

## 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.

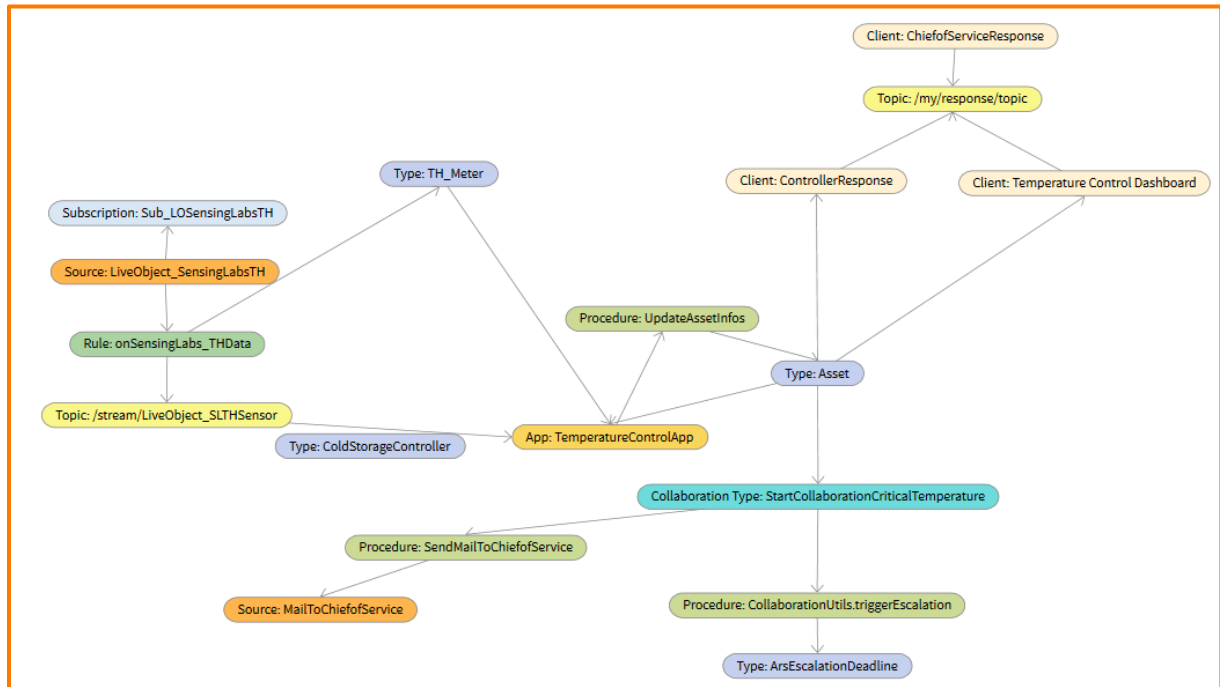
It is assumed that the developer has a working knowledge of the [VANTIQ IDE](#). It is recommended that a new developer completes the lessons in the [Introductory Tutorial](#) before starting this tutorial.

## Summary Table

1. Introduction.....	3
2. Create a new Project .....	4
3. Create Data Types.....	5
A. Create a new data Type TH_Meter .....	5
B. Record 5 sensors TH_Meter entries .....	5
C. Create a new DataType Asset .....	5
D. Record 5 Rooms entries .....	6
4. Create the Source to get data from Live Objects.....	7
5. Create the Control Application .....	9
A. Create the App .....	9
B. Create the TempStream .....	9
C. Create a procedure that transforms TH_Meter data in Asset data.....	10
D. Create the AssetStream .....	10
E. Add a Dwell Activity to detect a critical situation .....	11
F. Add a StartCollaboration Task .....	12
6. Define the Collaboration .....	14
A. Edit Collaboration configuration .....	14
B. Initiate Collaboration configuration .....	15
C. Notify a Controller .....	15
D. Create the Mobile App to receive and respond to the notification.....	15
E. Define the Notification in the Collaboration.....	18
F. Add an Escalation Task.....	20
G. Close the Collaboration.....	23
7. Create a web Application to monitor the rooms temperature .....	28
A. Client Configuration .....	28
B. Create the TopBar .....	32
C. Create Start Page with widget selection .....	33
D. Create the AssetDetails Page .....	35

## 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.

The screenshot shows a window titled "Type: TH\_Meter". At the top, there are two buttons: "+ Add New Record" and "Show All Records". Below these is a section titled "Properties" containing a table with the following data:

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:

AAAA00000000000001

AAAA00000000000002

AAAA00000000000003

AAAA00000000000004

AAAA00000000000005

The screenshot shows a window titled "Find Record Results: All Objects of type TH\_Meter". It includes a pagination bar with a refresh icon, navigation arrows, a page number "1", and a count "(1 - 5 of 5)". There are checkboxes for "Auto Refresh" and "Display Results as JSON". Below is a table with 5 rows of data:

<input type="checkbox"/>	Humidity	SensorId	Temperature	Time	--Actions--
<input type="checkbox"/>		AAAA00000000000001			
<input type="checkbox"/>		AAAA00000000000002			
<input type="checkbox"/>		AAAA00000000000003			
<input type="checkbox"/>		AAAA00000000000004			
<input type="checkbox"/>		AAAA00000000000005			

#### 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.

<input type="checkbox"/>	Humidity	IndoorLocation	Name	SensorData	SensorId	Status	Temperature	--Actions--
<input type="checkbox"/>	0	Lon 15, Lat 15	Room 1		AAAA0000000000000001	0		
<input type="checkbox"/>	0	Lon 20, Lat 25	Room 2		AAAA0000000000000002	0		
<input type="checkbox"/>	0	Lon 14, Lat 28	Room 3		AAAA0000000000000003	0		
<input type="checkbox"/>	0	Lon 8, Lat 22	Room 4		AAAA0000000000000004	0		
<input type="checkbox"/>	0	Lon 15, Lat 35	Room 5		AAAA0000000000000005	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

Seurveur URI : mqtt://liveobjects.orange-business.com:1883

The screenshot shows the 'Editing Source 'LiveObject\_SensingLabsTH'' interface. At the top, there are buttons for 'Save (Ctrl-S)', 'Reset', and 'Test Data Receipt'. Below these, the 'Source Type' is set to 'MQTT' and the 'Activation Constraint' is empty. The 'Source Properties' section contains a table with the following properties and values:

Property	Value
Keep Active?	<input checked="" type="checkbox"/>
Content Type	application/json
Username	json+bridge
Password	.....
Keep-Alive Interval (secs)	15
Connection Timeout (secs)	15
Maximum Inflight	10
Clean Session	<input checked="" type="checkbox"/>
Automatic Reconnect	<input checked="" type="checkbox"/>
Message Type	LiveObjectsDataMessage

On the right side, there are two sections: 'Server URIs' and 'Topics'. The 'Server URIs' section contains a table with one entry:

Server URI	--Actions--
mqtt://liveobjects.orange-business.com:1883	↑ ↓ + ✎ 🗑

The 'Topics' section contains a table with one entry:

Topic	--Actions--
fifo/vantiqFifo	↑ ↓ + ✎ 🗑

## 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

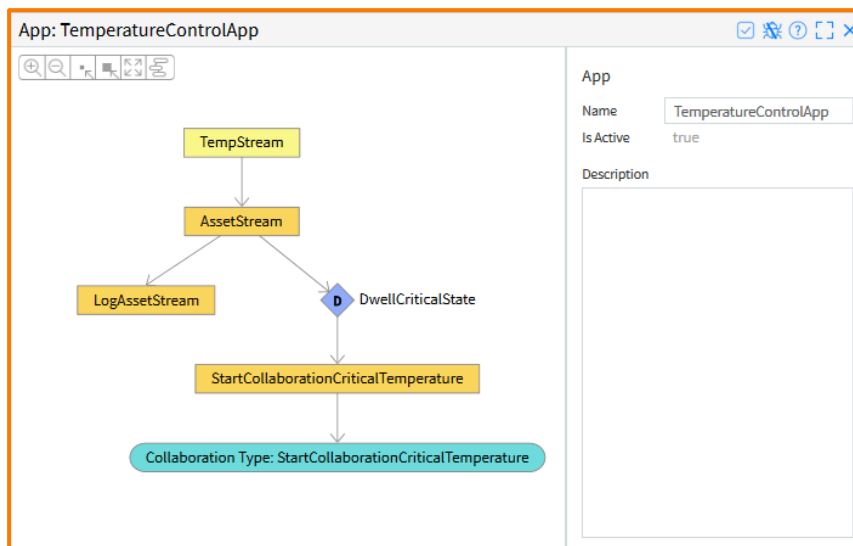
log.info("newSLSensor.Time" +stringify(newSLSensor.Time) )
log.info("newSLSensor.SensorId" +stringify(newSLSensor.SensorId) )
log.info("newSLSensor.Temperature" +stringify(newSLSensor.Temperature) )

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.

Specify configuration parameters for EventStream Activity 'TempStream'	
Required Parameter	Value
inboundResource (Enumerated)	topics <small>The resource on which the trigger is defined. Can currently be one of types, sources, or topics.</small>
inboundResourceId (String)	/stream/LiveObject_SLTHSensor <small>The specific resource on which the trigger is defined. For example MyType, MySource, or "/my/topic".</small>
Optional Parameter	Value
condition (VAIL Expression)	VAIL Expression <small>An optional conditional expression which will further restrict the trigger.</small>
op (Enumerated)	 <small>The op to trigger on. Only used when inboundResource is types, in which case the op can be INSERT, UPDATE or BOTH.</small>
schema (Enumerated)	TH_Meter <small>A type that describes the schema of inbound events. Specifying the schema here will simplify the configuration of downstream tasks.</small>

### 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"
}

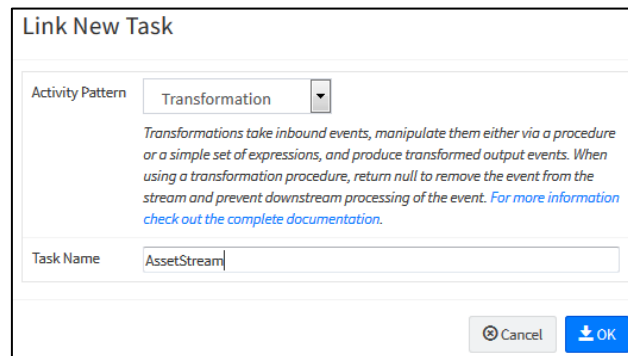
select * from Asset as myasset
{
  if (myasset.SensorId == AssetInfo.SensorId)
  {
    AssetInfo.Name = myasset.Name
    upsert Asset (AssetInfo)
  }
}

return AssetInfo
```

### D. Create the AssetStream

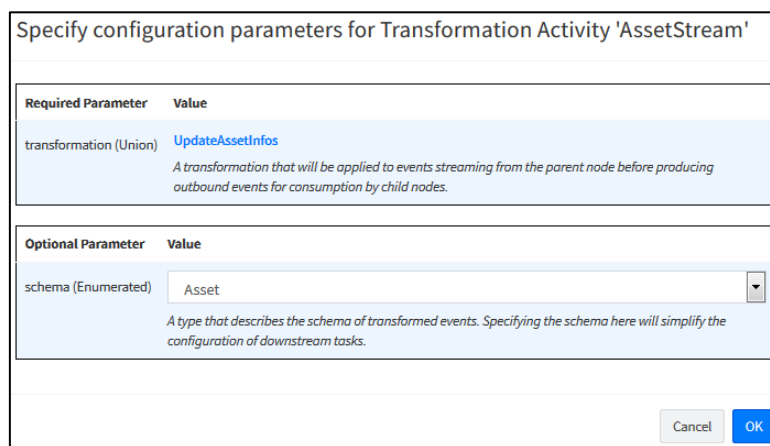
The Asset Stream is an enriched stream of TempStream that has additional information, information of the assets (the Rooms).

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

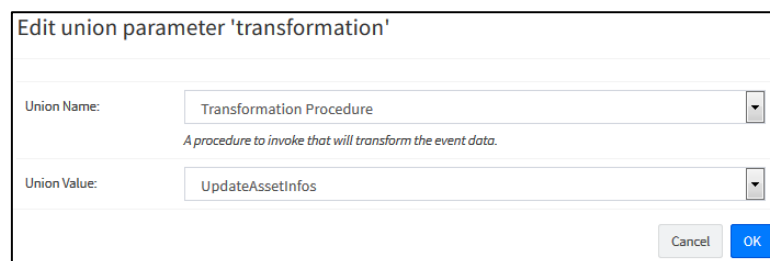


The 'Link New Task' dialog box has a title bar 'Link New Task'. It contains an 'Activity Pattern' dropdown menu set to 'Transformation'. Below this is a text description: 'Transformations take inbound events, manipulate them either via a procedure or a simple set of expressions, and produce transformed output events. When using a transformation procedure, return null to remove the event from the stream and prevent downstream processing of the event. [For more information check out the complete documentation.](#)'. There is a 'Task Name' text field containing 'AssetStream'. At the bottom right are 'Cancel' and 'OK' buttons.

Configure the Transformation Activity to call UpdateAssetInfos procedure



The 'Specify configuration parameters for Transformation Activity 'AssetStream'' dialog box has a title bar with the same text. It contains two sections. The first section, 'Required Parameter', has a table with two columns: 'Required Parameter' and 'Value'. It lists 'transformation (Union)' with the value 'UpdateAssetInfos' and a description: 'A transformation that will be applied to events streaming from the parent node before producing outbound events for consumption by child nodes.' The second section, 'Optional Parameter', has a table with two columns: 'Optional Parameter' and 'Value'. It lists 'schema (Enumerated)' with the value 'Asset' and a description: 'A type that describes the schema of transformed events. Specifying the schema here will simplify the configuration of downstream tasks.' At the bottom right are 'Cancel' and 'OK' buttons.



The 'Edit union parameter 'transformation'' dialog box has a title bar with the same text. It contains two text fields. The first is 'Union Name:' with a dropdown menu set to 'Transformation Procedure' and a description: 'A procedure to invoke that will transform the event data.' The second is 'Union Value:' with a dropdown menu set to 'UpdateAssetInfos'. At the bottom right are 'Cancel' and 'OK' buttons.

### 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.

Link New Task

Activity Pattern

Dwell

Detect a condition over an extended period of time. Once an initial event arrives indicating the specified condition is true, the timer starts and the most recent event to satisfy the condition is emitted when the timer expires as long as the condition remained true for all intermediary events. For more information check out the complete documentation.

Task Name

DwellCriticalState

Cancel

OK

Select the DwellCriticalState, edit its configuration as followed:

Specify configuration parameters for Dwell Activity 'DwellCriticalState'

Required Parameter	Value
condition (VAIL Expression)	<div>event.Temperature &gt;= -15</div> <div>The condition that must hold true for the specified duration in order to emit an event from this dwell task.</div>
duration (String)	<div>15 seconds</div> <div>The duration over which the condition must hold true. This should be an interval string like '1 minute' or '15 seconds'.</div>

Optional Parameter	Value
groupByProperty (Property)	<div>Name</div> <div>A property by which to group events which will independently produce dwell events.</div>
groupByWindow (String)	<div>string value</div> <div>The duration over which to hold the group by window open. If no event is seen in this window with a particular group value, all state related to that group will be lost. Defaults to an hour.</div>

Cancel

OK

## F. Add a StartCollaboration Task

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

Link New Task

Activity Pattern

StartCollaboration

Creates a new situation which can trigger the start of a collaboration. If the parent activity has an outbound resource type then the situation will automatically contain a reference to the object which can be accessed in the collaboration. For more information check out the complete documentation.

Task Name

StartCollaborationCriticalTemperature

Cancel

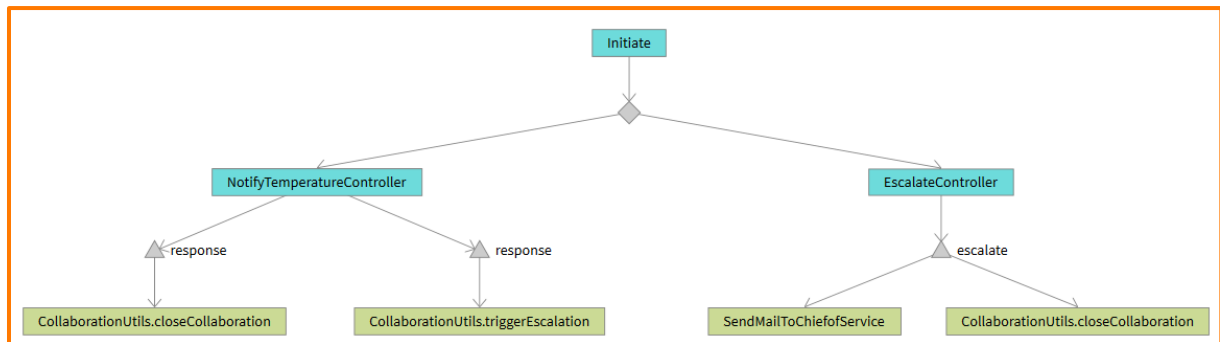
OK

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)	<div><div>▼</div><div>Name == event.Name</div></div> <p><i>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'.</i></p>
entityType (Enumerated)	<div><div>Asset</div><div>▼</div></div> <p><i>The type of the entity that should be bound to this situation and any downstream collaboration entity roles.</i></p>

## 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.

Collaboration Type: StartCollaborationCriticalTemperature

Initiate

Collaboration Type

Name: StartCollaborationCri

Is Active: true

Collaborator Roles: [Click to Edit](#)

Entity Roles: [Click to Edit](#)

Description:

Edit List of Entity Roles

Entity Roles define variable references that are used by activities and services.

+ Add an Entity Role

Variable Reference	Of Type	--Actions--
AssetInfo	Asset	↑ ↓ + 🗑️

Cancel OK

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.

Specify configuration parameters for InitiateCollaboration Activity 'Initiate'

Optional Parameter	Value
entityRole (Enumerated)	AssetInfo <small>The entity role which the initial triggering object should be bound to. If this is not specified you must use the assignment activity to bind entities to entity roles.</small>
initialTrigger (VAIL)	WHEN INSERT OCCURS ON ArsSituation AS situation WHERE situation.name == "StartCollaborati... <small>A VAIL WHEN clause, which when triggered will start a collaboration.</small>

Cancel OK

## 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).

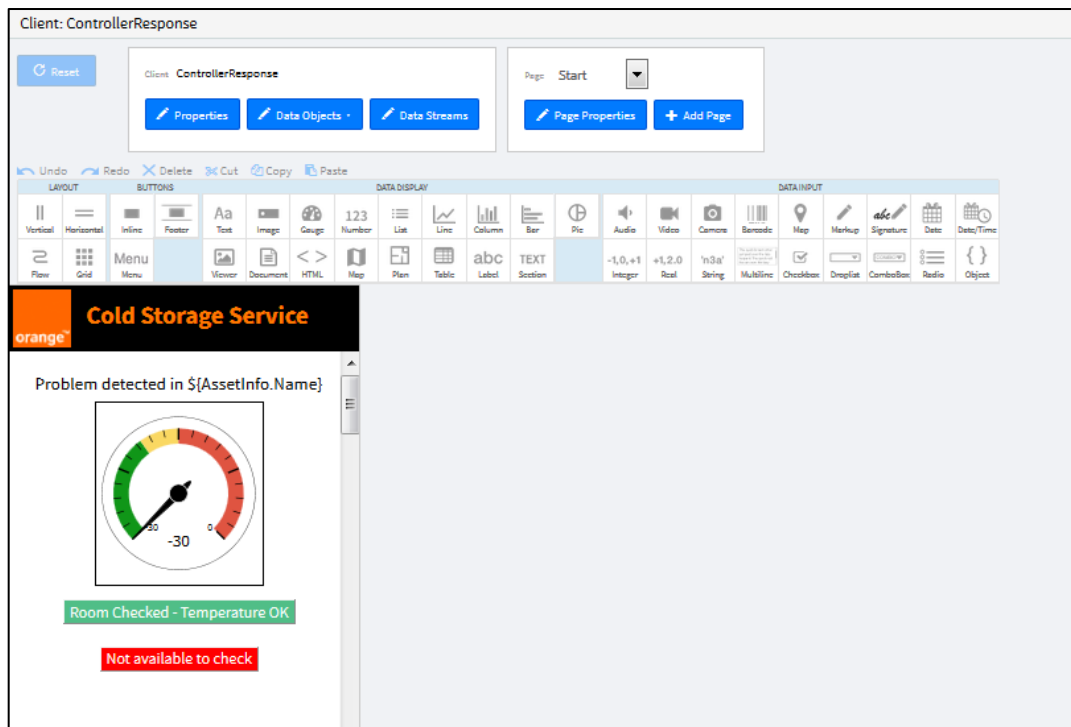
In order to notify the controller, we will create a Vantiq Client that contains 2 responses buttons for the controller ( "OK, I can fix the problem"/"NOK").

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

Use the **Add** button to select **Client**.

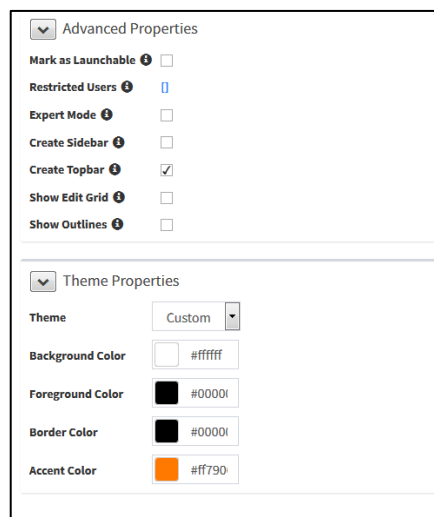
Use the **New Client** button to create the "NotifyController" client, and choose "Design for mobile apps" Layout Type.

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 Background 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:

PROPERTIES	
Type	Gauge
Name	Gauge3
Tooltip	
Data Stream	CurrentAssetDataStream (Timed Query) <input type="button" value="v"/>
Data Stream Property	Temperature (Real) <input type="button" value="v"/>
Border Thickness	2 <input type="button" value="v"/>
Minimum	-30
Green Zones	-30:-19
Yellow Zones	-19:-15
Red Zones	-15:0
Maximum	0
Minor Ticks	5

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>)

### Add Event Task

Event

always

▼

Events are (1) of type *always* whose tasks are always executed or (2) of type *custom* which require a VAIL WHEN clause or (3) associated with the parent activity.

Task Type

activity

▼

Tasks are of type *service* (a built-in procedure) or of type *activity* (predefined actions).

Activity Pattern

Notification

▼

Send a notification to a list of users via the Vantiq Mobile App. The notification can contain a payload with buttons and forms which the recipients can use to enter a response. When the first response is received, the first response behaviors will execute. When any response (including the first) is received the response behaviors will execute. To retract a notification that has not yet been responded to use the `Notification.retractPayload` procedure. [For more information check out the complete documentation.](#)

Activity Name

NotifyTemperatureController

Where

<null>

Optionally specify a VAIL WHERE boolean clause to qualify when a new task is started.

Cancel

OK

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.

Specify configuration parameters for Notification Activity  
'NotifyTemperatureController'

Required Parameter	Value
body (String)	"The temperature in \${AssetInfo.Name} has been below -15°C since 5 minutes. Please <i>Body text of this notification.</i>
clientName (Enumerated)	ControllerResponse <i>Client for this notification.</i>
title (String)	Cold Storage Control Alert <i>Title displayed in this notification.</i>

Optional Parameter	Value
pushSourceName (Enumerated)	 <i>The name of the push notification source used to send out this notification. If not specified one will be auto-generated.</i>

Cancel OK

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 occurred 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 occurred 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”

### Add Task

Task Type:

*Tasks are of type **service** (a built-in procedure) or of type **activity** (predefined actions).*

Activity Pattern:

*Escalations kick off a timer, which when expired will trigger the on escalation behaviors. Escalations can be triggered early via the `CollaborationUtils.triggerEscalation` procedure and can be cancelled via the `CollaborationUtils.cancelEscalation` procedure. [For more information check out the complete documentation.](#)*

Activity Name:

Cancel

Click OK to Save.

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

### Specify configuration parameters for Escalation Activity 'EscalateController'

Required Parameter	Value
escalationTime (Integer)	<input type="text" value="60"/>

*The length of time before triggering escalation behaviors in seconds.*

Cancel

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

### Add Event Task

Event:

*Events are (1) of type **always** whose tasks are always executed or (2) of type **custom** which require a VAIL WHEN clause or (3) associated with the parent activity.*

Task Type:

*Tasks are of type **service** (a built-in procedure) or of type **activity** (predefined actions).*

Service Name:

Where:

*Optionally specify a VAIL WHERE boolean clause to qualify when a new task is started.*

Cancel

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.

Optional Parameter	Value Type	Value
NoRoom (String)	Expression	AssetInfo.Name

### 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.

#### **Close Collaboration after Escalation:**

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**.

**Add Task**

Task Type: service

*Tasks are of type service (a built-in procedure) or of type activity (predefined actions).*

Service Name: CollaborationUtils.closeCollaboration

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:

Optional Parameter	Value Type	Value
collaborationId (String)	Expression	collaboration.id

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:



**Edit VAIL parameter 'where'**

*(The Where parameter is a Boolean expression that determines if activities or services attached to the event are executed. Use the menu below to select references to variables that make sense in the context of this event.)*

-- Insert Reference Variable --

```
1 event.submitValue == 0
```

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

**Add Event Task**

Event:

*Events are (1) of type **always** whose tasks are always executed or (2) of type **custom** which require a VAIL WHEN clause or (3) associated with the parent activity.*

Task Type:

*Tasks are of type **service** (a built-in procedure) or of type **activity** (predefined actions).*

Service Name:

Where:

*Optionally specify a VAIL WHERE boolean clause to qualify when a new task is started.*

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:

Specify parameters for Service 'CollaborationUtils.closeCollaboration'

Optional Parameter	Value Type	Value
collaborationId (String)	Expression	collaboration.id

Cancel OK

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:

Where Clause

(The Where clause is a Boolean expression that determines if tasks attached to the event are executed. Use the menu below to select references to variables that make sense in the context of this event.)

-- Insert Reference Variable --

```
1 event.submitValue == 1
```

Cancel OK

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

### Add Event Task

Event

response

Events are (1) of type *always* whose tasks are always executed or (2) of type *custom* which require a *WAIL WHEN* clause or (3) associated with the parent activity.

Task Type

service

Tasks are of type *service* (a built-in procedure) or of type *activity* (predefined actions).

Service Name

CollaborationUtils.triggerEscalation

Where

event.submitValue == 1

Optionally specify a *WAIL WHERE* Boolean clause to qualify when a new task is started.

Cancel

OK

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:

### Specify parameters for Service 'CollaborationUtils.triggerEscalation'

Optional Parameter	Value Type	Value
collaborationId (String)	Expression	collaboration.id
activityTypeName (String)	Literal	EscalateController

Cancel

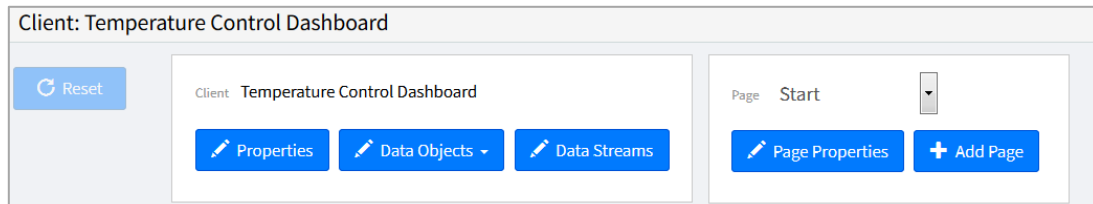
OK

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



### Properties

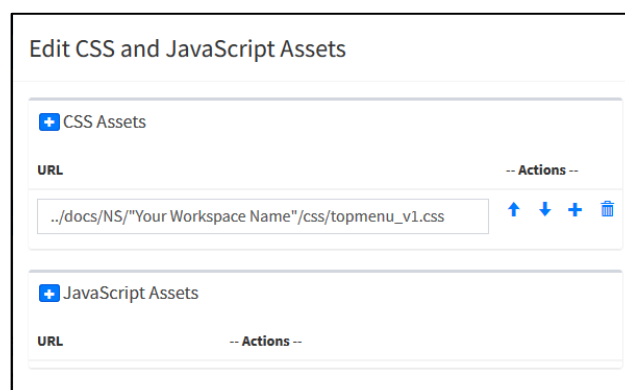
Add **Custom Code** to define the Orange header.

```
$('.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.

Edit Properties for Client 'Temperature Control Dashboard'

Name
Temperature Control Dashboard

Description

Target Device
Browser Only

Custom Code
[Click to Edit](#)

On Start
<None>

Custom Assets
[Click to Edit](#)

On Assets Loaded
<None>

Advanced Properties

Mark as Launchable
☒

Restricted Users
[]

Expert Mode
☒

Create Sidebar
☐

Create Topbar
☒

Show Edit Grid
☒

Show Outlines
☒

Theme Properties

Theme
Custom

Background Color
#ffffff

Foreground Color
#000000

Border Color
#000000

Accent Color
#ff7900

### 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

Editing Data Object: page.data for page 'AssetDetails'

[+ Add a Property](#)
[+ Add a 'Typed Object' based on Type:](#)
-- Choose Type --

Name	Data Type	Default Label	Default Value	Is Array?	-- Actions --
CurrentName	String	CurrentName	Default Value	<input type="checkbox"/>	<a href="#">+</a> <a href="#">⌂</a> <a href="#">⚡</a> <a href="#">🗑</a>
CurrentStatus	String	CurrentStatus	Default Value	<input type="checkbox"/>	<a href="#">+</a> <a href="#">⌂</a> <a href="#">⚡</a> <a href="#">🗑</a>

[Cancel](#)
[Save](#)
[Save and Exit](#)

## Data Streams

### Add in Data Streams

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

Edit Data Stream

Data Stream Name  
AssetStatusStream

☐ On Data Changed  
☐ On Publish Event  
☐ On Source Event  
☒ On Timed Query  
☐ On Client Event  
☐ On Event Stream

Timed Query

Data Type  
Asset

Update Interval (seconds)  
5

Group By (Optional)  
Name (String)

☐ Limit maximum number of records to return  
☐ Use Advanced Query

Basic Query Constraints

Humidity (Real) Any

Name (String) Any

Sensorid (String) Any

Status (String) Any

Temperature (Real) Any

On Data Arrived <None>

[Cancel](#)
[Save](#)

- 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.

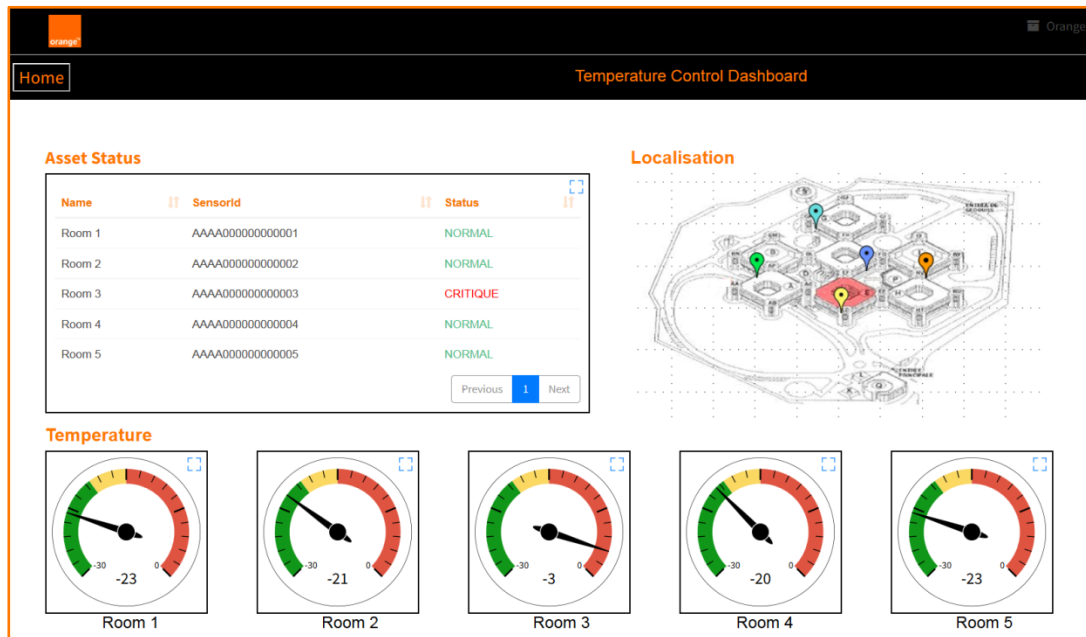
Add **Static Text** box , with Text “Temperature Control Dashboard”, Font Size 20, Font Color Custom #ff7900.

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);”

PROPERTIES

Type	DataTable
Name	DataTable4
Height Policy	Natural Size
Width Policy	Explicit
X	50
Y	90
Width	670
CSS Class	vantiqDataTable
Tooltip	
Data Stream	AssetStatusStream (Timed Query for
Columns	Name, SensorId, Status
Border Thickness	2
Rows Per Page	5
Header Font Face	inherit
Header Font Size	14
Header Font Style	Plain
Header Font Color	Black
Cell Font Face	inherit
Cell Font Size	14
Cell Font Style	Plain
Cell Font Color	Black
On Select	Click to Edit
On Deselect	<None>

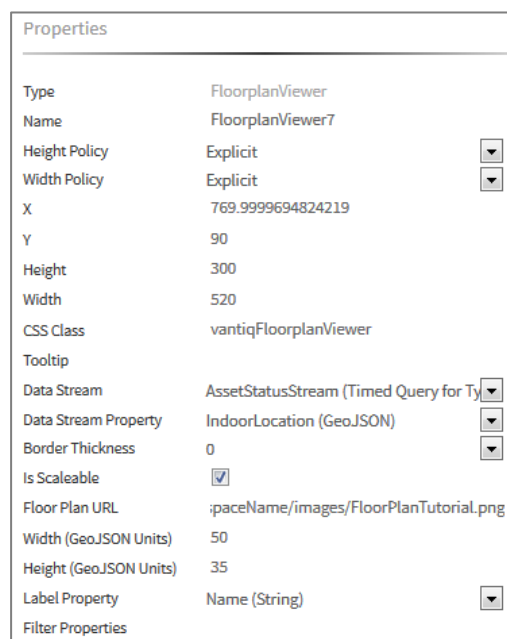
## Localisation

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

Add FloorPlanTutorial.png image 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



The screenshot shows the 'Properties' panel for a 'FloorplanViewer' widget. The panel lists various configuration options and their current values. The 'Data Stream' is set to 'AssetStatusStream (Timed Query for Ty...', 'Data Stream Property' is 'IndoorLocation (GeoJSON)', and 'Floor Plan URL' is 'spaceName/images/FloorPlanTutorial.png'. The 'Width (GeoJSON Units)' is 50 and 'Height (GeoJSON Units)' is 35. The 'Label Property' is 'Name (String)'.

Properties	
Type	FloorplanViewer
Name	FloorplanViewer7
Height Policy	Explicit
Width Policy	Explicit
X	769.9999694824219
Y	90
Height	300
Width	520
CSS Class	vantiqFloorplanViewer
Tooltip	
Data Stream	AssetStatusStream (Timed Query for Ty
Data Stream Property	IndoorLocation (GeoJSON)
Border Thickness	0
Is Scaleable	<input checked="" type="checkbox"/>
Floor Plan URL	spaceName/images/FloorPlanTutorial.png
Width (GeoJSON Units)	50
Height (GeoJSON Units)	35
Label Property	Name (String)
Filter Properties	

## 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

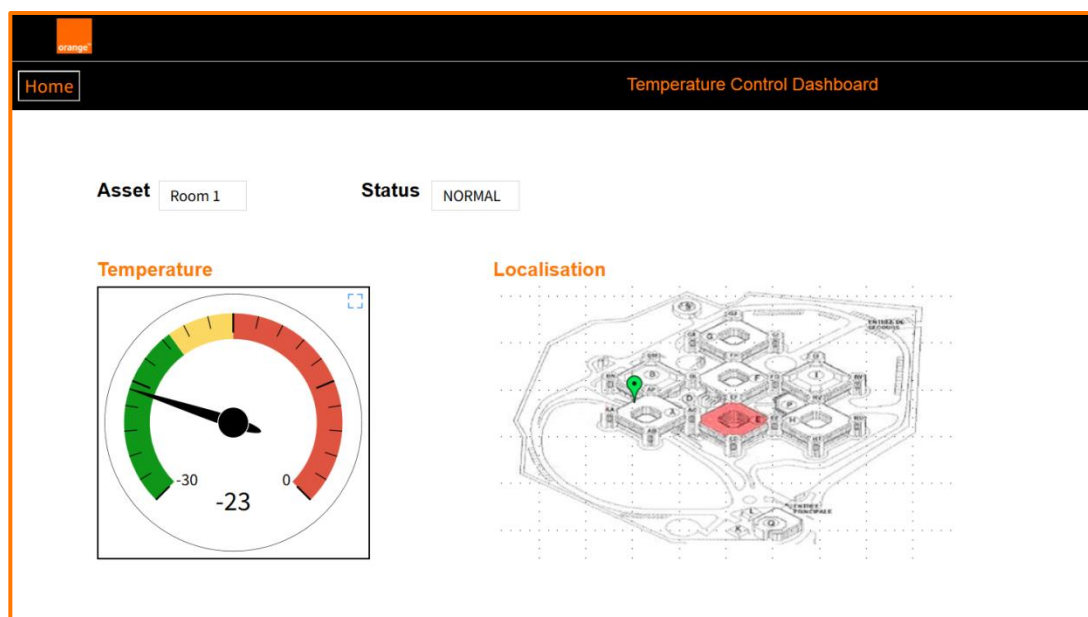
- Green Zones set to -30:-19
- Yellow Zones set to -19:-15
- Red Zones set to -15:0

PROPERTIES	
Type	Gauge
Name	GaugeRoom1
Height Policy	Explicit <input type="button" value="v"/>
Width Policy	Explicit <input type="button" value="v"/>
X	50
Y	430
Height	200
Width	200
CSS Class	vantiqGauge
Tooltip	
Data Stream	Asset1DataStream (Timed Query for Typ <input type="button" value="v"/>
Data Stream Property	Temperature (Real) <input type="button" value="v"/>
Border Thickness	2 <input type="button" value="v"/>
Minimum	-30
Green Zones	-30:-19
Yellow Zones	-19:-15
Red Zones	-15:0
Maximum	0
Minor Ticks	5

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

PROPERTIES	
Type	InputString
Name	InputString22
Height Policy	Natural Size
Width Policy	Size to Parent
X	160
Y	70
CSS Class	vantiqInputString
Tooltip	
Label Text	
Label Font Size	14
Label Alignment	Center
Label Style	Plain
Label Color	Transparent
Data Binding	page.data.CurrentName
Placeholder	
Is Optional?	<input checked="" type="checkbox"/>
Default Value	
Is Read Only?	<input type="checkbox"/>
Is Password?	<input type="checkbox"/>
On Change	<None>

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”

PROPERTIES	
Type	Gauge
Name	Gauge14
Height Policy	Explicit
Width Policy	Explicit
X	100
Y	200
Height	310
Width	310
CSS Class	vantiqGauge
Tooltip	
Data Stream	CurrentAssetStream (Timed Query for Ty
Data Stream Property	Temperature (Real)
Border Thickness	2
Minimum	-30
Green Zones	-30:-19
Yellow Zones	-19:-15
Red Zones	-15:0
Maximum	0
Minor Ticks	5

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

PROPERTIES	
Type	FloorplanViewer
Name	FloorplanViewer26
Height Policy	Explicit
Width Policy	Explicit
X	550
Y	200
Height	310
Width	540
CSS Class	vantiqFloorplanViewer
Tooltip	
Data Stream	CurrentAssetStream (T)
Data Stream Property	IndoorLocation (GeoJS)
Border Thickness	0
Is Scaleable	<input checked="" type="checkbox"/>
Floor Plan URL	../docs/NS/Orange_Sole
Width (GeoJSON Units)	50
Height (GeoJSON Units)	35
Label Property	Status (String)
Filter Properties	