AUTODESK UNIVERSITY

IOT15559

An Introduction to Fusion Connect

Allan O’Leary
Autodesk Inc

Brian Sherman
Autodesk Inc

Learning Objectives

- Understand the Fusion Connect object model
- Learn how to build basic applications based on real-world products
- Learn how to create a rich application reporting interface
- Learn how to emulate real-world data without connecting a single device

Description

This class will introduce attendees to the Fusion Connect development platform. You will learn how to model a digital twin of your machine assets, and (using a little business logic along with our point-and-click tools) you will learn how to build an Internet of Things application that turns your data into actionable events—visual status and graphs that you can capitalize on to collect operational feedback instantaneously from your customers and improve design or build unique value business offerings based on your products. This session features SeeControl (now Fusion Connect) and Fusion Connect.

Your AU Expert(s)

Allan O’Leary

After several years’ experience in the manufacturing industry involved in maintenance services and product design, working with and supporting Autodesk software, Allan O’Leary joined the Data Management Team at Autodesk, Inc., in 2008 where he focuses on the delivery and implementation of Vault software products. He is currently a product manager working out of the San Francisco office focused on the Fusion Connect Internet of Things (IoT) platform.

Brian Sherman
Brian Sherman is a senior software engineer on the Fusion Connect Team at Autodesk, Inc., based in the San Francisco office. He joined this team in 2014, and focuses on a number of areas, such as general development of the Internet of Things (IoT) cloud service, writing cloud adapters, training others on the use of the platform, and building customer solutions. Sherman has a bachelor’s degree in computer science and political science from the University of California, Berkeley.

Introduction

Fusion Connect is a cloud based Internet of Things application platform. Fusion Connect is comprised of 2 main components being a scalable data collection infrastructure and an application building environment.

One of the greatest challenges in development a bespoke IoT application is implementing a robust and scalable data collection service. Fusion Connect is built on a cloud native device connection interface referred to the Device Adapter Layer to collect data from physical devices regardless of message protocols via a large library of standard and custom device adapters.

The collected data is processed and normalized, depending on your requirements can be stored in your application, transferred to another data store or processed to simply provide system status or daily statistics.

Once your data is collected and analyzed it is presented to designers, sales, customers and other consumers in the actual application via a dashboard, detailed reports or system alerts of your own design.

Fusion connects provides a no-coding application development environment with all the tools necessary for one of your product specialists to construct a fully functional IoT App.

The diagram below provides an overview of the system architecture. The focus of this class is the application development including the business process routines for processing messages, and constructing a dashboard from product data and reports.

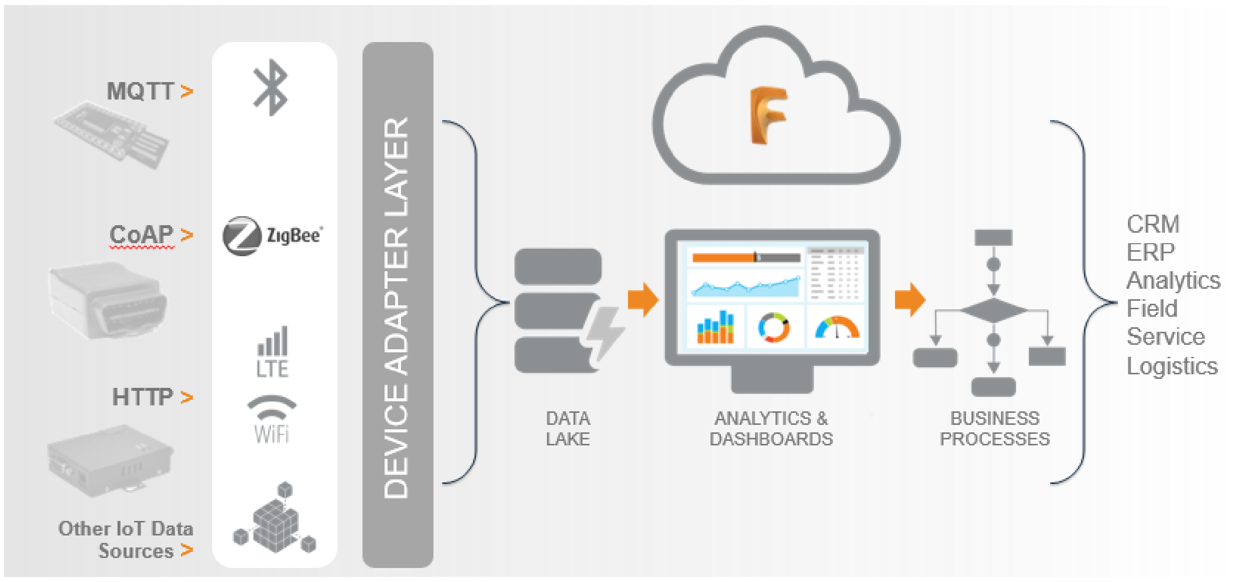


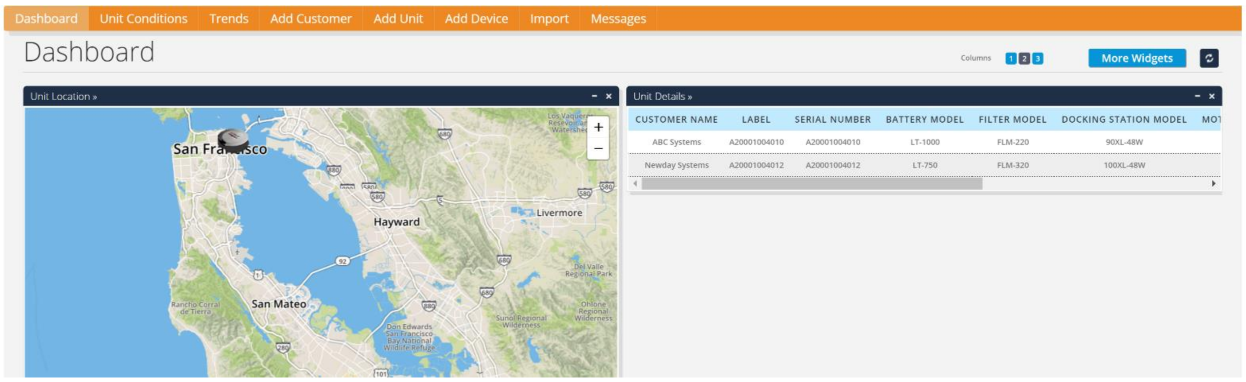
Figure 1. Fusion Connect System Overview

Applications Elements

When it comes to working on the Fusion Connect platform there are 2 fundamental environments we need to be familiar with. The Dashboard that provides the primary interface for end users to view and consume product information and the Design View for administrators to construct the end user application.

Dashboard

This is the landing page for your end user, whether that is an internal data consumer or a customer this is where we display widgets proving abridged report data that provides an overview of the "Things" we are monitoring.



CUSTOMER NAME	LABEL	SERIAL NUMBER	BATTERY MODEL	FILTER MODEL	DOCKING STATION MODEL	MOT
ABC Systems	A20001004010	A20001004010	LT-1000	FLM-220	90XL-48W	
Newday Systems	A20001004012	A20001004012	LT-750	FLM-320	100XL-48W	

Figure 2. Example Fusion Connect Dashboard

From the Dashboard users can click through to more detailed report data, or we can expose additional user access via the menu items to trend views, raw message data, forms for entering information or "things" and any other action the user custom role permits.

Design View

Those that have sufficient privileges (Usually your companies super user or site administrator) will have access to design view (a check box typically on the top right of the dashboard) which exposes the application building environment and an overview of the object model laid out for your application.

This is where we build and edit our application as well as having an overview of the application size, data and usage.

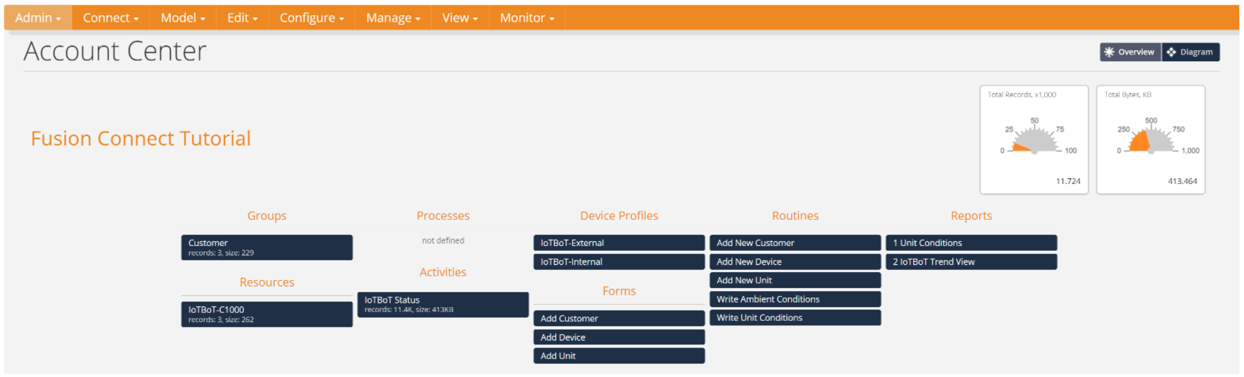


Figure 3. Fusion Connect Design View (Account Center)

Understand the Fusion Connect object model

The first step in building you Fusion Connect product application is piecing together the business level objects.

There are 3 primary objects really are a must in order to correctly manage your "Thing"

- Groups
- Resources
- Devices

In addition to this we will provide an overview of activities and processes

Groups

Groups are, as the name suggests, a method for grouping and organizing assets or "things" in your application.

A group may be a customer, a dealer, a region or physical building for example.

Groups can contain additional groups and they are one of the primary means for building a sub-accounts within your application to control user access to "things" and data.

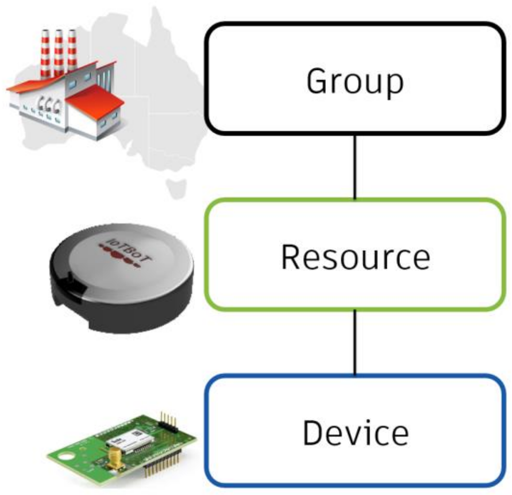


Figure 4. Basic Object Structure

Resources

Resources are the actual "Thing" that we are looking to monitor. This is the compressor, the pump for example that we want to track and report on.

Resource should contain two distinct data types, being basic asset data management information allowing us to identify the asset and dynamic condition readings, so we can show current operating status.

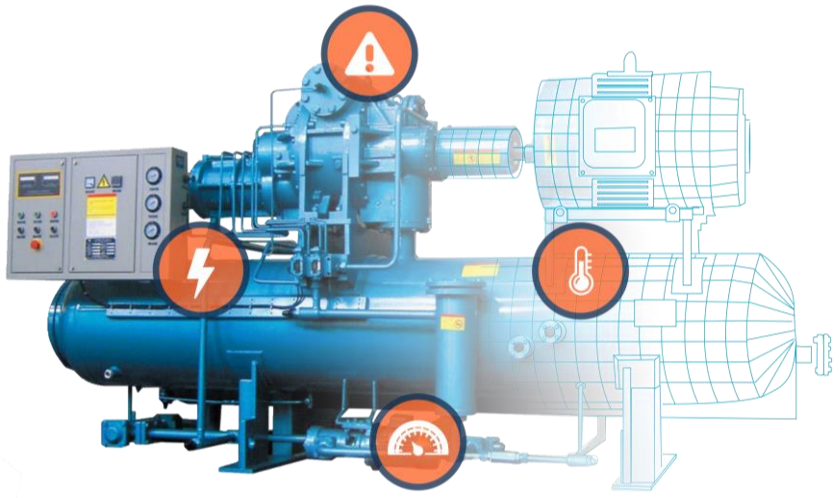


Figure 4. Basic Object Structure

Devices

Devices are the virtual instance of the physical sensors and controls used to collect data from the device adaptors and associated with either a resource or group.

Fusion Connect does not consider the device to be a representative of the "Thing" in our application environment, permitting more than one device to be attached to a single resource. In addition to this we assume that users may want to connect devices to a group to collect data like ambient temperature or light from a building for example.

The device object itself must have a unique identifier (typically something like the serial number of the physical device). When a message is received at the Fusion Connect cloud adapter the embedded source data is used to associate the message data with the virtual device and in turn with the correct resource, group or account.

Note, when a device message is posted to the Fusion Connect adaptor port, if no corresponding device is found the message is ignored. A virtual device must always exist for incoming message information.

Figure 5. Device Example

In the diagram below we see a more complicated and perhaps realistic object model.

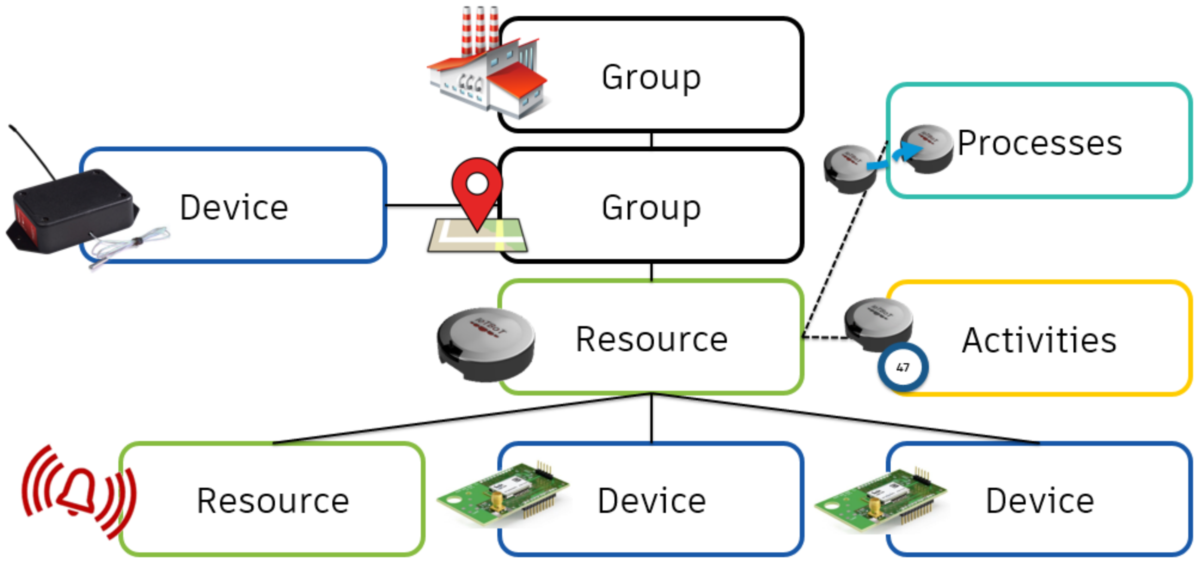


Figure 6. Expanded sample structure

Here we see that a group can contain a group, for example a customer containing a number of machine groupings or locations. A resource within these groups can contain one or more devices and in fact can be associated with another device (this could represent an alarm requiring some user interaction).

From a device perspective, as mentioned above we are able to connect a device to a Group or Resource, but note that a Resource is never able to contain a group, and likewise a device cannot contain a Resource or Group.

Over and above these three basic / essential objects we also have the ability to associate addition special purpose objects with our Resources.

Activities

Activities are not a necessary component in building your first application but if you are interested in tracking historical data and constructing trend views we require activities for this purpose.

Activities are essentially historical records or snapshots. Unlike a resource once created an Activity cannot be updated or the data edited. When a message is received or we manually enter some specific information we are able to extract details and build an activity with a time stamp, and relevant data.

We then use the activity type and its values to build out time based reports. We can see that in more detail below.

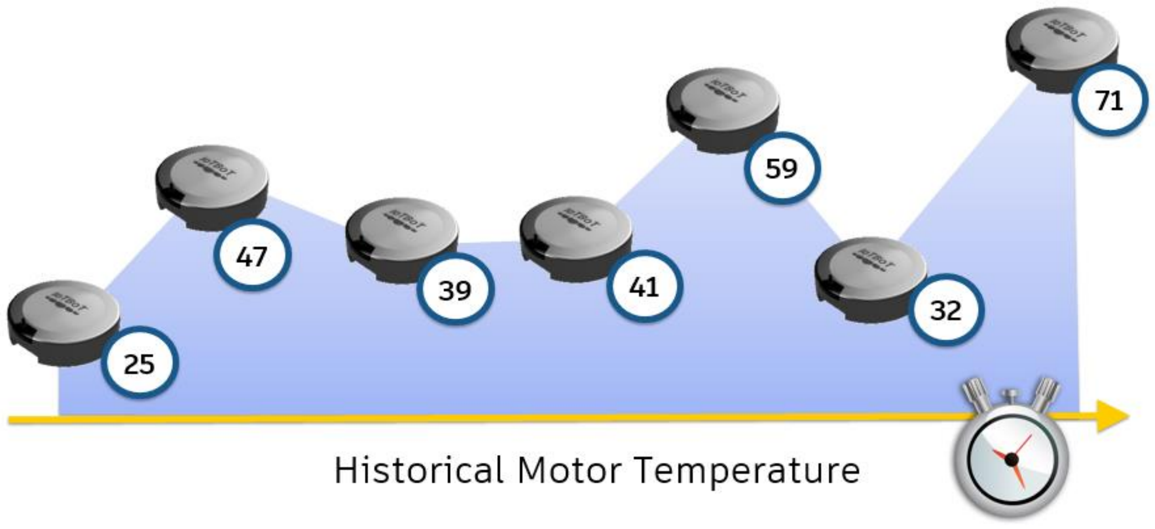


Figure 7. Activity – historical status data

Processes

As you collect data around the activity of your “thing” there are likely a number of discreet actions or series of actions that are critical in understanding your product usage.

For example if we are interested in understanding how long a machine is used per day or the average length of usage we need to look at when the machine is switched on and when its switched off which can be gathered by looking through the live readings but a process provides a simpler approach to capturing “run time” as we can define what conditions constitute the process we are interested in allowing the application to report on the process rather than the individual states.

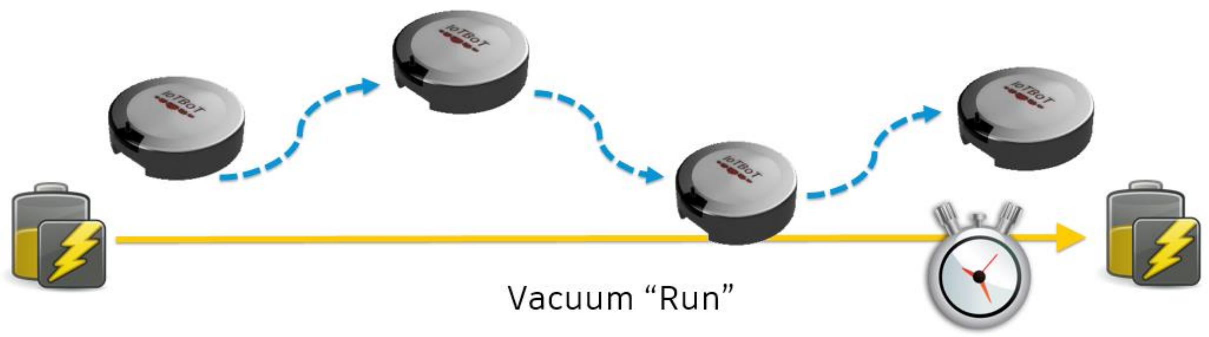


Figure 8. Process - vacuum leaves docking station for x minutes, returns to docking station

What's the difference between a Group Type and a Group?

For each object type, we have 2 components, the "Type" in the case of groups, resources, activities and processes, "Profile" in the case of a device – and the record itself.

The Group Type is how we define the object parameters or how we uniquely identify Group records, what detail we expect to enter or what data will be provided via device or routine inputs.

Likewise in the Type we define the relationships associated with that object, is it a top level Group? What other Groups or Resources does it contain? Does this Group Type generate sub-Groups for application security purposes?

We need to define the Object Type or Device Profile and set up the relationships prior to adding any actual records.

We can think of as the Group Type being a spreadsheet column header and the Group records being the rows.

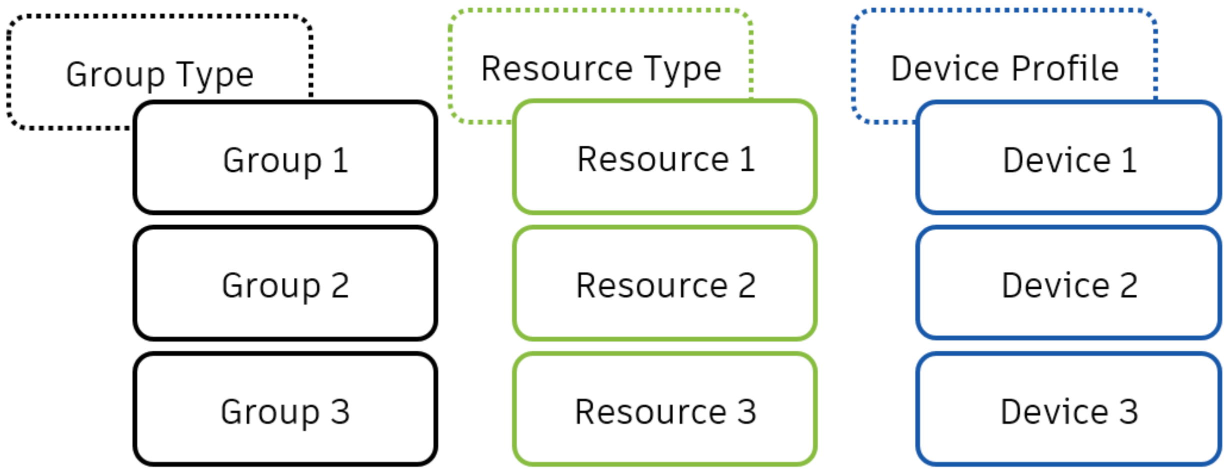


Figure 9. Type and Records

Below we will take a look at some examples and the order we suggest creating these.

Learn how to build basic applications based on real-world products

Now that we have an understanding of the basics of our object model we need to define each of the object types and create records.

For this class we have created a fictional industrial robot sweeper that reports basic data back to the application, on location, battery level, dirt hopper levels, and docking status.

To simplify the process of creating objects it is recommended we look at creating:

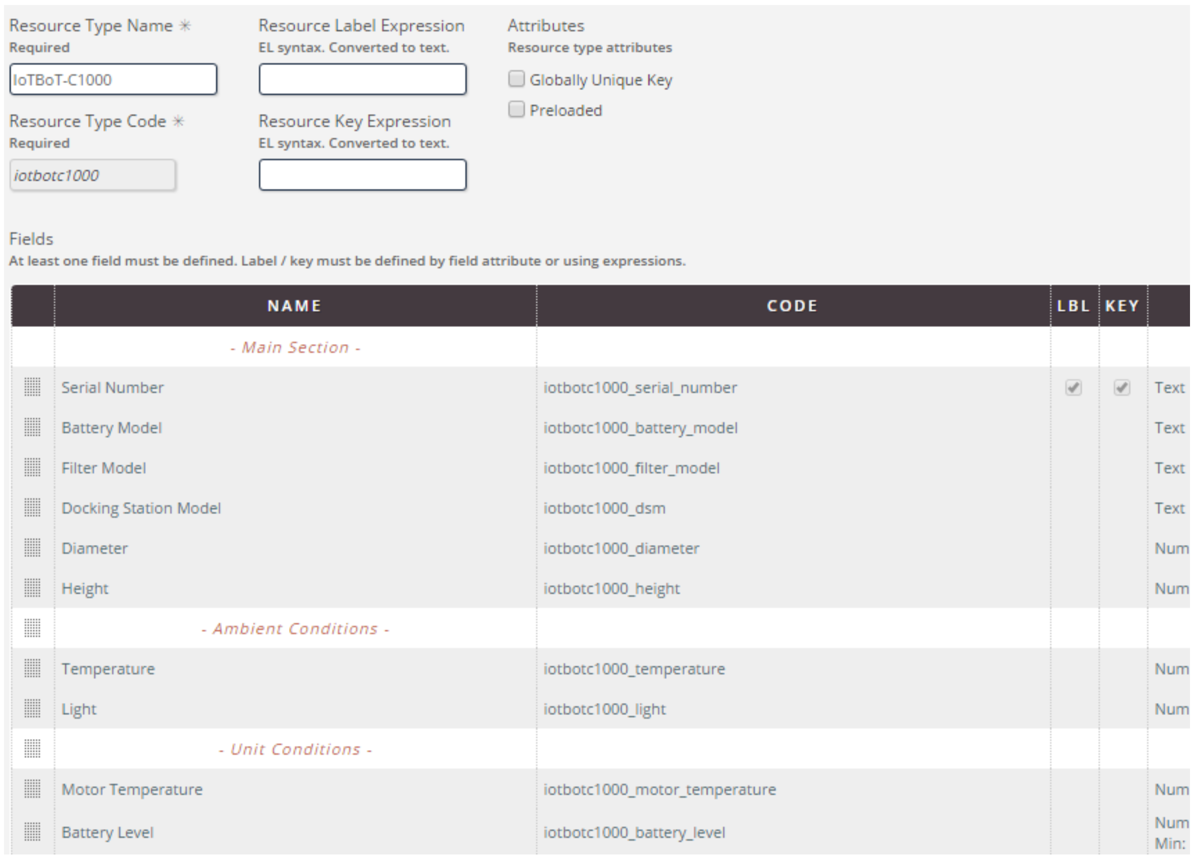
1) Resource Type(s)
2) Group Type(s)
3) Device Profile(s)

This way we are able to define the correct object structure dependencies for each object in the correct order.

Resource Type

The resource type is defined here as a specific model – we use a Serial Number field to uniquely identify each of the units as well as the record label.

The resource type is configured to capture other basic configuration data like the Battery information, Filter types and Unit size along with any other model variables.



Resource Type Name ✳ Required: IoTBoT-C1000

Resource Type Code ✳ Required: iotbotc1000

Resource Label Expression — EL syntax. Converted to text.

Resource Key Expression — EL syntax. Converted to text.

Attributes — Resource type attributes

- ☐ Globally Unique Key
- ☐ Preloaded

Fields — At least one field must be defined. Label / key must be defined by field attribute or using expressions.

NAME	CODE	LBL	KEY	
- Main Section -				
Serial Number	iotbotc1000_serial_number	✓	✓	Text
Battery Model	iotbotc1000_battery_model			Text
Filter Model	iotbotc1000_filter_model			Text
Docking Station Model	iotbotc1000_dsm			Text
Diameter	iotbotc1000_diameter			Num
Height	iotbotc1000_height			Num
- Ambient Conditions -				
Temperature	iotbotc1000_temperature			Num
Light	iotbotc1000_light			Num
- Unit Conditions -				
Motor Temperature	iotbotc1000_motor_temperature			Num
Battery Level	iotbotc1000_battery_level			Num Min:

Figure 10. Resource Type View

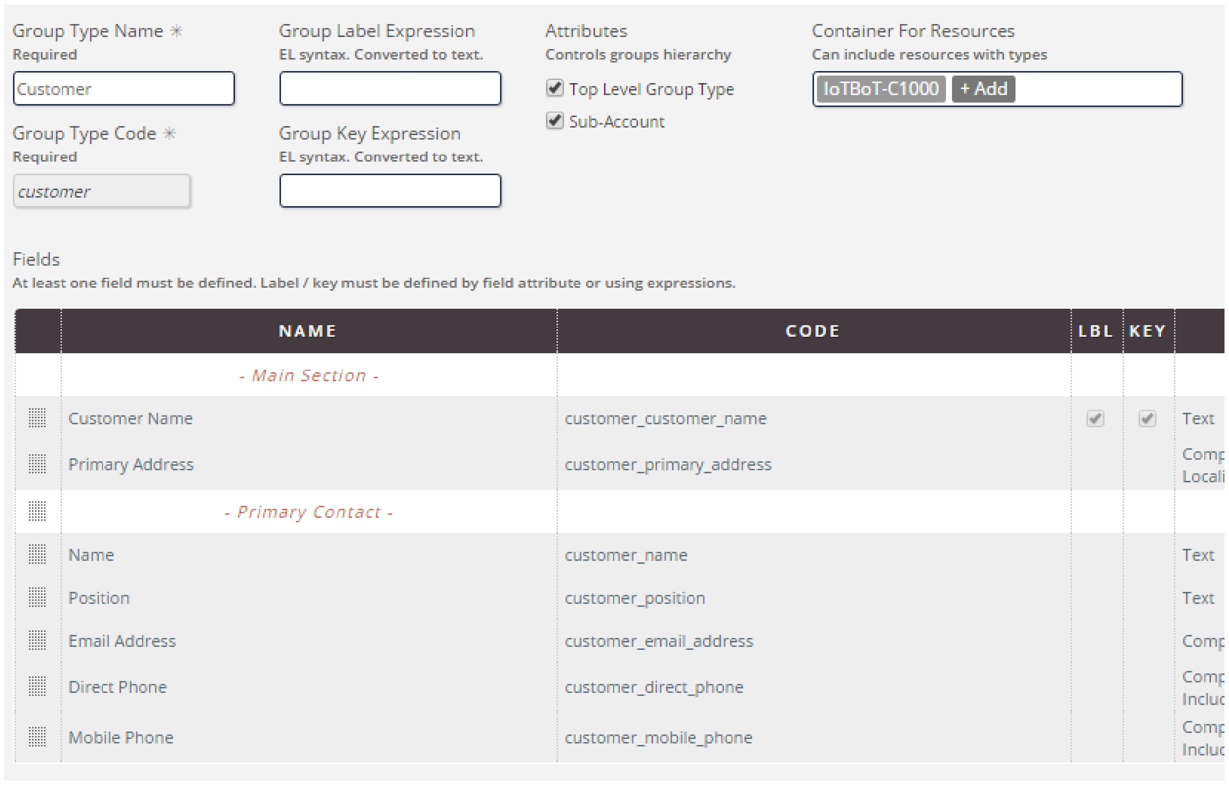
Apart from the basic unit details we also provide place holders for motor temperature, battery and hopper levels as well as the unit location, status information that will ultimately be updated via sensor hardware on the physical asset.

Note that we added an icon field to use when representing this "thing" on a map or in a table report.

Group Type

The Group example here is a customer, this could be a dealer or a building location too, but here we are simply going to gather the asset owners account name, shipping address, and the contact details of the primary contact, perhaps an asset manager or even maintenance manager in this case.

Note that this group is listed as a "Top Level Type" meaning it is not used in any other groups and the "Sub-Account" designation means it also generates sub-accounts automatically, so we can set access to units based on customer and provide this to customers to see only their assets.



Group Type Name ✳ Required: Customer

Group Type Code ✳ Required: customer

Group Label Expression EL syntax. Converted to text.

Group Key Expression EL syntax. Converted to text.

Attributes Controls groups hierarchy
- ✓ Top Level Group Type
- ✓ Sub-Account

Container For Resources Can include resources with types: IoTBoT-C1000 + Add

Fields
At least one field must be defined. Label / key must be defined by field attribute or using expressions.

NAME	CODE	LBL	KEY	
- Main Section -				
Customer Name	customer_customer_name	✓	✓	Text
Primary Address	customer_primary_address			Comp Locali
- Primary Contact -				
Name	customer_name			Text
Position	customer_position			Text
Email Address	customer_email_address			Comp
Direct Phone	customer_direct_phone			Comp Incluc
Mobile Phone	customer_mobile_phone			Comp Incluc

Figure 11. Group Type View

When we set up the group, we now define what this is a "container for", does it contain another group, or resource for example. Here the Groups Type is designated to contain the IoTBoT Resource Type we created.

This means when we start to create IoTBoT's, they will be looking for a customer group to be associated with.

Device Profile

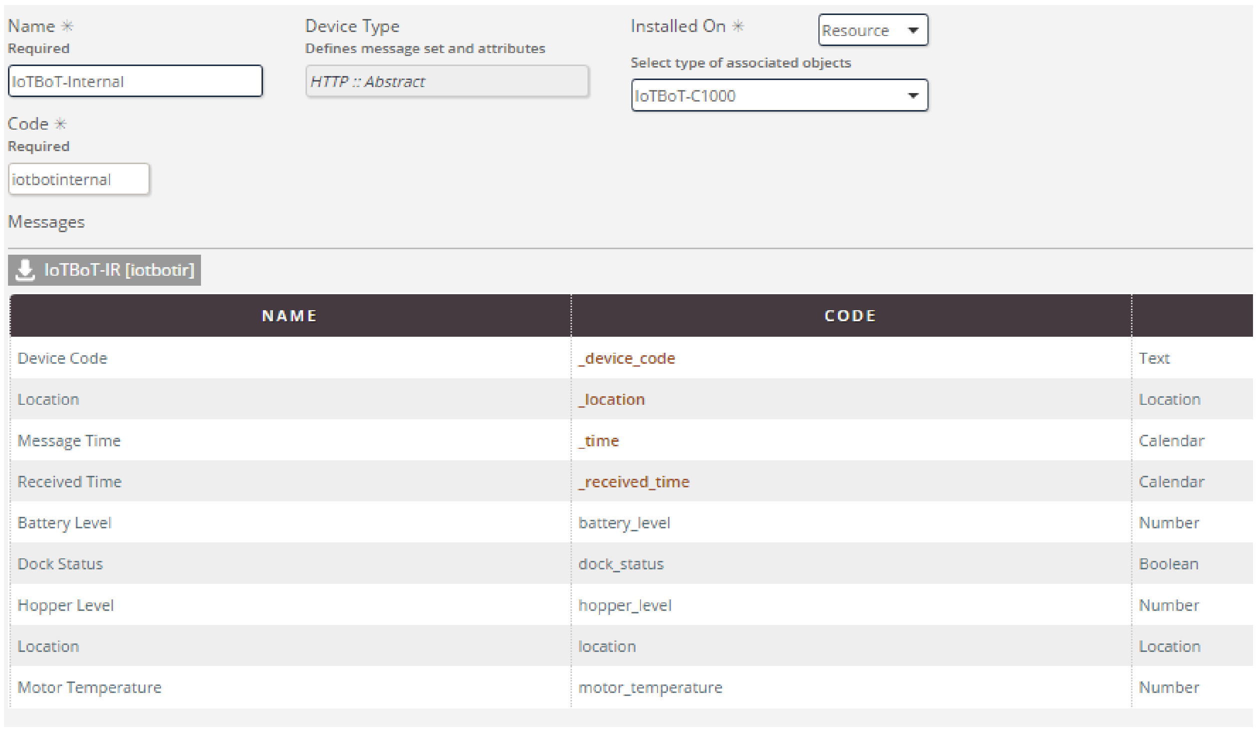
For the device profile we have configured this using the "HTTP::Abstract" adaptor which defines the message protocol. Depending on your physical product requirements (long battery life, high product lifecycle, form factors, operating conditions, connectivity options for example) you will need to choose a physical device type and a messaging protocol to communicate with Fusion Connect.

Fusion Connect offers a large number of existing adaptors and message protocols within the Device Profile setup and can work with you to meet any custom communication needs.

Once we define the protocol, give our Device Profile a name and identify which Resource or Group Type it will be attached to, we need to specify a "Message"

The message is a set of data fields we expect to receive from that device, we will always need to have a device ID (so we know where the information comes from), as well as information about when the message arrived – so these fields will be pre-populated.

Depending on the adaptor type you choose there may also be some other standard properties (like GPS location for example) and we then add the custom fields we are specifically collecting, here the motor temperatures, hopper and battery levels for example.



Name ✳ Required: IoTBoT-Internal	Device Type — Defines message set and attributes: *HTTP :: Abstract*	Installed On ✳: Resource — Select type of associated objects: IoTBoT-C1000
Code ✳ Required: iotbotinternal		

Messages

IoTBoT-IR [iotbotir]

NAME	CODE	
Device Code	_device_code	Text
Location	_location	Location
Message Time	_time	Calendar
Received Time	_received_time	Calendar
Battery Level	battery_level	Number
Dock Status	dock_status	Boolean
Hopper Level	hopper_level	Number
Location	location	Location
Motor Temperature	motor_temperature	Number

Figure 12. Device Profile View

Once we have defined the primary objects including the object structure and characteristics we are able to start adding records. This can be done directly through the Connect menu for devices and the Edit Menu for Groups and Resources.

Typically though we would now create a form to simplify the record creation process, but this is a topic for another day, so next we will look at how to present information coming into the system and updating these objects.

Learn how to create a rich application reporting interface

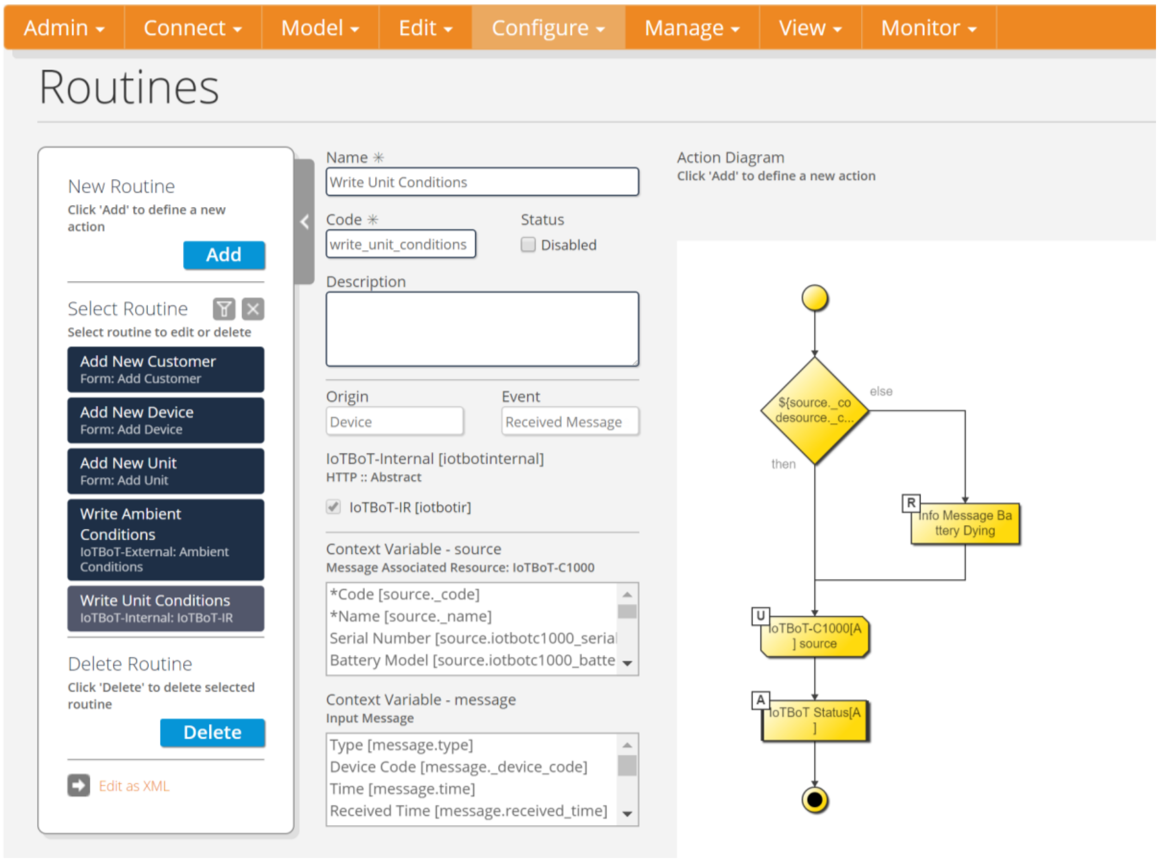
Now that we have a data model configured we need to design our application interface. The primary piece of the user interface is our dashboard populated primarily through system reports.

However, before we can start to generate these reports we need to map the flow of data through the application from the physical devices (or form entry) to the resources and groups, which is achieved through the concept of Routines.

Routines

Routines are the engine of the application, collecting inputs from data import, form submission and of course the actual device messages received via the device adapter.

The routines allow us to take actions on the data, including simple real time analytics on incoming messages, allowing us to check device readings for error states, compare values against historical or empirical data to send alerts or machine directives, update "thing" status and create new historical records (Activities).



Routines

New Routine
Click 'Add' to define a new action

Select Routine
Select routine to edit or delete

Add New Customer
Form: Add Customer

Add New Device
Form: Add Device

Add New Unit
Form: Add Unit

Write Ambient Conditions
IoTBoT-External: Ambient Conditions

Write Unit Conditions
IoTBoT-Internal: IoTBoT-IR

Delete Routine
Click 'Delete' to delete selected routine

Name ✳ Write Unit Conditions

Code ✳ write_unit_conditions

Status ☐ Disabled

Description

Origin: Device

Event: Received Message

IoTBoT-Internal [iotbotinternal]
HTTP :: Abstract

✓ IoTBoT-IR [iotbotir]

Context Variable - source
Message Associated Resource: IoTBoT-C1000

*Code [source._code]
*Name [source._name]
Serial Number [source.iotbotc1000_serial
Battery Model [source.iotbotc1000_batte

Context Variable - message
Input Message

Type [message.type]
Device Code [message._device_code]
Time [message.time]
Received Time [message.received_time]

Action Diagram
Click 'Add' to define a new action

${source._co desource._c...

Info Message Battery Dying

IoTBoT-C1000[A] source

IoTBoT Status[A]

Figure 13. Create Routine Interface

The simple "workflow diagram" interface is the key to the "no coding" approach. Typically this business logic would need to be baked into an application along with the report visualization, requiring a good deal of coding (and re-coding to make changes). The routines engine though is simple to configure, flexible and extensible as we can set and reference variables, leverage the EL mathematical language to implement mathematical functions as part of expressions along with simple data modification or calculation.

It is this processed data that drives the reports – by updating resource values, creating activities for historical trends and creating new alarm resources for example.

The Routine building interface shown in Figure 13 highlights the "Visio Style" method for creating and ordering "Actions" into a logical workflow (or "data flow"), something that most subject matter experts (engineers or product managers) should be able to navigate to build powerful application capabilities.

Actions

Actions (as seen in Figure 14 dialog) include all manner of logic operations (if / then), Object or Variable Find, Select, Add and Update operations. The ability to Delete objects, send Notifications like an alarm condition or return Messages about the routine status.

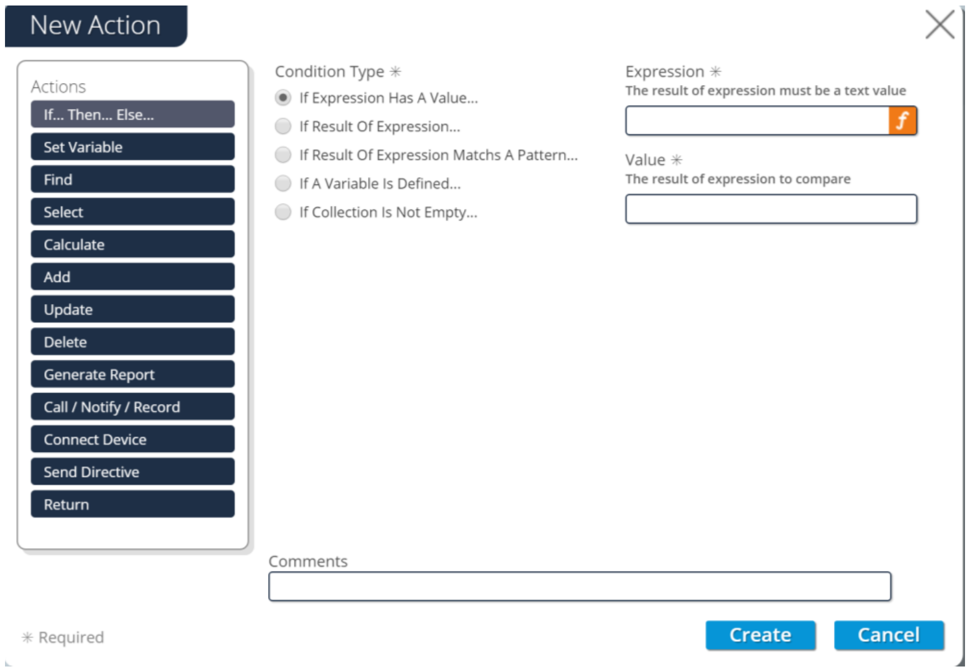

New Action
Actions
If... Then... Else...
Set Variable
Find
Select
Calculate
Add
Update
Delete
Generate Report
Call / Notify / Record
Connect Device
Send Directive
Return
Condition Type ✳
If Expression Has A Value...
If Result Of Expression...
If Result Of Expression Matchs A Pattern...
If A Variable Is Defined...
If Collection Is Not Empty...
Expression ✳
The result of expression must be a text value
Value ✳
The result of expression to compare
Comments
✳ Required
Create
Cancel

Figure 14. Action Dialog

We can through routines also carry out some very specific IoT actions such as connecting a Device to a Resource or Group and sending a "Directive" or taking some action on a machine based on an outcome from this routine.

Reports

Reports are how we typically present the data we have collected and calculated to application users. Reports come in many different forms depending on how the information is best presented. Table views for example are great for displaying a range of status values for a collection of assets or a list of recent alerts.

The Chart report view is useful for providing a clear representation of current machine values using line graphs, column graphs or pie charts of all varieties.

Gauges can be used to provide a telemetry style overview, trends provide a special report view where historical data is displayed along common time axis.

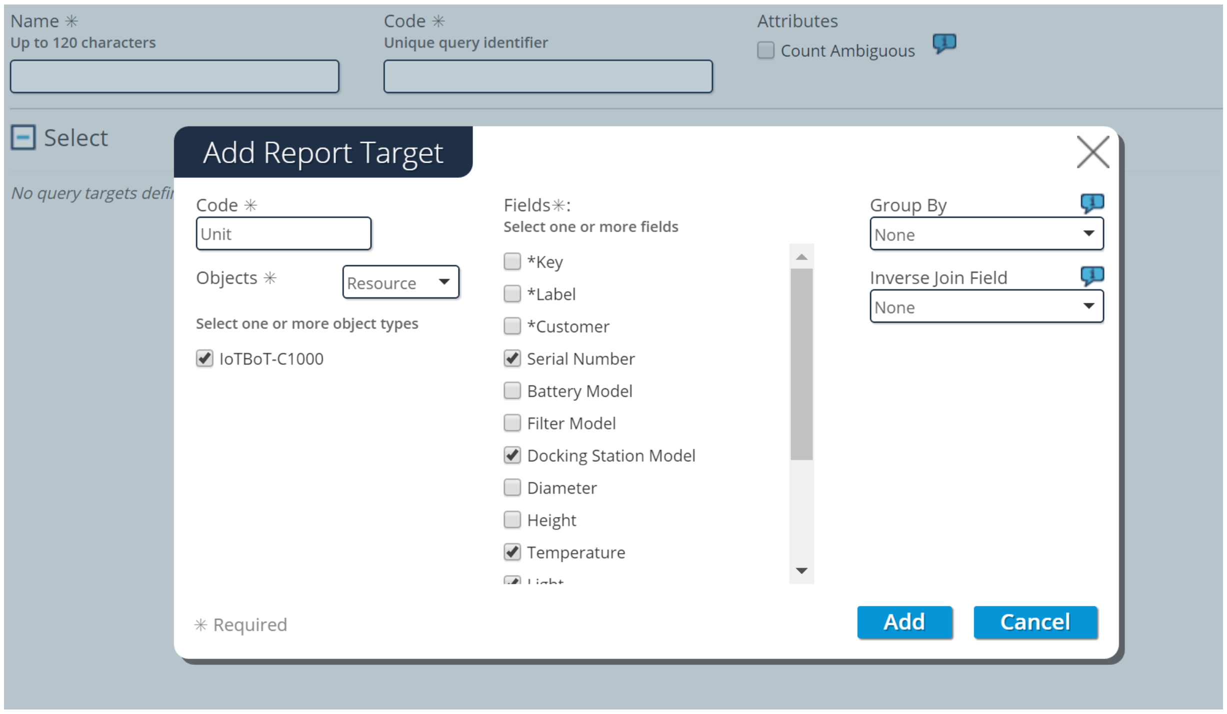
To create a Report

As a basic overview of this process

1) From the Configure Menu select Queries and Reports

2) Select the report target, are we creating a report on:

   - A Resource
   - A Group
   - A Process
   - An Activity



Name ✳ Up to 120 characters

Code ✳ Unique query identifier

Attributes: [ ] Count Ambiguous

Select

No query targets defi

Add Report Target

Code ✳: Unit

Objects ✳: Resource

Select one or more object types

[✓] IoTBoT-C1000

Fields ✳: Select one or more fields

- [ ] *Key
- [ ] *Label
- [ ] *Customer
- [✓] Serial Number
- [ ] Battery Model
- [ ] Filter Model
- [✓] Docking Station Model
- [ ] Diameter
- [ ] Height
- [✓] Temperature
- [✓] Light

Group By: None

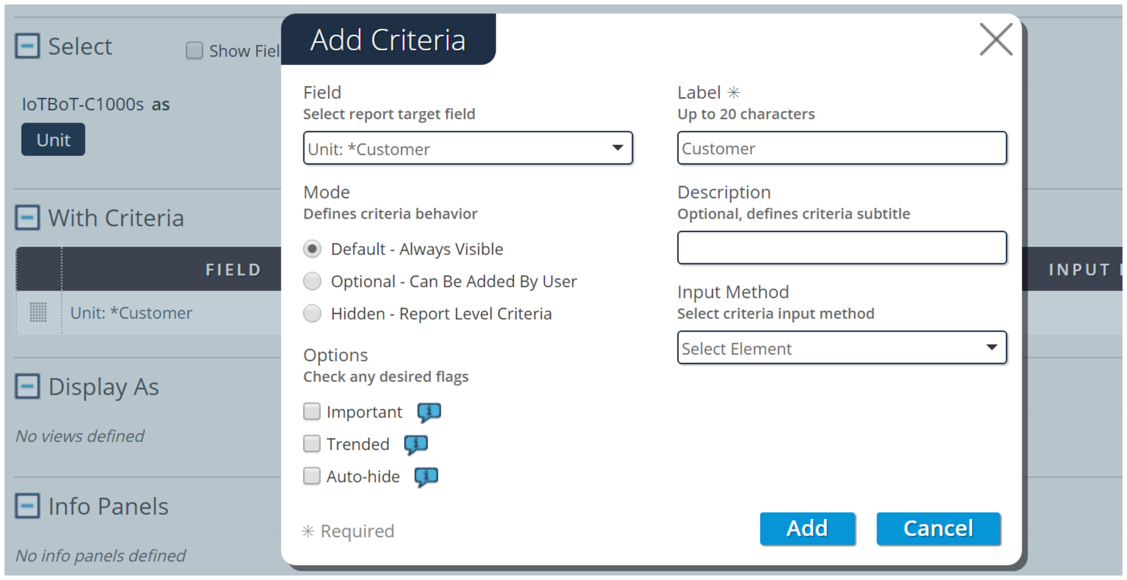
Inverse Join Field: None

✳ Required

Figure 15. Add Report Target / Target Fields

3) From your “Target” pick on the fields or object data you would like to appear on the report, this should include the values for filtering, grouping as well as the values we will render on the report (Temperature at a specific time for example).

4) Next we can drag and drop one of the target fields to create a criteria for filtering or from the "With Criteria" and Add Criteria to filter with options around hidden from user, selectable etc



Add Criteria

Field
Select report target field
Unit: *Customer

Label ✳
Up to 20 characters
Customer

Mode
Defines criteria behavior
[✓ Default - Always Visible]
Optional - Can Be Added By User
Hidden - Report Level Criteria

Description
Optional, defines criteria subtitle

Input Method
Select criteria input method
Select Element

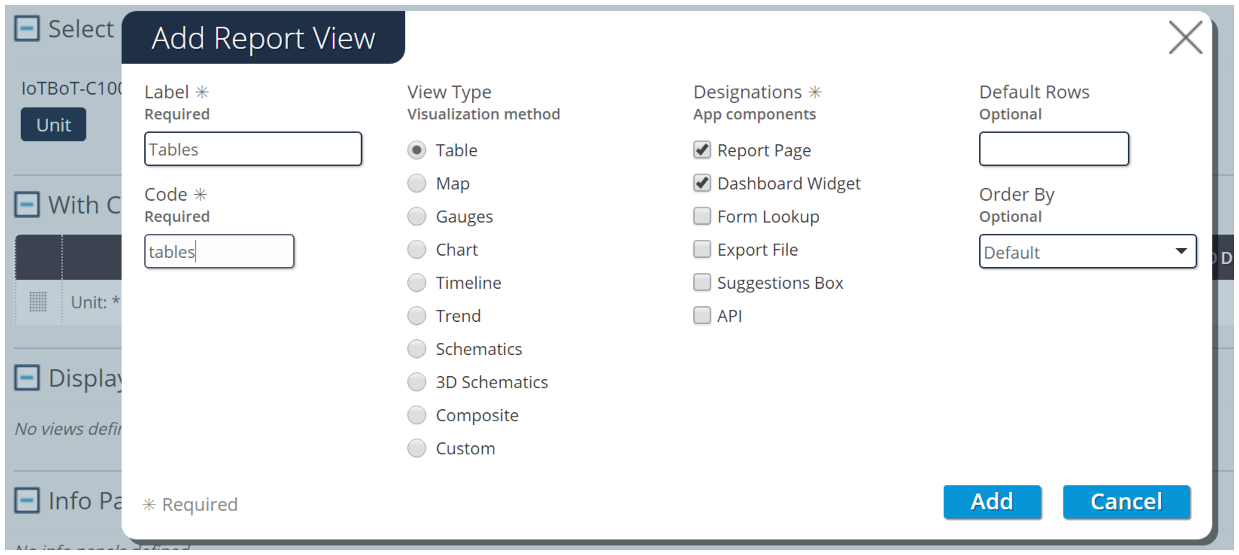
Options
Check any desired flags
☐ Important
☐ Trended
☐ Auto-hide

✳ Required

Add Cancel

Figure 16. Add Criteria

5) Underneath the "Display As" section we now define the report views. In Figure 16 we see the list of possible View Types.



Add Report View

Label ✳
Required
Tables

Code ✳
Required
tables

View Type
Visualization method
[✓ Table]
Map
Gauges
Chart
Timeline
Trend
Schematics
3D Schematics
Composite
Custom

Designations ✳
App components
☑ Report Page
☑ Dashboard Widget
☐ Form Lookup
☐ Export File
☐ Suggestions Box
☐ API

Default Rows
Optional

Order By
Optional
Default

✳ Required

Add Cancel

Figure 17. Add Report View

6) From designations select where you would like the report view to be accessible (Note that not all report types can be made available on the dashboard, trend views have their own space and API availability allows users to access the report on mobile or via a custom application connecting to your Fusion Connect Instance)

7) In this example if we select a simple table view we now need to select the fields to appear in the table from those identified in the target section this can be done by:
   - Dragging and dropping the target object to create a table with ALL fields
   - Dragging and dropping individual fields to populate a table
   - Manually "Add Column" to the table

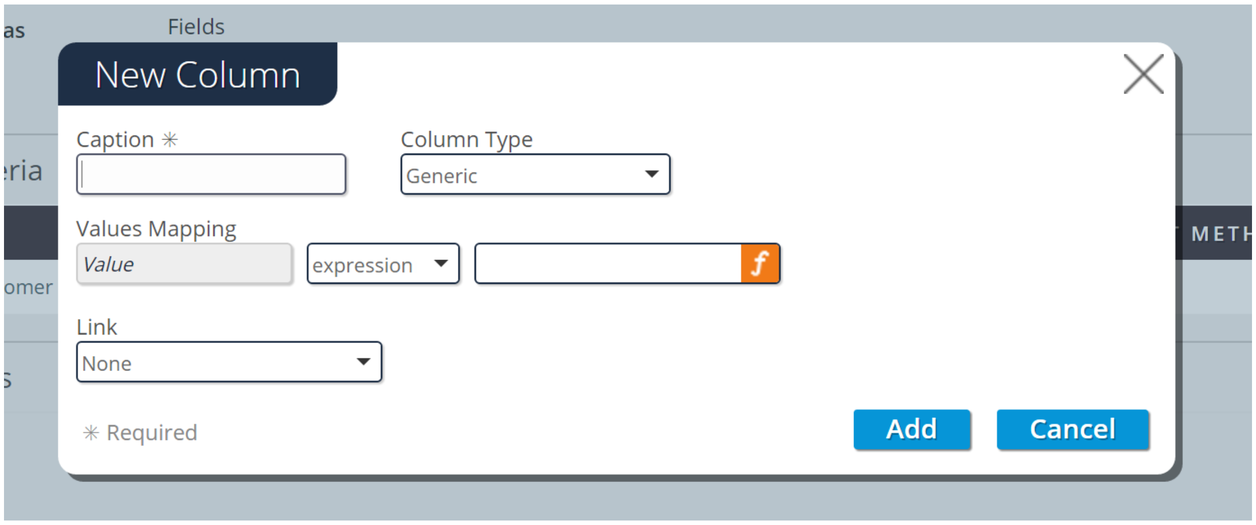


Figure 18. Add Report View

8) You can if you like also add an "Info Panel" to provide contextual information on what detail is being displayed for a record – extended detail so to speak.

9) Again, depending on the report view types we can continue to add additional report views (like a chart view, a map or gauge view) which will appear as separate tabs on your report.

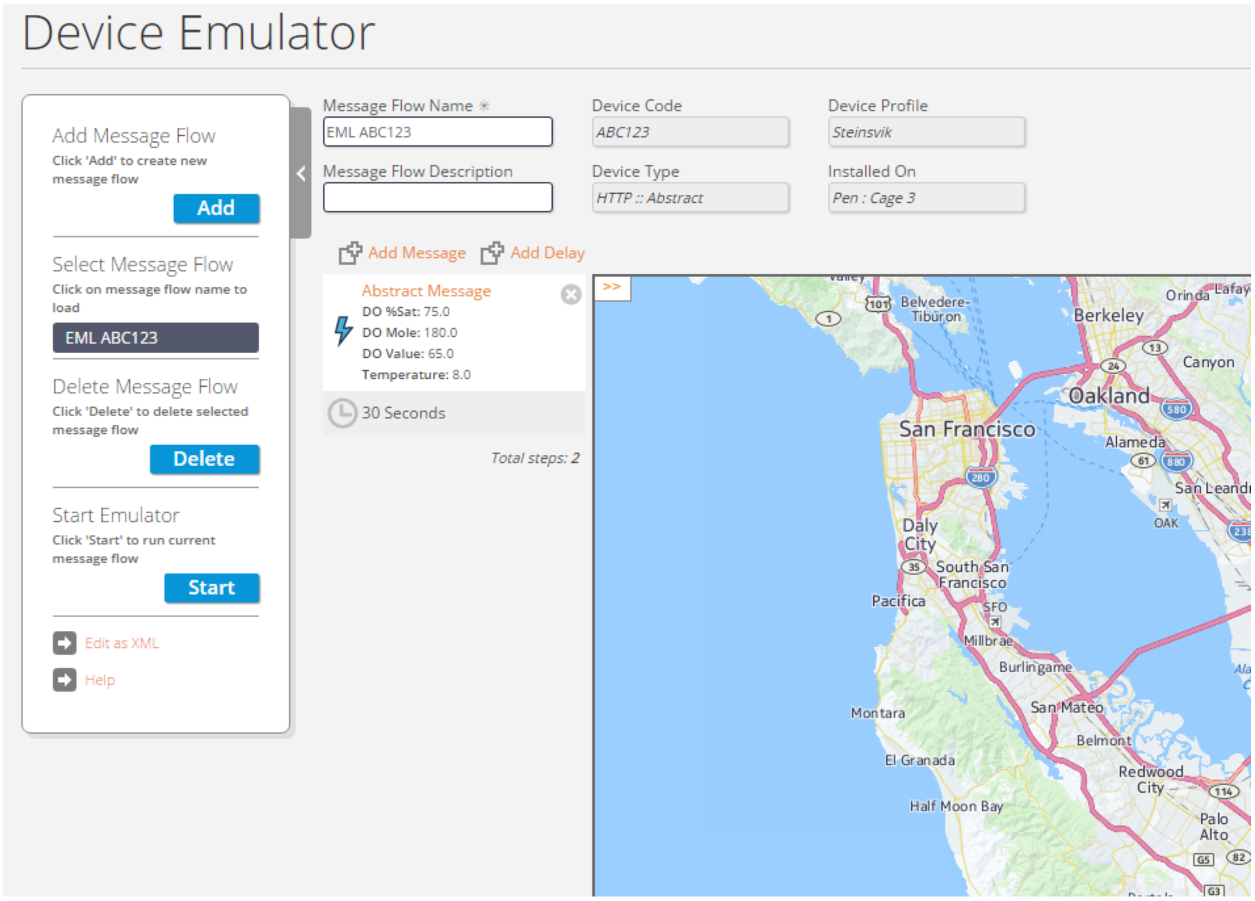
After you create your initial reports, you will see (most likely) the reports contain no data making it is hard to check that our routines and queries are working correctly. We will look next at how to address this and have a working application up and running in no time.

Learn how to emulate real-world data without connecting a single device

As you build out your application with the business logic and reporting interfaces, it is obviously highly advantageous to simulate real world data inputs and look for issues in the data processing and presentation.

When dealing with physical devices there can be a long lead time in accessing prototypes, developing custom hardware, installing and connecting to your units, working through data transmission and security to name a few facets of collecting live data, all of which will delay or limit your application development.

To address this challenge Fusion Connect provides a built in “device emulator”, a simple feature that allows us, based on the device profile configuration and systems virtual devices, to send sample data based on mean real system values with the ability to provide variation on this value for randomized device inputs.



Device Emulator

Add Message Flow
Click 'Add' to create new message flow
Add

Select Message Flow
Click on message flow name to load
EML ABC123

Delete Message Flow
Click 'Delete' to delete selected message flow
Delete

Start Emulator
Click 'Start' to run current message flow
Start

Edit as XML
Help

Message Flow Name: EML ABC123	Device Code: ABC123	Device Profile: Steinsvik
Message Flow Description:	Device Type: HTTP :: Abstract	Installed On: Pen : Cage 3

Add Message · Add Delay

Abstract Message
DO %Sat: 75.0
DO Mole: 180.0
DO Value: 65.0
Temperature: 8.0

30 Seconds

Total steps: 2

Figure 19. Device Emulator UI

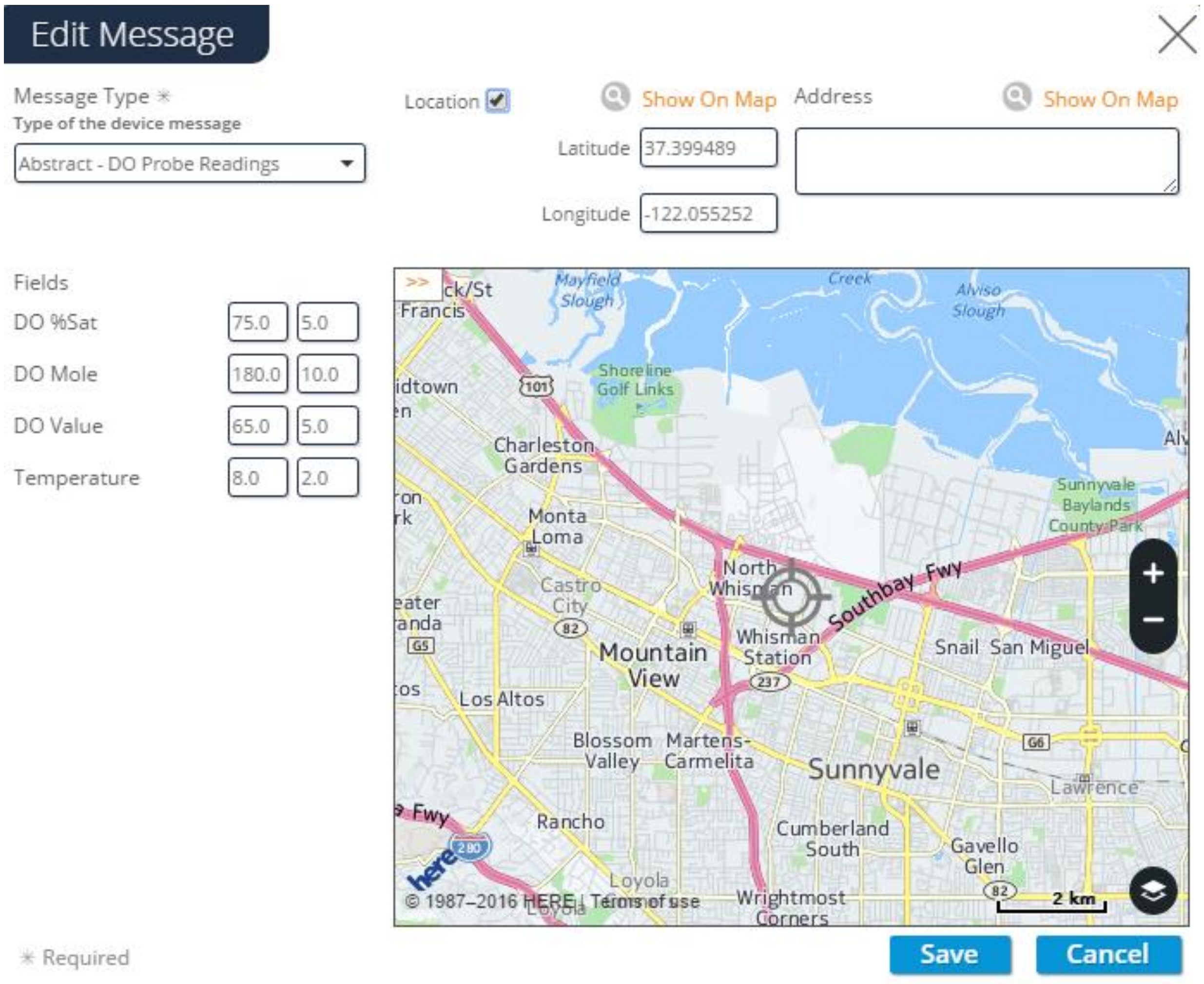
Running the device emulator for a set number of repetitions or time period allows our reports to build out a large number of data points, test routine alert set points (upper and lower reading value) and the overall application logic.

Set up a Message Flow

In order to start this process we need to set up a message flow which consists of selecting a device that we have configured in the system and adding a message with values in it.

To create a Message Flow

1) Go to the Connect menu and Select Device Emulator
2) Click add on the left menu to create a new Message Flow
3) From the list of available devices, select the device you would like to emulate
4) Now select "Add Message" to present the emulation UI where we can enter message location source, mean numeric values, along with any variants



Edit Message

Message Type ✳
Type of the device message
Abstract - DO Probe Readings

Location ✓

Show On Map

Latitude 37.399489
Longitude -122.055252

Address

Show On Map

Fields		
DO %Sat	75.0	5.0
DO Mole	180.0	10.0
DO Value	65.0	5.0
Temperature	8.0	2.0

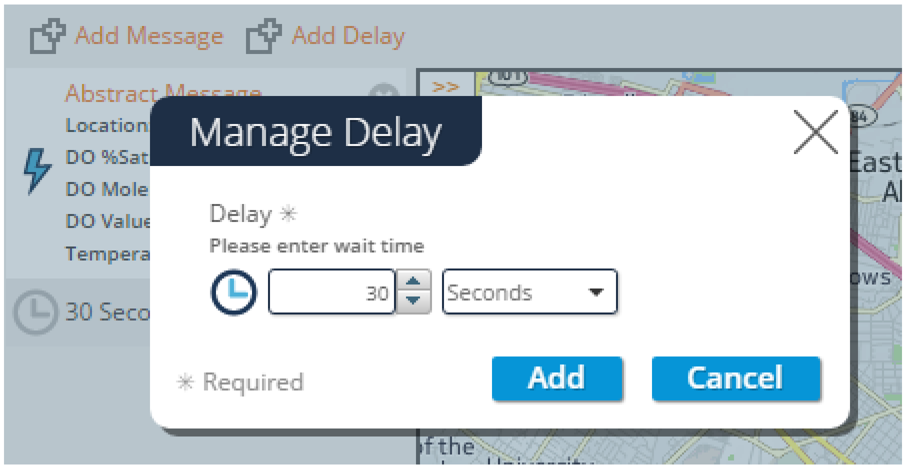
✳ Required

Figure 20. Edit Message Content

Note the field values represent the mean and variation – NOT upper and lower limits

5) Save the message settings and Add the Message Flow to save these settings

6) Note that once the message is set up we can configure message timing (the delay between messages) – for initial testing we can set this at a few seconds, but only for short periods



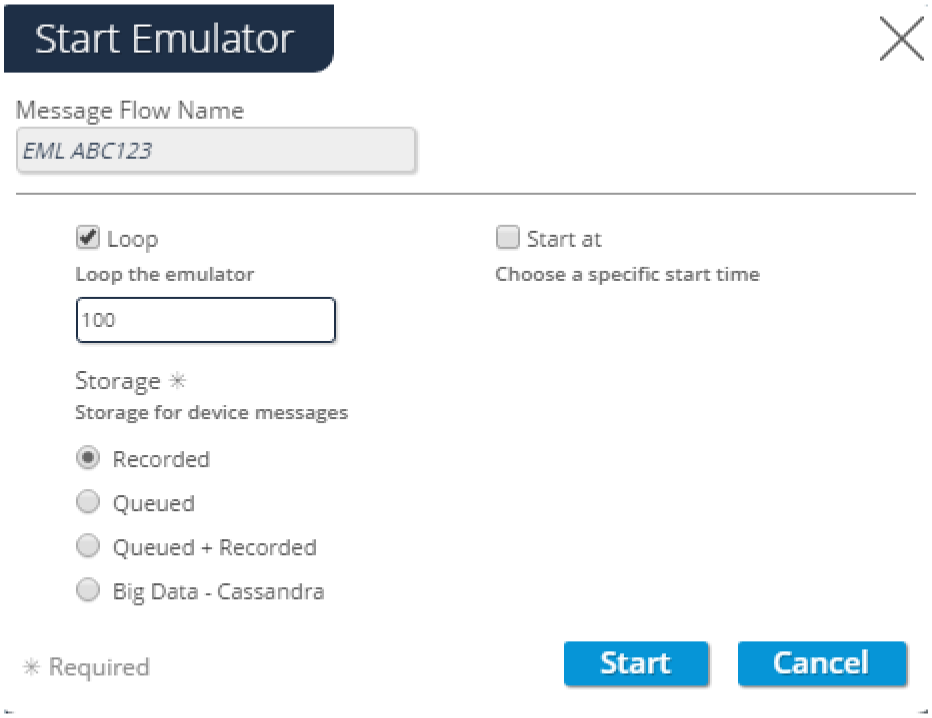
Manage Delay

Delay ✳
Please enter wait time
30 Seconds

✳ Required
Add
Cancel

Figure 21. Add Delay to Message

7) Now we simply select “Start Emulator” from the left menu



Start Emulator

Message Flow Name
EML ABC123

[✓] Loop
Loop the emulator
100

[ ] Start at
Choose a specific start time

Storage ✳
Storage for device messages
[✓ Recorded]
Queued
Queued + Recorded
Big Data - Cassandra

✳ Required
Start
Cancel

Figure 22. Set the emulator start time, number of loops and storage

8) Here we will be prompted to enter the number of times to loop the emulator (how many messages to send)

9) Once it begins you will see the emulator title highlight with a green dot to indicate that the emulator is running.

Review Device Messages

Once the emulator is running you can review the incoming messages.

Select Device Messages from the Monitor menu – here we can see the message statistics (how many have been received total, last 24 hours, what the size of the collected messages are and what rate we are currently receiving – this lets us know the emulator is working.

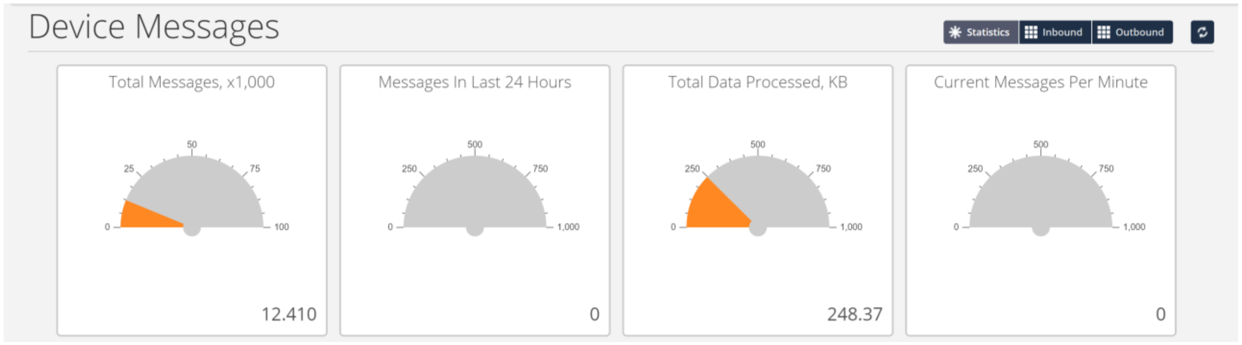
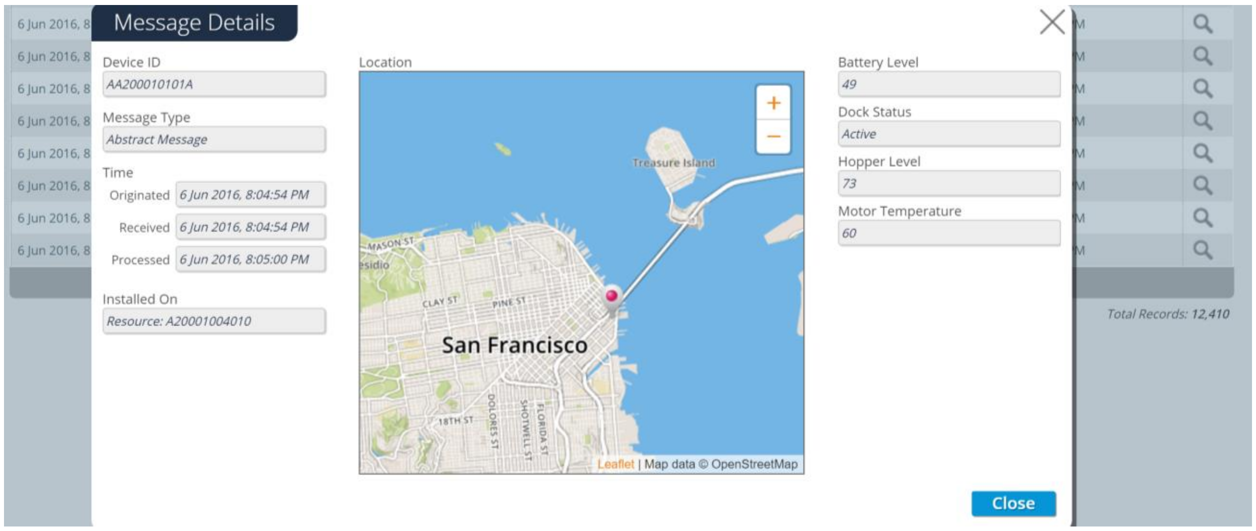


Figure 23. Device Message Statistical View

From top right select “Inbound” to view the actual list of incoming messages received from the emulator. This is another great tool for making sure the messages are processed correctly and we can click on the magnifying glass to look at message content.


Message Details

Device ID: AA200010101A
Message Type: Abstract Message
Time
Originated: 6 Jun 2016, 8:04:54 PM
Received: 6 Jun 2016, 8:04:54 PM
Processed: 6 Jun 2016, 8:05:00 PM
Installed On: Resource: A20001004010
Location
Battery Level: 49
Dock Status: Active
Hopper Level: 73
Motor Temperature: 60

Total Records: 12,410

Figure 24. View Message Content

If your routines and reports are operating correctly we should be able to view our reports and see incoming unit status, trending information and error or alert notifications.

If this is not the case we now have the opportunity to go back and check the resources to make sure data is being correctly associated, and dig back into our routines to see if we have broken logic or missing steps, an invaluable approach to application building in the context of what may be a complex and time sensitive project.

Configure Dashboard

Finally, now we have created all our objects, built the object structure and created some records, set up routines and reports and actually populated the application with data, we should add some report views to our dashboard.

Exiting Design Mode, click on the "More Widgets" button. We will now see all report views available on the dashboard.

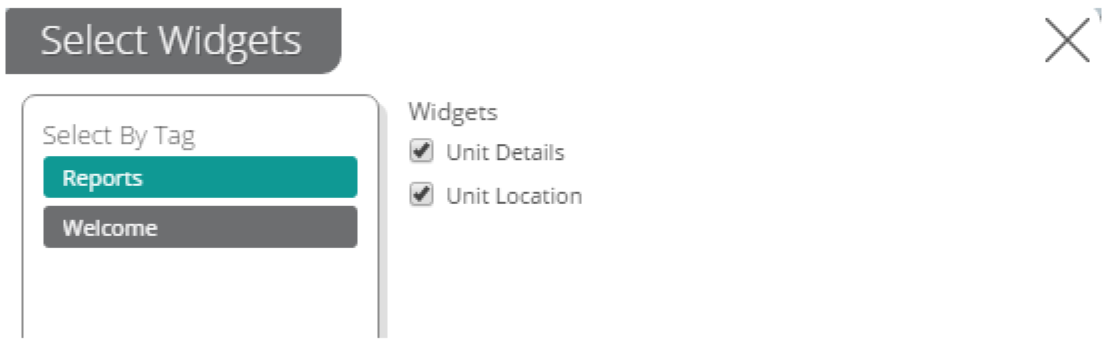


Figure 25.Set up Widgets on Dashboard

Check what you want to appear and organize the report views by selecting how many report view columns to display and dragging / dropping to change order.

Summary

By working through your product requirements and setting up a correlating object structure to represent your business units, customers, products and hardware we are able to introduce business logic to drive product condition data, run reports on key performance indicators and display this is a powerful application environment for your designers, partners, customers and any other stakeholder participating in your business.

Expanding on these basic concepts to build forms, more complicated routines and processes, linking into artificial intelligence capabilities as well as connecting with other business systems including spare parts and service, purchasing and even change management Fusion Connect becomes a powerful tool for business transformer and competitive differentiation.

More information is available from your Autodesk team on features and implementation of IoT technology for your business.