **Label** box , with Text “Asset Localisation”, Font Size 22, Font Weight Bold , Font Color Custom #ff7900.

Add FloorPlanTutorial.pngimage in the image of your workspace. In Show->Advanced-> Documents, add the FloorPlanTutorial.png image with the following link public/images/FloorPlanTutorial.png.

Add a **FloorplanViewer** to see the localisation of the different assets, with

- Data Stream set to “AssetStatusStream”
- Data Stream Property set to “IndoorLocation”
- Floor Plan URL set to ../docs/NS/”your workspace name”/images/FloorPlanTutorial.png
- Width (GeoJSON Units) set to 50
- Height (GeoJSON Units) set to 35

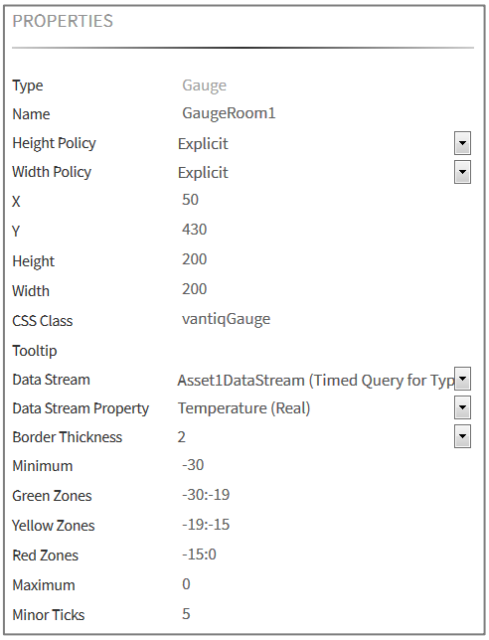


Temperatures

Add **a Text** box , with Text “Temperature”, Font Size 22, Font Weight Bold , Font Color Custom #ff7900.
Add a **Gauge** to show Temperature of Room 1, with

- Data Stream set to “Asset1DataStream”
- Data Stream Property set to “Temperature”
- Minimum set to -30

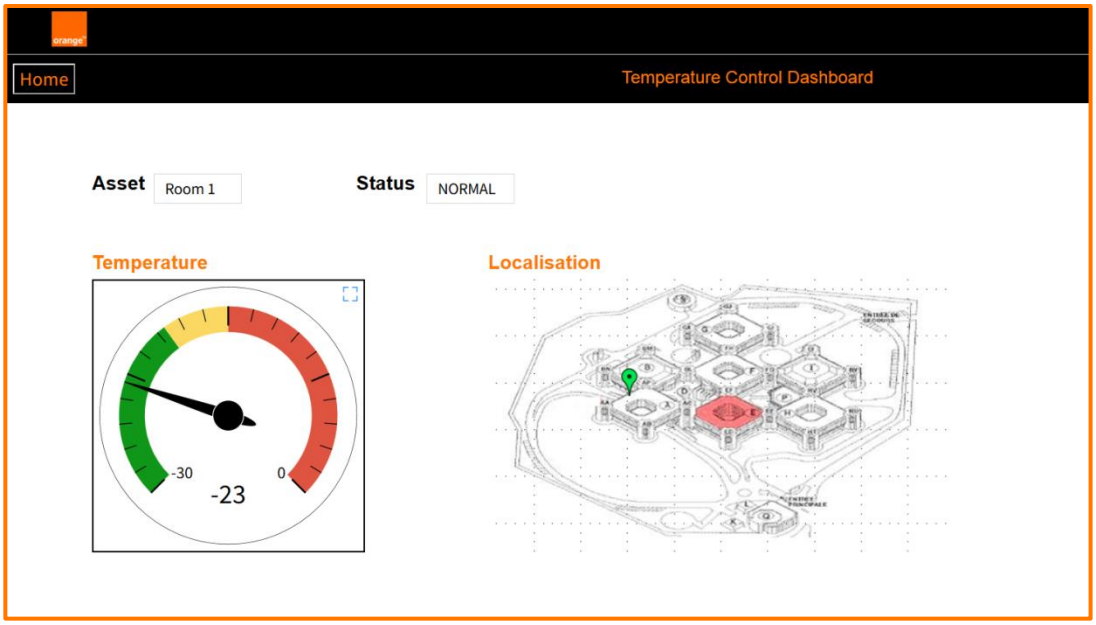
- Green Zones set to -30:-19
- Yellow Zones set to -19:-15
- Red Zones set to -15:0



Add a **Text box** with Text Room 1 below the first Gauge.

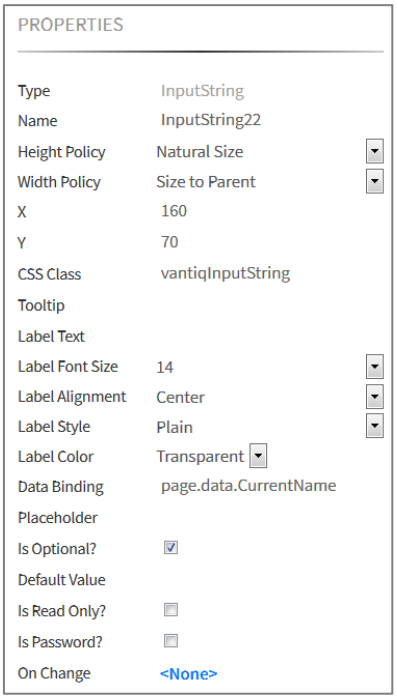
Add 4 other **Gauges** and **Text box** for Room2, Room3, Room4 and Room5.

D. Create the AssetDetails Page



Select **Page** AssetDetails.

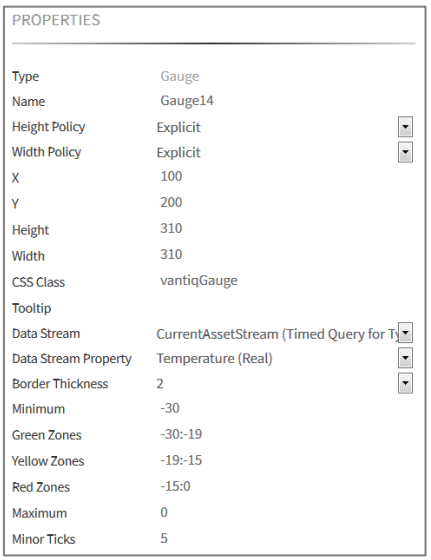
Add **Text** box , with Text "Asset", Font Size 22, Font Weight Bold , Font Color Black.
Add an **InputString**, with Data Binding set to page.data.CurrentName



Add **Text** box , with Text "Status", Font Size 22, Font Weight Bold , Font Color Black.
Add a **InputString**, with Data Binding set to page.data.CurrentStatus

Add **a Text** box , with Text "Temperature", Font Size 22, Font Weight Bold , Font Color Custom #ff7900.
Add a **Gauge** to show Temperature of selected Room, with

- Data Stream set to "CurrentAssetStream"
- Data Stream Property set to "Temperature"



Add **a Text** box , with Text “Localisation”, Font Size 22, Font Weight Bold , Font Color Custom #ff7900.

Add a **FloorplanViewer** to see the localisation of the selected asset, with

- Data Stream set to “CurrentAssetStream”
- Data Stream Property set to “IndoorLocation”
- Floor Plan URL set to ../docs/NS/”your workspace name”/images/FloorPlanTutorial.png