



sepasoft

MES Training Manual

2018 Summer Edition

OEE Downtime | Track & Trace | SPC | Recipe/Changeover
Web Services | Instrument Interface | Barcode Scanner
MES Enterprise

www.sepasoft.com
800.207.5506 | info@sepasoft.com

MES Training Manual

Copyright 2018
All rights reserved

Sepasoft, Inc
1264 Hawks Flight Court
Suite #250
El Dorado Hills, CA 95762

MES Training Manual

© 2018 Sepasoft, Inc.

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

All Sepasoft products are trademarks or registered trademarks of Sepasoft. Other brand and product names are trademarks or registered trademarks of their respective holders.

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. Sepasoft shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

Printed: August 2018 in the U.S.A.

Classroom Training Itinerary

We provide an instructor led classroom training course in El Dorado Hills, California that covers a broad range of topics including MES project management and follows the online tutorial for the class practical. We start each day at 9:00am and end at 4pm, except for Friday, which ends at 12pm. The course content is outlined below.

Class Outline

Day	Morning Topic	Afternoon Topic
Monday	MES Product Suite Overview	Track & Trace
Tuesday	Track & Trace	Track & Trace
Wednesday	OEE 2.0	OEE 2.0
Thursday	OEE 2.0	Web Services / Recipe Management
Friday	SPC, Instrument Interface	

MES Product Suite Overview

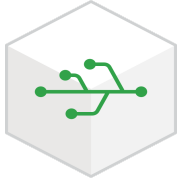


In this class, we'll cover....

- Product Suite Overview
 - Understanding the Driver for MES Projects
 - Risk Management and challenges faced with implementing MES projects
-

- Enterprise Architecture
- Interfacing to ERP and other Information Systems
- Setting up a Gateway server for MES

Track & Trace



In this class, we'll cover....

- ISA-95 Standard and Object Model
- Track & Trace Overview and the Material Flow
- MES Objects, Segments and Operations
- Production Model
- Mixed Nuts Practical
- Tracking data through components and scripting
- Using custom properties
- Inventory Management and Lot Tracking

OEE 2.0



In this class, we'll cover....

- What is OEE
 - Features of OEE 2.0
 - Framework
 - OEE Downtime 2.0 Settings Tab
 - Additional Factors and MES Counters
 - Downtime Detection Methods
 - Live Analysis
 - OEE Equipment Manager, Equipment Modes and States
 - Shift Management
-

- OEE Material Manager, Material Production Settings
- Packaging Nuts Practical
- Work Order Scheduling
- Using the Run Director component
- Using Live Analysis
- OEE Downtime Table component
- OEE Time Chart component
- Using the Analysis Selector and Controller components
- Parameterised Analysis
- Datapoints and Filters
- Creating Reports using Analysis data source
- Editing values using the Value Editor

Web Services



In this class, we'll cover....

- Web Technologies
- Web Service consumer practical
- Web Service provider

Recipe Management



In this class, we'll cover....

- Machine Settings vs. Batching
 - Adding Tags to the production model
 - Recipe Values
 - Recipe Editor
 - Master and Child Recipes
 - Loading Recipes by component and through scripting
 - Recipe Changes
-

- Variance Log
- Recipe practical

SPC



In this class, we'll cover....

- What is SPC
- SPC Samples
- SPC Variation
- SPC Values and Attributes
- SPC Control Charts
- Locations
- Tag Sample Collectors
- Creating Sample Definitions
- Adding Samples
- Scheduling Samples
- SPC Practical

Instrument Interface



In this class, we'll cover....

- Module Overview
 - Serial Settings
 - File Monitor
 - Parse Template
 - File Monitor Component
-

Table of Contents

1	Training Manual	6
1.1	Sections	6
2	Let's Get Set Up	8
2.1	Pre-Requisites	8
2.2	Install the MES Modules	8
2.3	Configure the Gateway	9
2.3.1	Configure the Database Connection	10
2.3.2	Configure the Gateway MES Settings	11
2.3.3	Configure the Production Simulator	12
2.3.4	Installing the Base MES Project	13
2.4	Installing Utility Screens	14
3	Track and Trace	21
3.1	ISA-95	21
3.2	Practical - Mixed Nuts	21
3.2.1	Creating the Material Flow Artifact	21
3.2.2	Configure Production Model	22
3.2.3	Configure Material Unloading	24
3.2.4	Configure Mixing	44
3.2.5	Create Inventory Operations	72
3.2.6	Copying and Deriving MES Objects	78
3.2.7	Operation Execution Through Scripting	84
3.2.8	Using Custom Properties	95
3.2.9	Tracking Personnel	100
3.3	Track and Trace Review	108
3.3.1	What We Covered	108
3.3.2	What We Didn't Cover	109
3.3.3	Additional Resources	110

4	OEE 2.0	111
4.1	Features	111
4.2	Framework	111
4.3	Practical - Package Nuts	111
4.3.1	Configure Packaging	111
4.3.2	Test Packaging	139
4.3.3	Add Lot Tracking	144
4.3.4	Add Sub Lot Tracking	154
4.3.5	Scheduling Operations	157
4.3.6	Analysis and Reporting	161
4.4	OEE 2.0 Review	171
4.4.1	What We Covered	171
4.4.2	What We Didn't Cover	172
4.4.3	Additional Resources	173
5	Web Services 2.0	174
5.1	Web Service Provider	174
5.1.1	Create Data Sources	175
5.1.2	Create getWorkOrders() Endpoint	177
5.1.3	Create getProductCodes() Endpoint	183
5.1.4	Create getProductCodeAttributes() Endpoint	186
5.1.5	Create postProductionResults() Endpoint	190
5.2	Web Service Consumer	195
5.2.1	Get Product Code List	195
5.2.2	Get Product Attributes	199
5.2.3	Get Work Order List	203
5.2.4	Post Production Results	205
5.3	Web Service Review	208
5.3.1	What We Covered	208
5.3.2	What We Didn't Cover	209
5.3.3	Additional Resources	209

6	Recipe Management	210
6.1	Configure Recipe Tags	210
6.2	Add Recipe Tags to Production Model	214
6.3	Create Process Recipes	215
6.4	Viewing Recipes Changes	216
6.5	Test Recipes with the Recipe Selector	218
6.6	Viewing Recipe Variances	220
6.7	Add Real Time Recipe Monitoring	221
6.7.1	Add Alarming	223
6.8	Add Recipe Selection to the Operations Segment	225
6.9	Using Analysis to Obtain Recipe Information	227
6.10	Recipe Review	228
6.10.1	What We Covered	228
6.10.2	What We Didn't Cover	229
6.10.3	Additional Resources	229
7	SPC	230
7.1	SPC Module Features	230
7.2	Application Of SPC	230
7.3	Control Charts	230
7.4	Control Limits	230
7.5	Gathering the SPC Requirements	231
7.6	Add SPC to Packaging Line	231
7.6.1	Add Checkweigher Tag Sample Collector	234
7.6.2	View Checkweigher Sample Data	237
7.6.3	Modify Packing Screen	238
7.7	Add SPC to Mixing Line	243
7.7.1	Create Sample Definition	243

7.7.2	Add Sample Attributes	244
7.7.3	Define Sample Locations	245
7.7.4	Add Control Limits and Signals	246
7.7.5	Add Mixing Samples	247
7.7.6	View Mixing Sample Data	248
7.8	Notification Of Out-Of-Control Processes	249
7.8.1	OPC Production Server Tags - SPC	250
7.9	SPC Review	254
7.9.1	What We Covered	254
7.9.2	What We Didn't Cover	255
7.9.3	Additional Resources	255
8	Instrument Interface	256
8.1	File Monitoring	256
8.2	Parse Template	257
8.3	File Monitoring Component	260
8.4	Using the Instrument Interface with the SPC Module	262
8.5	Instrument Interface Review	267
8.5.1	What We Covered	267
8.5.2	What We Didn't Cover	267
8.5.3	Additional Resources	267
9	Training Complete	268

1 Training Manual

Welcome to the MES 2.0 Training Manual! This is a comprehensive guide designed to help you improve the quality of your MES projects. It has been put together as a tutorial that walks you through the steps of implementing an MES solution for an imaginary company **Nuts Unlimited**. As we go through the steps, we'll add lot tracking, production scheduling, OEE metrics, recipe management, sampling and SPC to their manufacturing process. We'll use the utility modules to capture data from legacy devices and the Web Services module to create an information exchange with their ERP system to pull work orders and product codes.

You build the project as you go along and we'll discuss each topic in detail and provide links to reference material in the help manual. Whenever you see a panel with bullet points as shown, there is a task for you to perform to build out your own MES Project.

Example

- Drag a button onto the screen and name it **btnStartSimulator**

 Last updated - 2/8/2018

1.1 Sections

The tutorial is divided up into sections for each module. It is strongly recommended that you go through each section as there are dependencies between section exercises. Even if you are only interested in learning about one module, knowing the capabilities of each module and how they interact with each other will provide you with a better understanding of how best to realize your MES solution.

Section	Description	Time
Let's Get Setup	Install ignition, modules, database and configure gateway settings	30 mins
Track and Trace	Add Lot Tracking and Production Control	3hrs 30 mins
OEE 2.0	Add Work Order Scheduling and OEE analysis to production runs	5hrs

Section	Description	Time
Web Services 2.0	Connect to an ERP system to pull work orders and product codes, and return production data	1hr 30 mins
Recipe Management	Create a Recipe Management system to track changes to recipe and monitor production variances	3hrs
SPC	Add Statistical Process Control to monitor weight of finished goods packages	2hrs
Instrument Interface	Pull SPC data from lab instrumentation through flat file parsing	1hr
Total		16 hours 30 minutes

Good Luck with your training project!

» LET'S GET SET UP

- › Prerequisites
- › Install the MES Modules
- › Configure the Gateway
- › Install Utility Screens

2 Let's Get Set Up

We will need to set up an Ignition Gateway and database and then configure it. This is what we'll be using throughout the tutorial.



- Use a **clean Ignition install** and **new database** for this tutorial. We don't want remnants from old projects interfering with our training.
- This tutorial has been developed to walk you through building an MES solution using all our modules. It is not recommended to miss out sections as there are some dependencies between sections.

2.1 Pre-Requisites

You will need a computer with Ignition installed and access to a database server. If you do not have access to a database server, then follow the steps to install a MySQL database server, or you can use any of the other [databases supported by Ignition](#).

- [Download and Install Ignition](#) - Tutorial has been verified against Ignition version 7.9.6 and MES Modules version 2.9.2 RC1.
- [Download and Install MySQL Database](#) - Select the MySQL Community Installer MSI file. You will want to install the MySQL Server and MySQL Workbench.
- Open up **MySQL Workbench** and create a database called **mes**.

2.2 Install the MES Modules

Now we will install all the Sepasoft MES modules that will be used throughout the tutorial. We'll be building our application on our MES 2.0 platform, so all modules we download will be for that platform, except for the production simulator and instrument interface modules that we'll download from the original platform. Both these modules work on either platform.

- Download the following **MES 2.0** modules from the Sepasoft downloads page. Choose the latest 2.x version.
 1. Production Module
 2. Track & Trace
 3. OEE Downtime
 4. SPC
 5. Recipe
 6. Web Services
- Download the following **MES 1.0** modules from the Sepasoft downloads page. Choose the latest 1.x version.
 1. Instrument Interface
 2. Production Simulator
- Once the modules are downloaded, log in to the Ignition Gateway webpage and select **Configuration > Modules** from the menu.
- Install the modules (.modl files) one at a time by scrolling down to the bottom of the list and clicking on **Install or Upgrade a Module....**



→ Install or Upgrade a Module...

Note: For details about a module's status, see the [Module Status](#) page.


- Browse for your module and click **Install**.

2.3 Configure the Gateway

In this section, we will configure our Gateway to connect to the database we created, tell MES to use this database to store production data and create a production simulator device.

2.3.1 Configure the Database Connection

The MES modules create their own database schema to store configuration information and production data in. All you have to do is provide the database, create a database connection on the Ignition Gateway and let the MES modules know to use this database. You should have already created the database when you installed the Database instance in the [Let's Get Set Up](#) section. If not, open up **MySQL Workbench** and create a database called **mes**.

 To download and install MySQL, go to the MySQL website at <http://dev.mysql.com/downloads/mysql/>

We'll create a database connection in the Ignition Gateway configuration section. If you are using a different type of database server than MySQL, then you'll need to set it up accordingly. The instructions below are for MySQL.

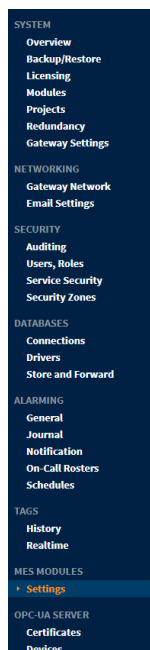
- Log in to Configuration and select **Databases > Connections** from the menu.
- Click on the **Create new Database Connection...** link to add a new connection.
- Choose the **MySQL ConnectorJ** for this database.
- Give the connection a name, specify the connection URL, and the credentials to use.
 - Name: MES
 - Connect URL: `jdbc:mysql://localhost:3306/mes`
 - Username: *select the username i.e. root*
 - Password: *select the password you setup for this user in mysql*
- Click on the **Create New Database Connection** button to finish creating the connection.

- Verify the connection is valid in the database status page before continuing to the next section. Now we have a database to store the MES data in.

2.3.2 Configure the Gateway MES Settings

The MES Modules store data in a SQL database. Because Ignition can be configured to use multiple databases, the MES Module Settings configuration page is used to select which databases will be used by the MES module(s) to store data in. If only one database has been configured in Ignition, then it will be selected by default.

- Log in to the Ignition Gateway configuration page and select **MES Modules > Settings** from the menu.
- In the Authentication section, select the user source profile that the MES system will use. For this training, we will use **default**.
- Set the **Runtime Database** to point to the MES database connection we created in the previous step.
- Set the **Analysis Database** to point to the MES database connection we created in the previous step.
- Save your changes to lock in those settings.



MES Module Settings

Authentication	
User Source Profile	<div>default</div> <div>The user source profile to use for MES modules.</div>

Runtime Datasource	
Runtime Database	<div>mes</div> <div>The database connection to store runtime production data. (Stop all production runs before changing this setting.)</div>

Analysis Datasource	
Analysis Database	<div>mes</div> <div>The database connection to store historical analysis production data to. Multiple sites can be set to the same analysis database to allow enterprise reporting. (Stop all production runs before changing this setting.)</div>
Analysis Database (Auxiliary)	<div>- none -</div> <div>The auxiliary or mirror database connection to store historical analysis production data to. (Stop all production runs before changing this setting.)</div>
Analysis Query Cache Duration	<div>300</div> <div>Number of seconds to cache analysis results. Increasing this setting will reduce the load on the database but, will delay the propagation of current production information to the analysis results.</div>
Analysis Read Lock Timeout	<div></div> <div>When executing analysis, the number of seconds to wait when the analysis engine configuration is being updated. The analysis engine configuration is updated when equipment, equipment modes, equipment states, counters or additional factors are changed.</div>

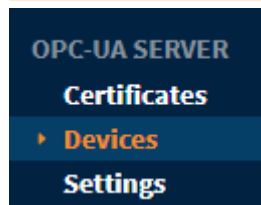
2.3.3 Configure the Production Simulator

Earlier on we installed the free production simulator module. This module will allow us to realistically simulate a production line and we'll use this as part of the training.


- In the configuration area of the Ignition Gateway, select **OPC-UA SERVER > Devices** from the menu.
- Click on the **Create new Device...** link.
- Select the **Production Simulator** driver.
- Name it **Simulator** and create the device.



Naming the Simulator device to something other than **Simulator** will require modifications to tag UDT OPC members later on in this tutorial.

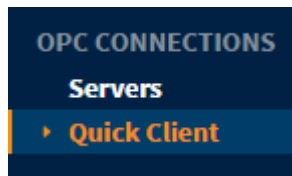


The Production simulator module is completely driven by the data stored in csv files.

- Navigate to the **C:\Program Files\Inductive Automation\Ignition\data\drivers** directory ('var/lib/ignition/data/drivers' for ubuntu). If the **drivers** directory doesn't exist, following may be the reason:
 1. Production simulator is not installed or running.
 2. Ignition may have been installed on an alternate disk. Try "D:" or another drive letter.
-  Download the following simulator files **plc_MixingLine1.csv**, **plc_PackagingLine1.csv**
- Navigate to the folder where the downloaded production simulator csv files are located and copy the two files to the **drivers** directory just created.
- Restart the production simulator module from the modules page on the gateway server, to pick up the mixing and packaging line tags. The production simulator will create tags from the csv files in the drivers folder on module startup.


OPC Quick Client

- Verify the simulator is working correctly by clicking on the **OPC Connections > Quick Client** in the configuration area. Locate the simulator device by expanding the tree starting from "Ignition OPC-UA Server." You should see a separate folder for the 2 CSV files.

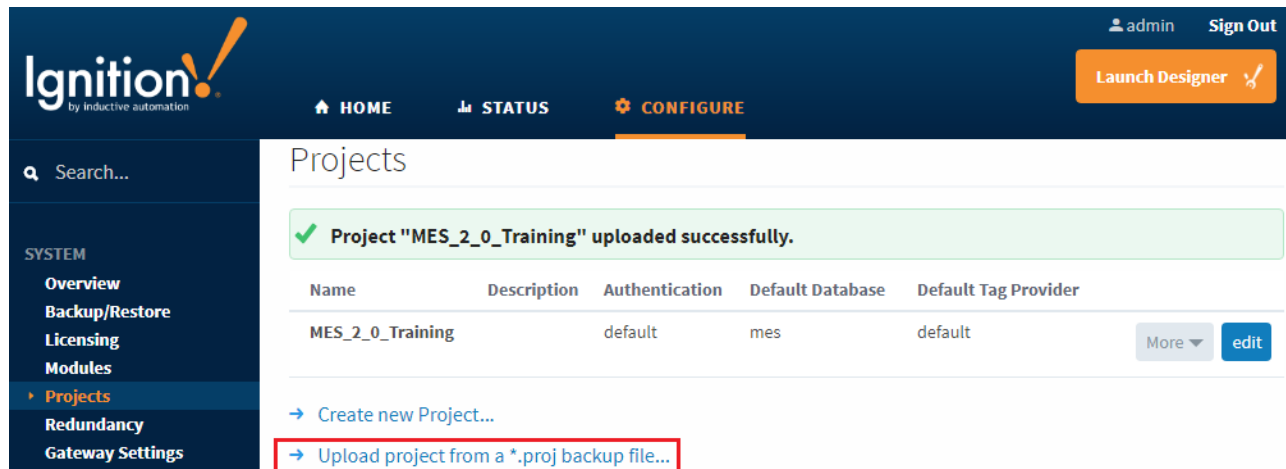


2.3.4 Installing the Base MES Project

We have provided a base project that contains a number of windows and templates to minimize the number of steps required to build out our project. These windows and templates that can be copied and used in your production system applications.

-  Click on [MES_2_0_Training - Base.proj](#) to download the base project.
- In the Ignition Gateway, select **Projects** under **Configure** and select **Upload project from a *.proj backup**.

If the Authentication, Default Database or Default Tag Provider is not set up as shown, edit the project to set it up correctly.



Ignition! by Inductive Automation

admin Sign Out

Launch Designer

HOME STATUS CONFIGURE

Search...

SYSTEM

- Overview
- Backup/Restore
- Licensing
- Modules
- Projects
- Redundancy
- Gateway Settings

Projects

✓ Project "MES_2_0_Training" uploaded successfully.

Name	Description	Authentication	Default Database	Default Tag Provider
MES_2_0_Training		default	mes	default

More edit

→ Create new Project...

→ Upload project from a *.proj backup file...


The base training project that we'll be using has links to images that we also need to download and import into our project.

- Launch the designer and open the **MES_2_0_Training** project.
-  Click [here](#) to download the images zip file.
- Unzip the downloaded file.
- Upload the **Sepasoft** folder using the **Tools->Image Management** menu item in the Ignition designer. Click the Upload icon  and navigate to the **Sepasoft** folder where the unzipped images reside. Select all images and select **Open**.

You have now completed the setup to be ready to start the MES training tutorial. Good Luck!

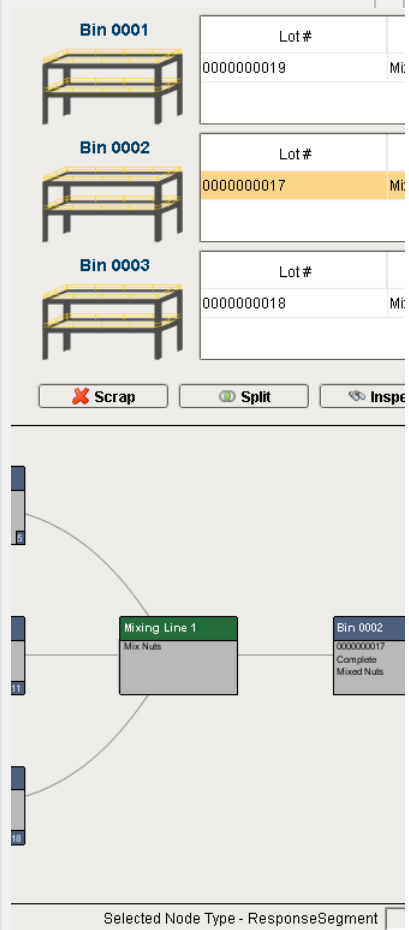

2.4 Installing Utility Screens

We have provided some utility screens, tags and scripts that can be used during the tutorial or in your own projects that provide added functionality or can help debug and troubleshoot issues. We will keep adding more stuff here so you should check back every now and then.

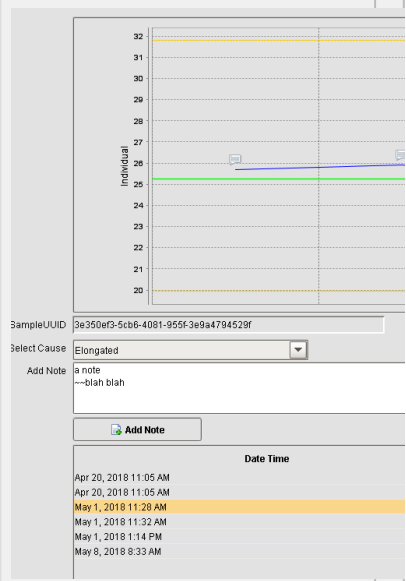
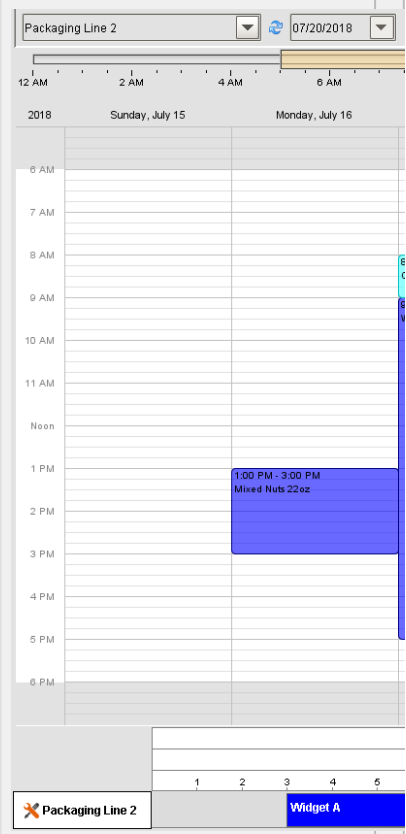
-  Download the following project files and import the screens and templates into your project through the Ignition Designer.

Module	Utility	Description	Updated																															
OEE 2.0 T&T	MES Diagnostics. proj	Diagnostics screen that can be used to see all active operations and segments under a site and end them if necessary.	4/17/18	<div><div>Folsom</div><div>Get Oper</div><table><thead><tr><th>Line</th><th>Path</th></tr></thead><tbody><tr><td>Nut Unloading</td><td>Nuts Unlimited\Folsom\Receiving\Nut U</td></tr><tr><td>Nut Unloading</td><td>Nuts Unlimited\Folsom\Receiving\Nut U</td></tr><tr><td>Almond Silo</td><td>Nuts Unlimited\Folsom\Receiving\Nut S</td></tr><tr><td>Bay 1</td><td>Nuts Unlimited\Folsom\Warehouse\Fini</td></tr><tr><td>Bay 2</td><td>Nuts Unlimited\Folsom\Warehouse\Fini</td></tr><tr><td>Finished Goods</td><td>Nuts Unlimited\Folsom\Warehouse\Fini</td></tr><tr><td>Folsom</td><td>Nuts Unlimited\Folsom</td></tr><tr><td>Mixing</td><td>Nuts Unlimited\Folsom\Mixing</td></tr><tr><td>Mixing Line 1</td><td>Nuts Unlimited\Folsom\Mixing\Mixing Li</td></tr><tr><td>Nut Storage Silos</td><td>Nuts Unlimited\Folsom\Receiving\Nut S</td></tr><tr><td>Peanut Silo</td><td>Nuts Unlimited\Folsom\Receiving\Nut S</td></tr><tr><td>Receiving</td><td>Nuts Unlimited\Folsom\Receiving</td></tr><tr><td>Walnut Silo</td><td>Nuts Unlimited\Folsom\Receiving\Nut S</td></tr><tr><td>Warehouse</td><td>Nuts Unlimited\Folsom\Warehouse</td></tr></tbody></table><div><div>Pens</div><div><input checked="" type="checkbox"/> Queries Per Second</div><div>Apply</div></div></div>	Line	Path	Nut Unloading	Nuts Unlimited\Folsom\Receiving\Nut U	Nut Unloading	Nuts Unlimited\Folsom\Receiving\Nut U	Almond Silo	Nuts Unlimited\Folsom\Receiving\Nut S	Bay 1	Nuts Unlimited\Folsom\Warehouse\Fini	Bay 2	Nuts Unlimited\Folsom\Warehouse\Fini	Finished Goods	Nuts Unlimited\Folsom\Warehouse\Fini	Folsom	Nuts Unlimited\Folsom	Mixing	Nuts Unlimited\Folsom\Mixing	Mixing Line 1	Nuts Unlimited\Folsom\Mixing\Mixing Li	Nut Storage Silos	Nuts Unlimited\Folsom\Receiving\Nut S	Peanut Silo	Nuts Unlimited\Folsom\Receiving\Nut S	Receiving	Nuts Unlimited\Folsom\Receiving	Walnut Silo	Nuts Unlimited\Folsom\Receiving\Nut S	Warehouse	Nuts Unlimited\Folsom\Warehouse
Line	Path																																	
Nut Unloading	Nuts Unlimited\Folsom\Receiving\Nut U																																	
Nut Unloading	Nuts Unlimited\Folsom\Receiving\Nut U																																	
Almond Silo	Nuts Unlimited\Folsom\Receiving\Nut S																																	
Bay 1	Nuts Unlimited\Folsom\Warehouse\Fini																																	
Bay 2	Nuts Unlimited\Folsom\Warehouse\Fini																																	
Finished Goods	Nuts Unlimited\Folsom\Warehouse\Fini																																	
Folsom	Nuts Unlimited\Folsom																																	
Mixing	Nuts Unlimited\Folsom\Mixing																																	
Mixing Line 1	Nuts Unlimited\Folsom\Mixing\Mixing Li																																	
Nut Storage Silos	Nuts Unlimited\Folsom\Receiving\Nut S																																	
Peanut Silo	Nuts Unlimited\Folsom\Receiving\Nut S																																	
Receiving	Nuts Unlimited\Folsom\Receiving																																	
Walnut Silo	Nuts Unlimited\Folsom\Receiving\Nut S																																	
Warehouse	Nuts Unlimited\Folsom\Warehouse																																	

Module	Utility	Description	Updated																					
OEE 2.0 T&T	MES Object Exporter. proj	<p>Screen that provides the ability to export and import MES objects (configuration information) between gateways. See Exporting MES Objects Between Gateways for more information.</p> <ul style="list-style-type: none">Added fix for Material Root object missing on new gateway to <i>visionWindowOpened</i> event scriptAdded support for <any person> personnelClass	5/23/18	<div><div><div>Export Objects</div><div>Import Objects</div></div><div>Select MES Object Type <div>Equipment Class</div><div><input type="checkbox"/> Select all objects of this type <input checked="" type="checkbox"/> Include Related MES Objects</div><div>Select MES Object <div>Mixed Nut Bins</div></div><div>MES Object UUID <div>201bbf70-940b-49a1-b3be-1a64...</div><div> Get Object(s)</div></div><table><thead><tr><th>Name</th><th>MES Object ...</th><th>Selected</th><th>Note</th></tr></thead><tbody><tr><td>Mixed Nut Bins</td><td>EquipmentCl...</td><td><input checked="" type="checkbox"/></td><td></td></tr><tr><td>Bin 0002</td><td>Equipment</td><td><input checked="" type="checkbox"/></td><td></td></tr><tr><td>Bin 0003</td><td>Equipment</td><td><input checked="" type="checkbox"/></td><td></td></tr><tr><td>Bin 0001</td><td>Equipment</td><td><input checked="" type="checkbox"/></td><td></td></tr></tbody></table></div></div>	Name	MES Object ...	Selected	Note	Mixed Nut Bins	EquipmentCl...	<input checked="" type="checkbox"/>		Bin 0002	Equipment	<input checked="" type="checkbox"/>		Bin 0003	Equipment	<input checked="" type="checkbox"/>		Bin 0001	Equipment	<input checked="" type="checkbox"/>	
Name	MES Object ...	Selected	Note																					
Mixed Nut Bins	EquipmentCl...	<input checked="" type="checkbox"/>																						
Bin 0002	Equipment	<input checked="" type="checkbox"/>																						
Bin 0003	Equipment	<input checked="" type="checkbox"/>																						
Bin 0001	Equipment	<input checked="" type="checkbox"/>																						
ALL	WIW.proj	'Window In Window' templates. Allows you to create a window within a window using templates and passing parameters as a python dictionary.	1/26/18	<div><div>Downtime</div><div>Run Chart</div><div>Tra</div></div> <div><div><div><div>Walnut Silo</div><div>W1000</div><div>Complete</div><div>Bulk Walnuts</div><div>1</div><div>11</div></div><div><div>Almond Silo</div><div>A1001</div><div>Complete</div><div>Bulk Almonds</div><div>1</div><div>1</div></div><div><div>Peanut Silo</div><div>P1000</div><div>Complete</div><div>Bulk Peanuts</div><div>1</div><div>11</div></div><div><div>Mixing L</div><div>Mix Nut</div><div>1</div><div>11</div></div></div><div>Selected Node Type - ResponseSegment</div></div>																				

Module	Utility	Description	Updated	
T&T	Inventory Functions. proj	<p>Templates that provides a standard set of inventory functions:</p> <ul style="list-style-type: none"> • Scrap Material • Split Material • Inspect Material • Adjust Quantity <p>This template also requires the operations to be created or imported (MES Object Export Operations Definition - Inventory Functions.xml). See the Create Inventory Operations page).</p> <ul style="list-style-type: none"> • Added support for multiple operations running 	3/12/18	 <p>Bin 0001</p> <p>Lot # 0000000019</p> <p>Bin 0002</p> <p>Lot # 0000000017</p> <p>Bin 0003</p> <p>Lot # 0000000018</p> <p>Scrap Split Inspect</p> <p>Mixing Line 1</p> <p>Mix Nuts</p> <p>Bin 0002</p> <p>0000000017</p> <p>Complete Mixed Nuts</p> <p>Selected Node Type - ResponseSegment</p>
ALL	Date Range Selector. proj	<p>Template that allows for easy selection of preset date ranges. Pop this onto a screen to drive start and end dates. Bind the Start Date and End Date template properties to date properties on the windows' root container or to client tags and then bind other components date properties to the client of container properties.</p>	4/16/18	 <p>12 Today From: 02/05/20</p>

Module	Utility	Description	Updated																																		
SPC	Quality Check Table Report.proj	SPC data primarily works with the provided control charts, however if you are using the module simply to schedule and capture quality checks, and want to display the data in a power table or report, this project shows you how to achieve this through scripting.	5/3/18	<div><div>Select locationLine 1 Quality</div><div>Select SampleQuality Checks</div><div>From05/01/2018 08:08 AM</div><div>To05/05/2018 08:08 AM</div><div>Get Samples</div></div> <table><thead><tr><th>Date</th><th>Visual Inspection</th><th>K</th></tr></thead><tbody><tr><td>May 1, 2018 8:55 AM</td><td><input checked="" type="checkbox"/></td><td></td></tr><tr><td>May 2, 2018 3:40 PM</td><td><input checked="" type="checkbox"/></td><td></td></tr><tr><td>May 3, 2018 7:55 AM</td><td><input checked="" type="checkbox"/></td><td></td></tr></tbody></table>	Date	Visual Inspection	K	May 1, 2018 8:55 AM	<input checked="" type="checkbox"/>		May 2, 2018 3:40 PM	<input checked="" type="checkbox"/>		May 3, 2018 7:55 AM	<input checked="" type="checkbox"/>																						
Date	Visual Inspection	K																																			
May 1, 2018 8:55 AM	<input checked="" type="checkbox"/>																																				
May 2, 2018 3:40 PM	<input checked="" type="checkbox"/>																																				
May 3, 2018 7:55 AM	<input checked="" type="checkbox"/>																																				
OEE 2.0	Equipment State Importer.proj	<p>Can be used to import equipment states and downtime reasons/codes from a csv file generated from a plc or exported from an OEE 1.0 project. csv files only needs to contain state name and state code. This screen allows you to define the equipment state type for each state.</p> <ul style="list-style-type: none">Added check for blank state names and invalid characters in state names before adding them	6/18/18	<div><div>Select CSV File</div><div>Filler Cell Downtime reasons example.csv</div><table><thead><tr><th>Reason Name</th><th>Reason Code</th><th></th></tr></thead><tbody><tr><td>Stop</td><td>0</td><td>true</td></tr><tr><td>Running</td><td>1</td><td>false</td></tr><tr><td>CO2 out of spec</td><td>2</td><td>true</td></tr><tr><td>Machine Fault</td><td>3</td><td>true</td></tr><tr><td>Blocked</td><td>4</td><td>false</td></tr><tr><td>Starved</td><td>6</td><td>false</td></tr><tr><td>CIP</td><td>8</td><td>false</td></tr><tr><td>Over Temperature</td><td>9</td><td>true</td></tr><tr><td>Container Jam</td><td>22</td><td>true</td></tr><tr><td>Bypassed</td><td>9000</td><td>false</td></tr></tbody></table><div><div>Select State Name Column0Reason Name</div><div>Select State Code Column1Reason Code</div></div></div>	Reason Name	Reason Code		Stop	0	true	Running	1	false	CO2 out of spec	2	true	Machine Fault	3	true	Blocked	4	false	Starved	6	false	CIP	8	false	Over Temperature	9	true	Container Jam	22	true	Bypassed	9000	false
Reason Name	Reason Code																																				
Stop	0	true																																			
Running	1	false																																			
CO2 out of spec	2	true																																			
Machine Fault	3	true																																			
Blocked	4	false																																			
Starved	6	false																																			
CIP	8	false																																			
Over Temperature	9	true																																			
Container Jam	22	true																																			
Bypassed	9000	false																																			

Module	Utility	Description	Updated	
SPC	Custom Causes.proj	SPC Control Charts provides a method for adding notes and 'cause ' to sample data. If you need to provide a specific list of selectable causes by sample /attribute type, then this screen shows how to implement that functionality.	5/10/18	
OEE 2.0 T&T	Calendar Scheduler.proj	<p>Operations scheduler using the Ignition Calendar components to create and modify Operations Schedule objects.</p> <ul style="list-style-type: none"> Added support for displaying and editing equipment modes at the line level 	7/13/18	

Module	Utility	Description	Updated	
OEE 2.0	Line Run Data.proj	Updated Live Analysis Production Run Template. Requires Live Analysis 'Run Data' and global script 'getLiveAnalysisTagPath()' to be created (see instructions on Create Packaging Screen).		<div> <div>Line</div> <div>Batch #</div> <div>Product Co</div> </div> <div> <div>Packaging Line 1</div> <div></div> <div>Mixed</div> </div> <div> <div>Line Mode</div> <div>Line State</div> </div> <div> <div>Production</div> <div>Running</div> </div>


- **Save** the changes.

» TRACK & TRACE

- › ISA-95
- › Practical—Mixed Nuts
- › Track & Trace Review

3 Track and Trace

In this section, we'll be using the Track & Trace module to provide production control and lot tracking for our Nut Mixing process.

-  Download and import the [MES_2_0_Training - Base TT.proj](#). This contains some pre-built screens and templates that we will use during the tutorial.
- **Save** your changes.

Before we dive in though, let's take some time to familiarize ourselves with the Track & Trace module and the ISA-95 standard on which our modules are built on.

3.1 ISA-95

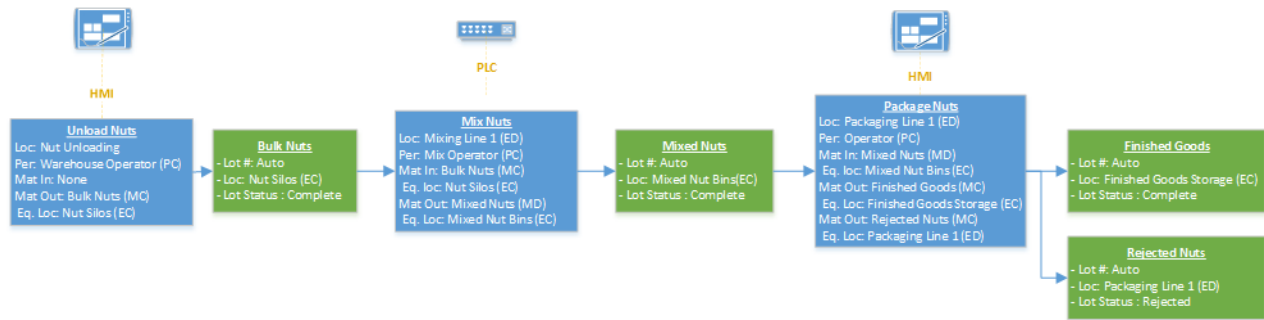
The Track & Trace and OEE 2.0 modules have been developed against the ISA-95 standard. Unless you are a paid member of the ISA-95 standards committee, you would do well to familiarize yourself with the overview below of what the ISA-95 standard is comprised of. Much of the terminology used in the training and help regarding process segments, operations definitions and resources are taken from ISA-95 definitions.

3.2 Practical - Mixed Nuts

Now that we are well-versed in what the Track & Trace module can do and what ISA-95 is all about, we'll start using it to build an MES solutions for a manufacturing company called Nuts Unlimited. In this practical, we will create the materials, equipment and operations to model the material flow for the mixed nuts manufacturing process.

3.2.1 Creating the Material Flow Artifact

Creating a material flow diagram is a critical first step in implementing a Track & Trace solution. Understanding the processes, equipment, material lots consumed and created, as well as touch points to other information systems will shape how we develop the production model, material classes and definitions, process segments and operations. The diagram below shows the material flow for our Nuts Unlimited project.

















We have already created the material flow for the online tutorial for you. Never miss doing this step when you are implementing Track & Trace for your project. It will help you avoid mis-steps, re-work, and our Tech Support and Design Consultants will ask to see it should you call us for help.

3.2.2 Configure Production Model

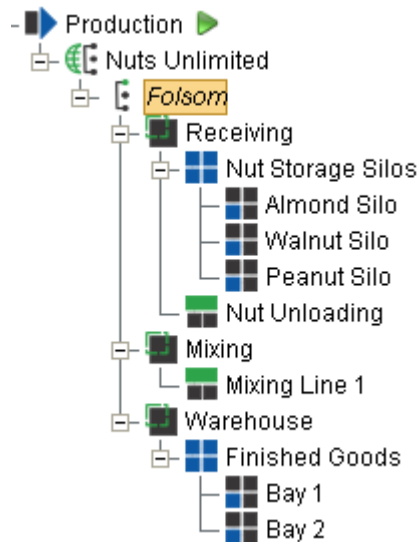
The production model allows us to model our plant as defined by the ISA-95 standard. For the Track & Trace section of this tutorial, we will start off by building out the Receiving, Mixing and Warehouse areas.

- Create the following production items in the production model to match the image shown.

Icon	Production Item	Value	Icon	Production Item	Value
	Enterprise	Nuts Unlimited		Area	Mixing
	Site	Folsom		Line	Mixing Line 1
	Area	Receiving		Area	Warehouse
	Storage Zone	Nut Storage Silos		Storage Zone	Finished Goods
	Storage Unit	Almond Silo		Storage Unit	Bay 1
	Storage Unit	Peanut Silo		Storage Unit	Bay 2
	Storage Unit	Walnut Silo			
	Line	Nut Unloading			

Don't worry about the settings for each as we will be coming back to the production model and changing settings as we proceed through the training. Make sure to save your changes in the designer.

- **Enable** the Production model in the Designer Production Model Enabled: ☒
- **Save** and **Publish** your project.



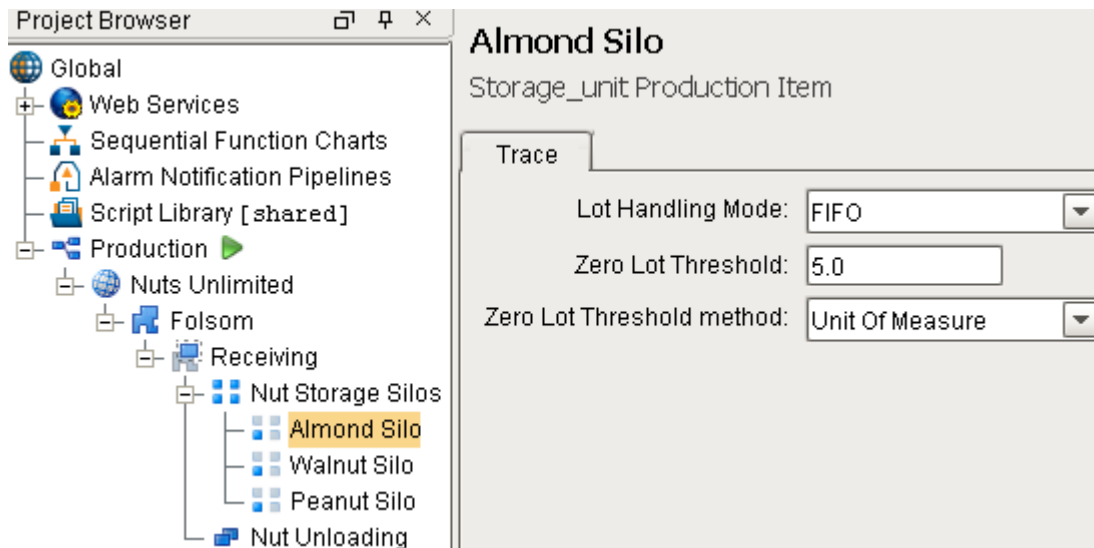
3.2.3 Configure Material Unloading

In this section we will be doing all the configuration necessary to unload nuts at the dock into the Nut Storage Silos. This will be the entry point of material into the tracking system. We'll create a Nut Unloading window that an operator can use to unload nuts and record the material received.

Configure Storage Silo Lot Handling Mode

When material lots are stored or moved to equipment, different lot handling modes can be used. By default, equipment lot handling is set to 'random lot' which means multiple lots of different material can be stored and consumed in any order. For our training project, we will set the Nut Silos **lot handling mode** to **FIFO**. This will force us make sure the first lot stored in the silo is used first and the second lot used second and so on. The Zero Lot Threshold will cause the system to automatically zero the lot if not all of the lot was used from the silo. This cleans up a potentially large number of leftover lots with very small quantities remaining. See the section below for more information about lot handling modes.

- Change the settings for the **Almond Silo**, **Walnut Silo** and **Peanut Silo** Production Items by clicking on the Production Item in the production model and selecting the Trace tab, to the following:
 - Lot Handling Mode: **FIFO**
 - Zero Lot Threshold: **5.0**
 - Zero Lot Threshold Method: **Unit Of Measure**
- **Save** your changes in the designer.

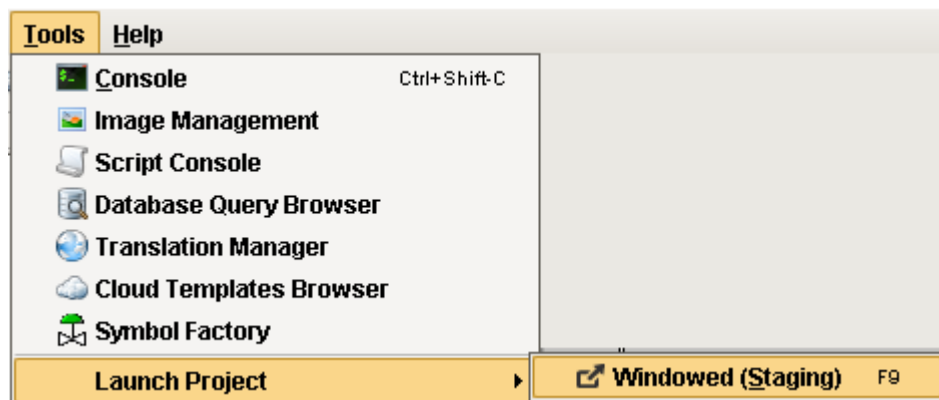


Configure Unloading Equipment

When nuts are unloaded, it involves two different types of equipment. First we need to know where the Unloading Nuts operation is being done. In our case, it will be at the **Nut Unloading** Station that we defined in the Production Model. Second, we need to know where we are storing the nuts. The **Almond Silo**, **Peanut Silo** and **Walnut Silo** storage units we defined in the Production Model will be used for this purpose.



We will create an Equipment Class called **Nut Silos** that will itself contain all of our Nut Silo Storage Units. We can do this in the client using the [MES Object Editor](#) component.

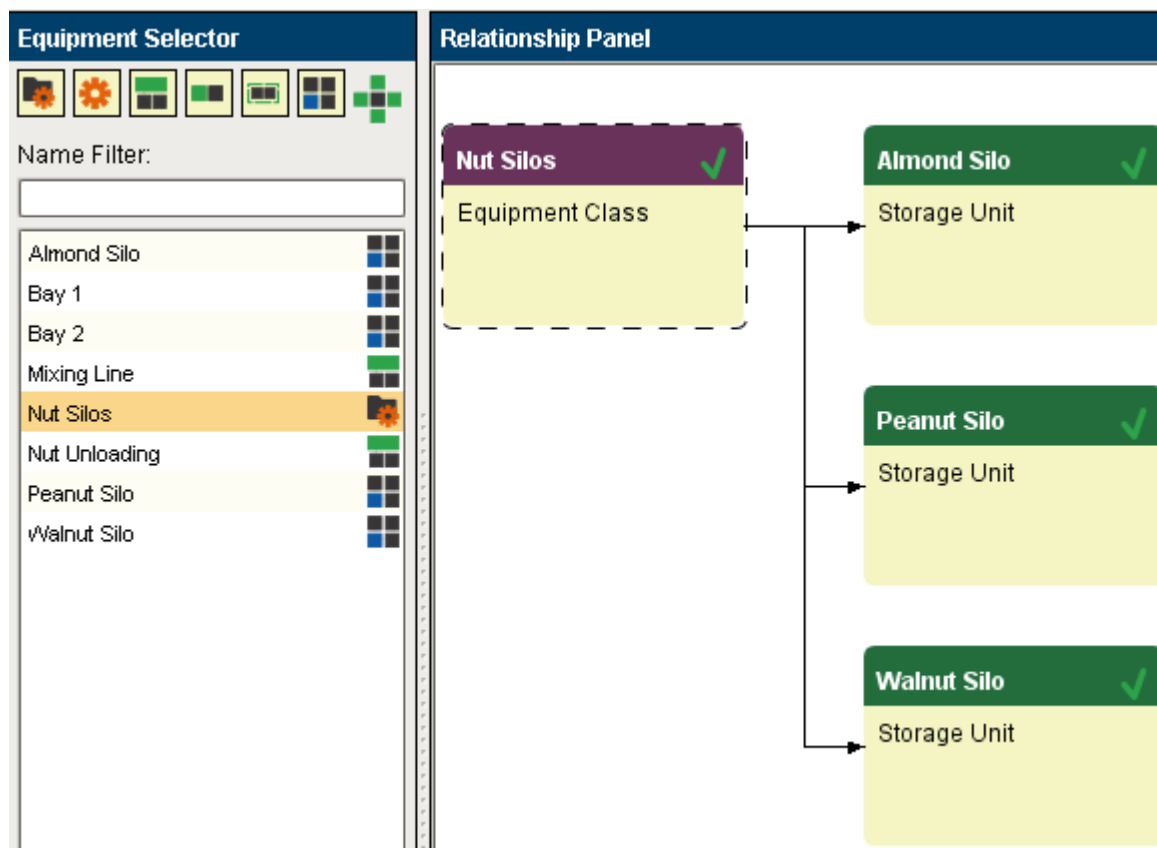
- In the designer, go to **Tools > Launch Project > Windowed (Staging)** or press **F9**.
- Open the **Administration/ MES Management** window. (It might also be called MES Object Editor).



- Select the **Equipment** tab.





Notice that some of the Production Items from the Production Model are listed. Additional supplemental equipment such as pallets, bins, dies, etc can be added here. We will do this in a later section.

- Click on the  icon and select **Add Equipment Class**.
- Enter **Nut Silos** into the name field.
- Click the **Save** button.
- Left-click the **Almond Silo** storage unit, hold the mouse button and hover over the **Nut Silos** Equipment Class until you see the  icon. Release the mouse when you reach the icon.
- Do the same for **Peanut Silo** and **Walnut Silo** storage units. When dragging the silos over, release when it is directly on top of the plus sign that appears. You should see the same as the image.



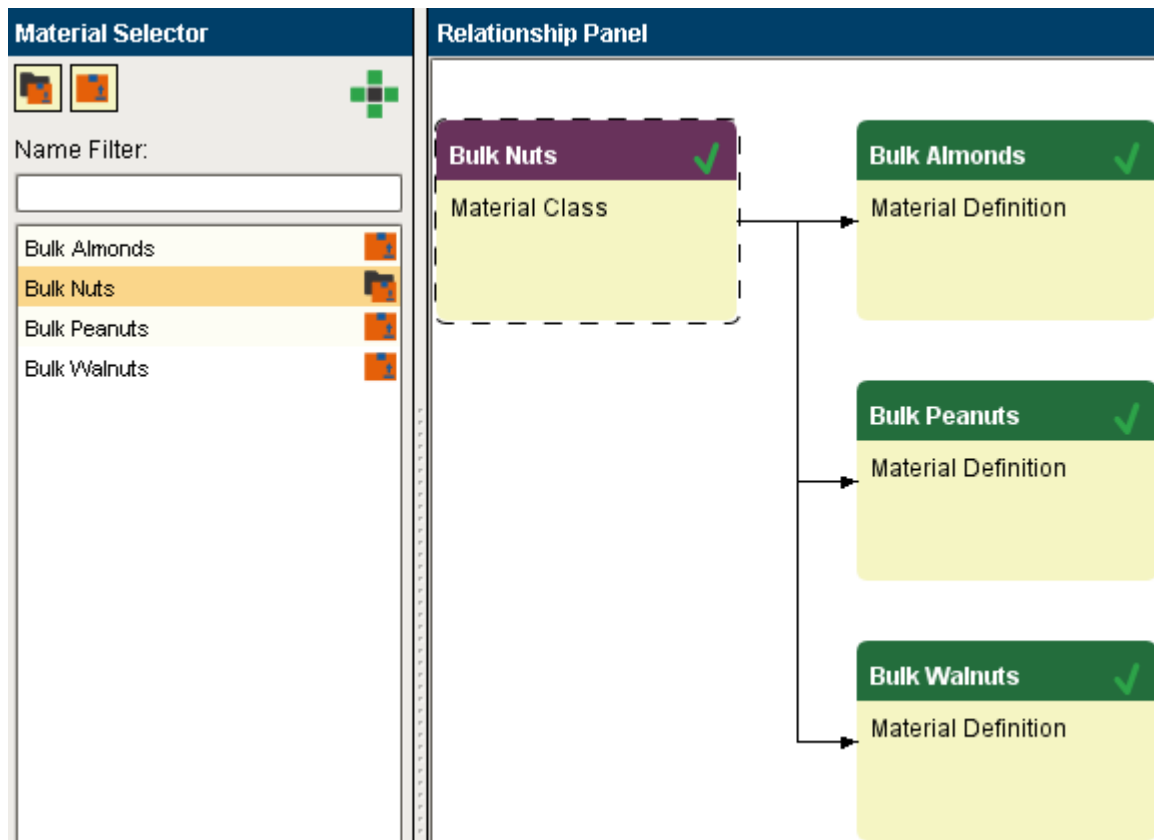
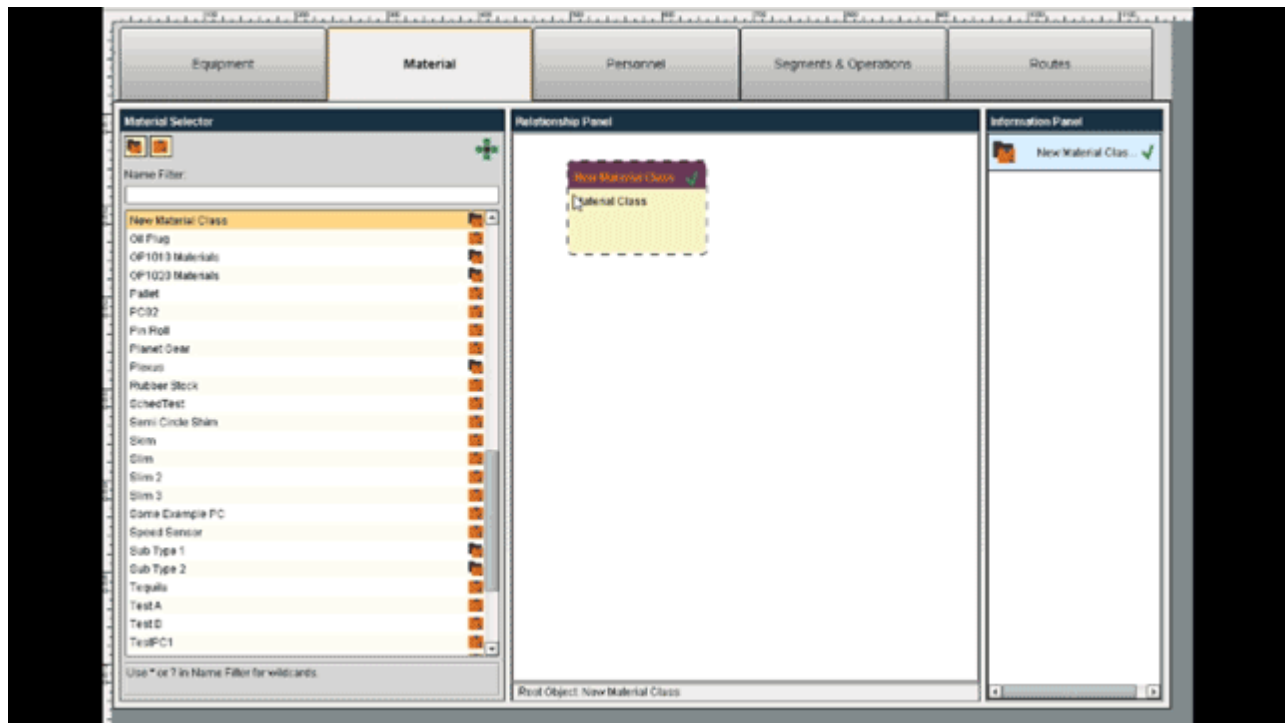
Create Raw Material Objects

Now that the equipment has been setup for unloading nuts, the material that will be unloaded must be defined .

- Select the **Material** tab in the MES Manager.
- Click on the  icon and select **Add Material Class**. Enter **Bulk Nuts** into the name field and click the **Save** button.
- Click on the  icon and select **Add Material Definition**. Enter **Bulk Almonds** into the name field and click the **Save** button.
- Click on the  icon and select **Add Material Definition**. Enter **Bulk Peanuts** into the name field and click the **Save** button.
- Click on the  icon and select **Add Material Definition**. Enter **Bulk Walnuts** into the name field and click the **Save** button.
- Select **Bulk Nuts** in the selector list on the left hand side of MES Manager.
- Left-click and hold the mouse button on the **Bulk Almonds** material definition and drag it to the **Bulk Nuts** node.
- Do the same for **Bulk Peanuts** and **Bulk Walnuts** material definitions. You should see the same as the image shown.

We have now created material definitions for all the bulk nuts that we have received and we put them in a class called **Bulk Nuts**. The class will allow us to associate a process segment with any material definitions that are in this class.

In case you struggle with the interface, here is a GIF of the click-drag-drop process to associate Materials with a Material Class.

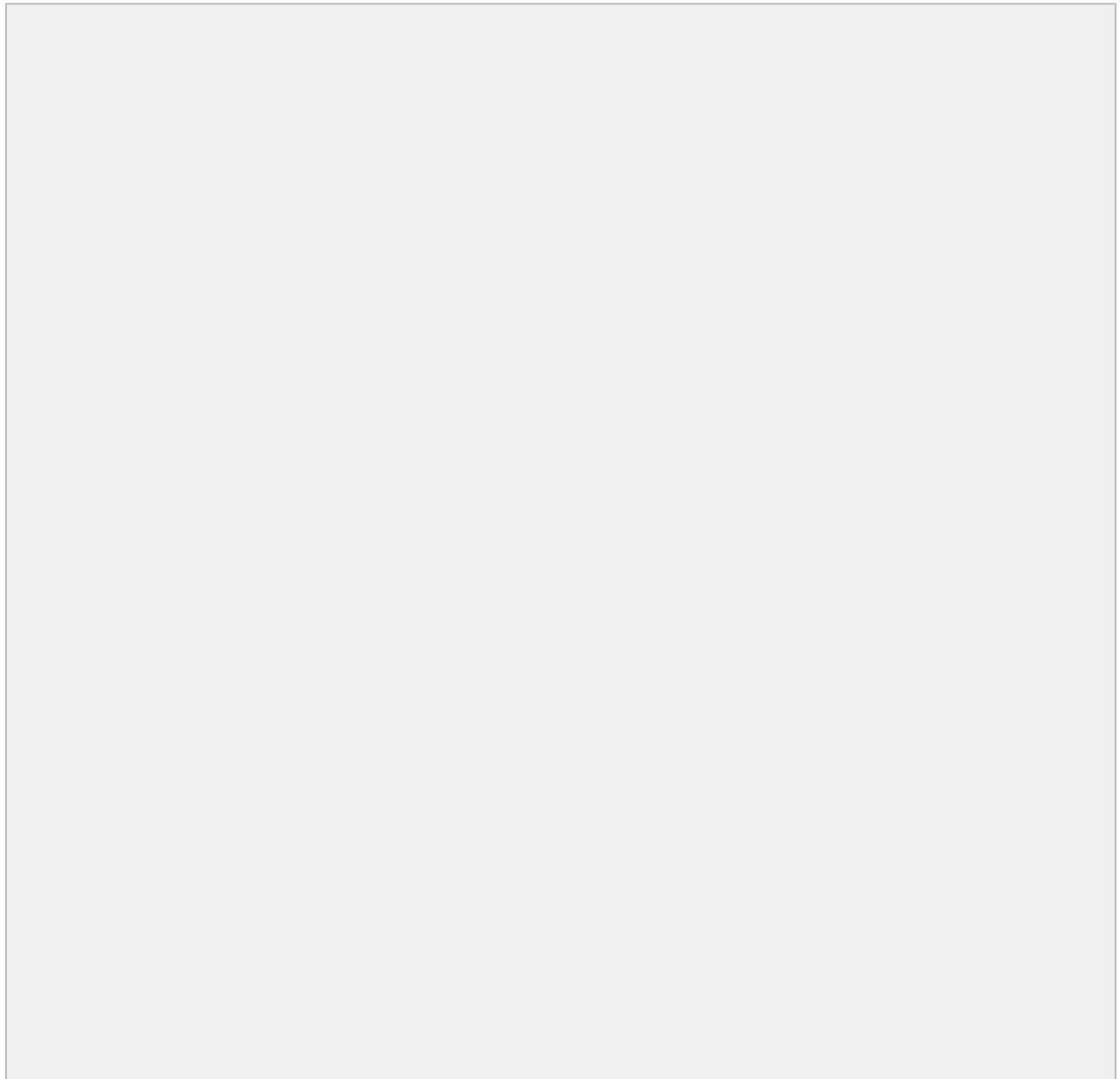



Refer to the [Create Material Objects Through Scripting](#) section in the help for more details on using scripting to create these objects.

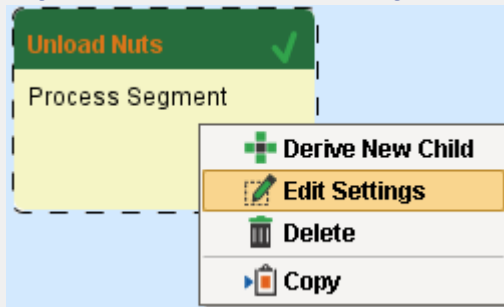
Create Unload Nuts Process Segment

Now that the resources needed to unload nuts were defined in the prior sections, a process segment to unload nuts can be defined.

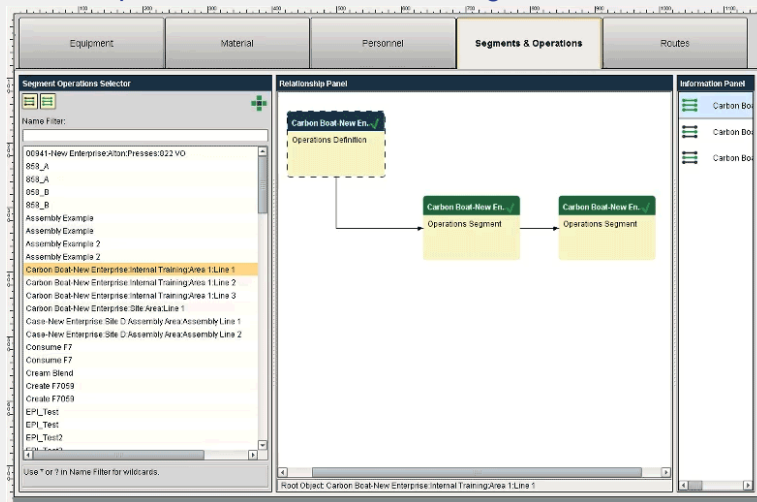
We'll create a segment that represents the production task of unloading nuts using the following steps:



- Select the **Segments & Operations** tab in the MES Manager.
- Click on the  icon and select **Add Process Segment**.
- Enter **Unload Nuts** into the name field.
- Right-click on the Process Segment and select **Edit**.



- Select the **Material** header and click on the plus icon to add a new material resource.
- UI Example: Add a Resource to a Segment








- Change the **name** property for the material resource to **Received Nuts**. This is the name that we can reference this material resource by in script.
- Change the **Use** property to **Out**. This tells the system that this material will be created as a result of this segment (production task).
- Check the **Auto Generate Lot** property box to **true**. This tells the system to create a new material lot object for this material.
- Double-click on the **Material Reference** property and a selection dialog will appear. Select **Material Class** and then **Bulk Nuts** then click the OK button.

Remember that the 'Bulk Nuts' material class was created in a previous section and contains Bulk Almonds, Bulk Peanuts and Bulk Walnuts material definitions. By selecting the 'Bulk Nuts' material class here, this segment will allow unloading Bulk Almonds, Bulk Peanuts and Bulk Walnuts but will prevent unloading any other material.

- Double-click on the **Lot Equipment Reference** property and a selection dialog will appear. Select **Equipment Class** and then **Nut Silos** then click the OK button.

Remember that the 'Nut Silos' equipment class was created in a previous section and contains Almond Silo, Peanut Silo and Walnut Silo Storage Units. By selecting the 'Nut Silos' equipment class here, this segment will allow storing the material to the Almond Silo, Peanut Silo and Walnut Silo Storage Units, but will prevent storing the material at any other equipment.




- Change the **Lot Number Source** property to **Manual**. This tells the system that the lot number of material being unloaded will be entered manually or with script.
- Change the **Quantity Source** property to **Manual**. This tells the system that the quantity of material being unloaded will be entered manually or with script. Enter **Lbs** in the **Units** field.
- Change the **Rate Period** property to **None**. This is used for scheduling segments which is not being covered at this time.

[-] Material	
[-] Received Nuts	Out, Material Class, Bulk Nuts
Name	Received Nuts
Optional	<input type="checkbox"/>
Production Selectable	<input checked="" type="checkbox"/>
Use	Out 
Auto Generate Lot	<input checked="" type="checkbox"/>
Material Reference	Material Class, Bulk Nuts
Lot Equipment Reference	Equipment Class, Nut Silos
Enable Sublots	<input type="checkbox"/>
Lot Number Source	Manual 
Lot Number Source In Link	
Quantity Source	Manual 
Quantity Source Link	
Quantity	
Units	lbs
Rate Period	None 
Rate	
Cycle Time	0
Final Lot Status	
Auto Lot Quantity Completion	Disabled 
Material Lot Depletion Warning	0

- Select the **Equipment** header.
- Click on the plus icon to add a new equipment resource.
- Change the **name** property for the equipment resource to **Station**. This is the name that we can reference this equipment resource by in script.
- Click on the Equipment Reference property, select **Line** and then **Nut Unloading** and click OK. This is the Production Item (equipment) that the Unload Nuts process segment can be run at. The image shows the equipment settings you should see at this point.
- Click **Save** and select **Yes** when asked **Create operation for this process segment?**

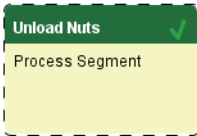
Equipment	
Station	Line, Nut Unloading
Name	Station
Equipment Reference	Line, Nut Unloading
Use	
Quantity	0.0
Units	
Custom Properties	

You can check that the process segment has been setup correctly by ensuring you have check marks against the segment, equipment and material in the information panel as shown. Hover over the properties shown on the information panel to get more information if a warning symbol is shown.


Segment Operations Selector
  
Name Filter:


Unload Nuts


Unload Nuts

Relationship Panel


Information Panel

 Unload Nuts ✓

 Station ✓

 Received Nuts ✓

A required new material lot will be created with a manually assigned lot number going to Nut Silos that contain Bulk Nuts material

Quantity will be manually entered as null units.

No production rate is selected.

The final lot status will be set to Complete.

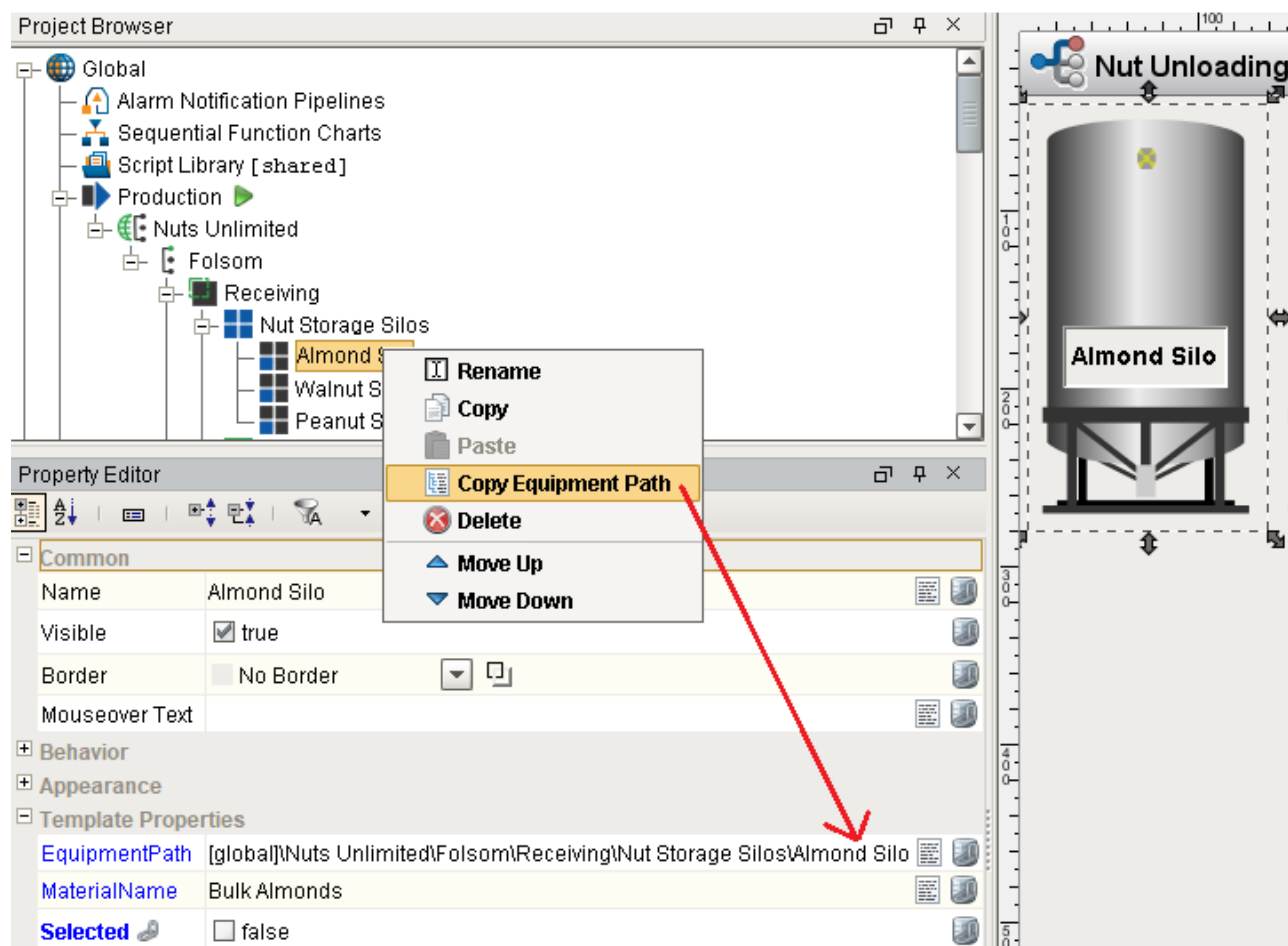
Now that the resources have been defined and the segment is created, let's move on to building a window to utilize them.

Configure Unloading Screen

We will now create a window that will allow the operator to select the material being unloaded, enter a raw material lot number and a quantity. This window will use the resources and segment defined in the previous sections.

- In the designer, open the **Nut Unloading** window in the Unloading folder that was imported from the base MES training project.
- In the Production model, right-click on the **Almond Silo**, select **Copy Equipment Path** and paste it into the *EquipmentPath* property of the Almond Silo component.
- In the Production model, right-click on the **Peanut Silo**, select **Copy Equipment Path** and paste it into the *EquipmentPath* property of the Peanut Silo component.
- In the Production model, right-click on the **Walnut Silo**, select **Copy Equipment Path** and paste it into the *EquipmentPath* property of the Walnut Silo component.

We now have three silo templates that we can use to select which silo we want to unload nuts to. Each silo has a script on the *mousePressed* event that sets the root container *SelectedMaterialName*, *SelectedSiloName* and *SelectedSiloPath* to the selected silo.



User Input and Segment Execution

After an operator has selected the silo that he wants to unload nuts to, we'll need to provide a method for them to enter in a lot number and a quantity. When we defined the **Unload Nuts** process segment, we said that this information would be manually entered.

- Drag the **Nut Unloading Pane** template onto the window.

This template simply provides the ability to input a lot number and a quantity that we will use with an Unload Nuts operation. The template has four parameters that we will pass it:

- **eqPath** - Copy the equipment path of the **Nut Unloading** line we created under Receiving in the production model
- **materialName** - Bind this to the window *Root Container.SelectedMaterialName*
- **siloPath** - Bind this to the window *Root Container.SelectedSiloPath*

Template Properties	
eqPath	[global]\Nuts Unlimited\Folsom\Receiving\Nut Unloading
materialName	Bulk Walnuts
siloPath	[global]\Nuts Unlimited\Folsom\Receiving\Nut Storage Silos\Walnut Silo

The **Record** button uses scripting to execute the **Unload Nuts** operation that we defined earlier. The script creates a response segment passing it parameters for the silo, lot number and quantity based on the user input. The code is already in the button *action performed* script and is here for reference to show how an operation segment can be executed through the use of scripting. Later on when we create the Packing Nuts screen, we will show how operation segments can be executed using components.

```

unloadStaPath = event.source.parent.parent.eqPath #This is where the
nuts are being unloaded
siloPath = event.source.parent.parent.siloPath #Get the silo where
the unloaded nut are being stored
matName = event.source.parent.parent.materialName #Get the material
name for the selected nuts
lotNumber = event.source.parent.getComponent('txtLotNumber').text
#Make sure this lot number doesn't already exist
try:
    system.mes.getLotInventoryByLot(lotNumber)
    system.gui.messageBox("The lot number " + lotNumber + " has
already been used. Please select another")
except:
    unloadQuantity = event.source.parent.getComponent('numQty').
intValue
    seg = system.mes.createSegment('Unload Nuts', unloadStaPath, True
) #Create a new Unload segment
    seg.setMaterial('Received Nuts', matName, siloPath, lotNumber,
unloadQuantity) #Set the material information for the nuts that are
being unloaded
    seg.execute() #Execute the segment. This will begin and
immediately end the segment in one call

    #Reset all of the user selections
    event.source.parent.getComponent('txtLotNumber').text = ''
    event.source.parent.getComponent('numQty').intValue = 0

```

Displaying WIP Inventory

Before we actually try this out and receive any nuts, we'll want to display the WIP inventory in the silos as material is received and consumed. We'll use an Ignition power table component and get the data from binding functions provided by the Trace module.

- Add a **Power Table** component to the Nut Unloading window below the silos.
- Add binding to the Power Table component's **Data** property. Click the **Functions** binding under **Other**.
- Select **Equipment WIP** binding function.
- Enter 'Nut Silos' as the name of the equipment class that we want to get the WIP for and then click **OK** to complete the binding.
- Right-click on the table component and use the **Table Customizer** to hide all columns except the **Lot Number**, **MaterialName**, **In Quantity**, **Out Quantity**, **Net Quantity** and **Units** columns.
- Go to **preview** mode by pressing **F5** to display the values in the power table.

The Nut Unloading window is now complete and ready for testing.

Property Binding: Root Container.Power Table

Binding Types

Tag

☒ Tag

☐ Indirect Tag

☐ Tag History

Property

☐ Expression

☐ Property

Database

☐ Named Query

☐ DB Browse

☐ SQL Query

Other

☐ Cell Update

☒ Functions

☒ No Binding

Binding Function

Equipment WIP

Equipment Path, Supplemental Equipment Name, or Equipment Class Name

Nut Silos

Lot Number Filter (optional)

Lot Status Filter (optional)

Polling Mode

☐ Off ☒ Relative ☐ Absolute

Polling Rate

Rate = (Base Rate) +/- 0 sec

Retain Rows

☐ false

OK Cancel

Test Unloading

The Nut Unloading window, resources and segment were created in the previous sections. Now we can move on to testing.

- Go into preview mode by pressing **F5**.
- Select the **Almond Silo**.
- Enter a raw material lot number such as **BA 1001**.
- Enter a **quantity**. (Our mixing operation will need 200lbs of each type of nut, so make sure you unload enough nuts).
- Click the **Record** button.

The inventory should show the BA 1001 lot number and quantity.

- Unload a number of lots of almonds, peanuts and walnuts into all three silos several because we'll use these in the mixing section.

Now that we have collected a bit of data, we can use the Trace Graph to visually see how the data is tracked.

- Open the **Trace Graph** window located in the **Analysis** folder and select one of the lot numbers used to unload nuts to see the results.
- Hover over the **Nut Unloading** node in the Trace graph. You'll see information displayed on the left popup about the Nut Unloading operation.

We see a simple graph showing the unloading process. Each box on the trace graph is called a node. The Nut Unloading node (green) represents the segment (production task) that was executed. The Almond Silo node (blue) represents the material lot as a result of the Nut Unloading segment.

You can hover on the Unload Nuts node (green) to view trace information for that operation. You can view any raw material and any finished good and anything along the way. You can also click on the **Table** tab to view the data in tabular form.


Each node can be configured to add custom user menu items. You can use this to extend the trace graph functionality to bring back any data associated with that lot or process that was not captured as part of Track & Trace. An example would be to bring back OEE or SPC data or tag process history data. For more information on creating custom user menu items, see the `UserMenuItemClicked` eventHandler in the [MES Trace Graph](#) Help for more details.



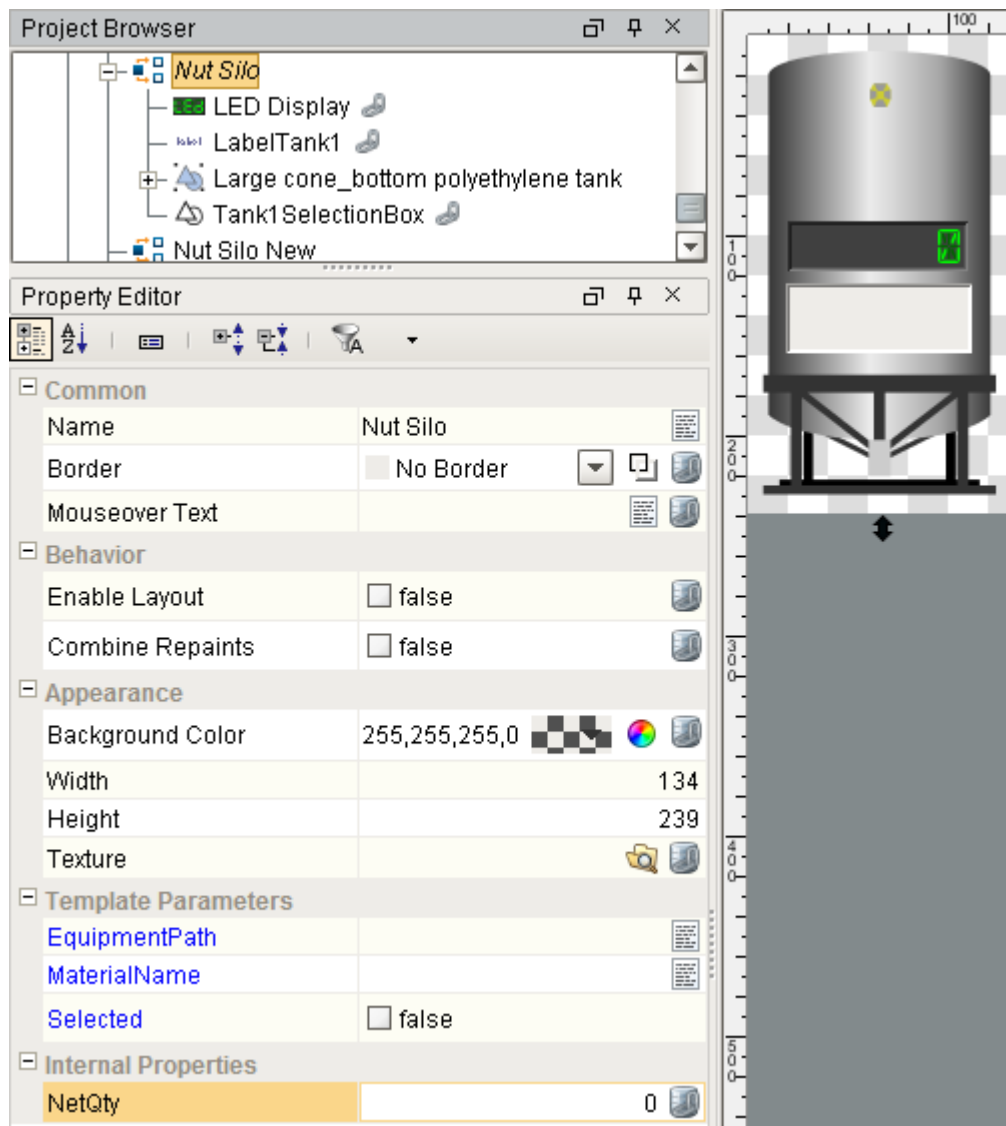
The colors of the different node types is configurable and can be modified by changing the Node configuration dataset of the Trace graph object. See [MES Trace Graph Help](#) for more details.

Display WIP Inventory at Silos

Earlier we added a WIP Inventory Power Table to the nuts unloading screen, that used a binding function to show us which lots were in each silo. Let's add another visual indicator of how many nuts we have in each silo as reported by Track & Trace. In a real world application, this would probably come from a level sensor, but this will help us validate that we are tracking quantities correctly and give us a chance to look at how we can obtain current inventory through scripting functions.

- Click on the  menu item for the Nut Silo template and add a new internal property called **NetQty** of data type integer.
- Now add a LED display component in the Nut silo template and bind the **Value** property to the new **NetQty** property.
- Change the Number format of the LED display to **#0**.

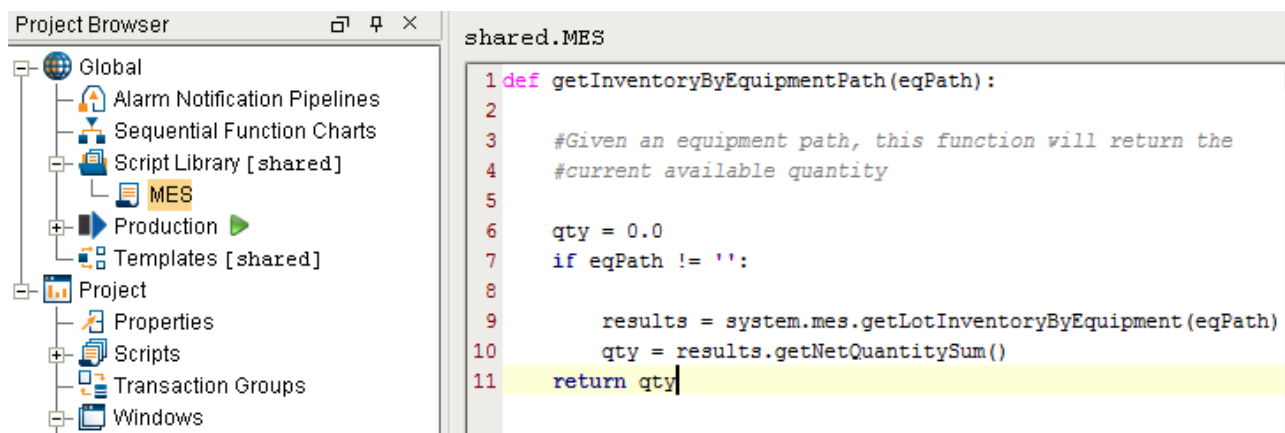
*The **NetQty** property will be used to get the current quantity in this silo.*



- Create a new global script called MES and add a function called `getInventoryByEquipmentPath()`. You can copy the following code into the MES script. Then, commit, save and publish.

```
def getInventoryByEquipmentPath(eqPath):
    #Given an equipment path, this function will return the
    #current available quantity
    qty = 0.0
    if eqPath != '':

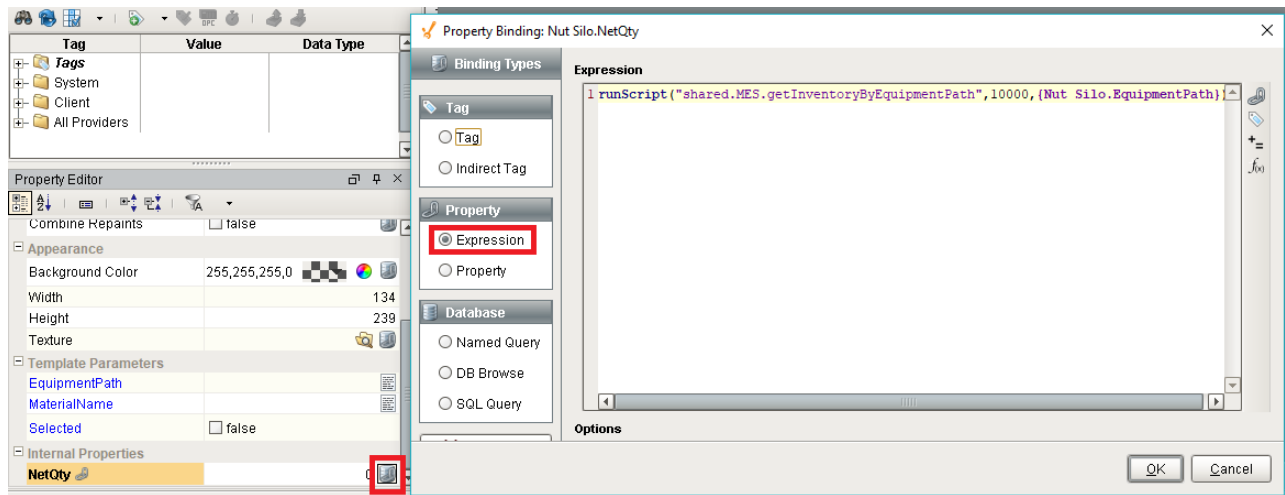
        results = system.mes.getLotInventoryByEquipment(eqPath)
        qty = results.getNetQuantitySum()
    return qty
```



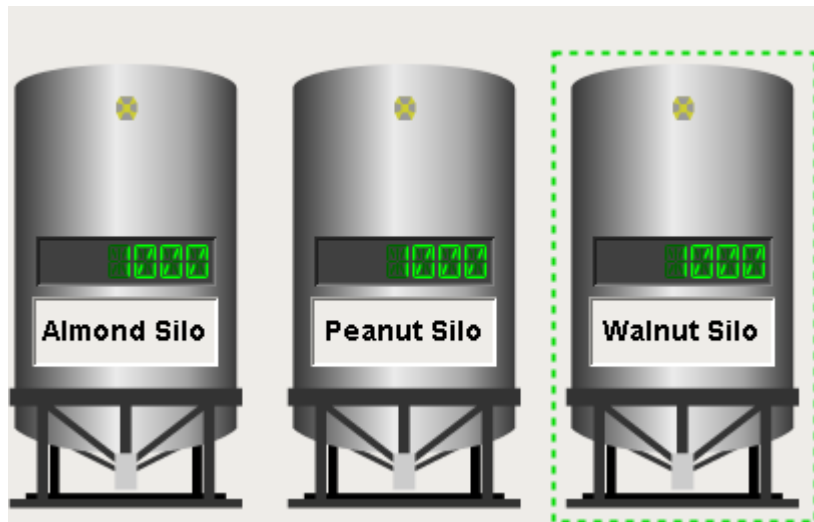
- In the binding of the Nut Silo template **NetQty** internal property, add the following script as shown in the screenshot:

```
runScript("shared.MES.getInventoryByEquipmentPath", 10000, {Nut
Silo.EquipmentPath})
```

The runScript expression will call our global script every ten seconds passing it the equipment path to our nut silo and obtaining the current inventory for this silo.



If everything went according to plan, we should have total values for all our silos. So what did we exactly do here? We created a global script using the [system.mes.getLotInventoryByEquipment](#) function and called it from an expression bound to a custom property of our nut silo template. We could also have used the [MES Lot Filter](#) object to define the lots we are interested in and passed that filter object to the functions [system.mes.getInventory\(\)](#) or [system.mes.getLotList\(\)](#). Alternatively, we could have used [system.mes.loadMaterialLot\(\)](#) if we were interested in a specific material lot. There are plenty of ways to obtain lot information, the trick is deciding the best approach for your implementation.



3.2.4 Configure Mixing

In this section we will mix the received nuts together and store them in bins on the production floor. Bins are used to demonstrate how to use [supplemental equipment](#) for material storage. Supplemental equipment is defined using the [MES Object Editor](#) in the client rather than through the production model in the designer. The Mixing Operation will use the material lots of bulk nuts that we just unloaded into the nut silos. We'll start and stop the Nut Mixing operation using a tag UDT ([User Defined Types](#)) and we'll use the production simulator that we setup earlier to simulate signals coming from the mixer PLC.

Configure Mixing Equipment


There are three pieces of equipment that are needed for the Mixing Operation.

- The **nut silos** where the input material will come from have already been configured in the previous section.
- **Mixing Line 1** will be the equipment where the mixing will be done.
- The equipment where the output material will be stored.

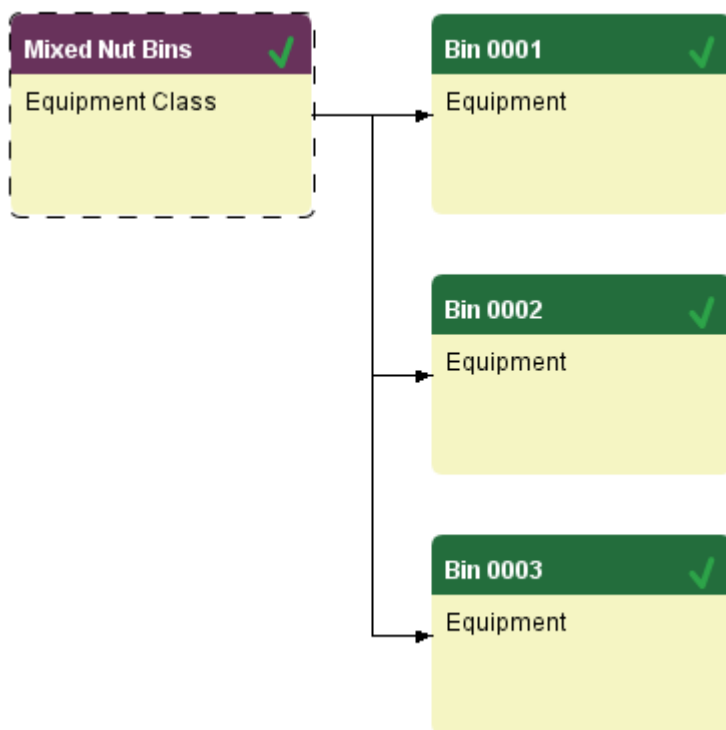
✔ Supplemental Equipment

In ISA-95, location is considered as **Equipment** which is defined as a physical fixed location. Equipment Definitions include Fixed location, Storage zone, Storage Unit, Lines and Cells. Equipment Definition does not include mobile equipment. Fixed locations are defined through the Production Model designer. Mobile storage locations are defined through the MES Management screen.

We will now define these mobile storage bins by following the steps below to store the mixed nuts that are produced by the Mixing operation..

- In the client, navigate to the **MES Object Editor** window located in the Administration folder.
- Click on the **Equipment** Tab and click on the  icon and select **Add Equipment Class**.
- Enter **Mixed Nut Bins** into the name field and click **Save**.
- Click on the green plus icon and select **Add Supplement Equipment**.
- Enter **Bin 0001** into the name field.
- Change the **Lot Handling Mode** to **Single Lot** to only allow one lot in the bin at a time
- Select **Zero Lot Threshold Method** to **Unit Of Measure** and **Zero Lot Threshold** to **0.0** and click **Save**.
- Repeat steps 4 through 7 to add **Bin 0002** and **Bin 0003**.
- Select the **Mixed Nut Bins** class node. Left click and hold the mouse button on the **Bin 0001** supplement equipment name in the list and drag it to the **Mixed Nut Bins** node.
- Do the same for **Bin 0002** and **Bin 0003** supplemental equipment. You should see the same as the image.

This is all that is needed for the equipment. The next step is to define the material needs for mixing nuts.






If you get an error that the personnel class <any person> cannot be found, you'll need to create it using the MES Object Editor.

Configure Mixing Material


Now that the equipment has been setup, we will define the materials. The **Bulk Almonds**, **Bulk Peanuts** and **Bulk Walnuts** material definitions were created in the unloading section, but the **Mixed Nuts** material definition still needs to be created. Because for our training purposes there is only one **Mixed Nuts** material definition, there is no need to create a material class. **Unloading Nuts** provided an example of a material class so it will not be used here.

- Select the **Material** tab in the **MES Object Editor**.
- Click on the  icon and select **Add Material Definition**.
- Enter **Mixed Nuts** into the name field.
- Click the **Save** button.

The resources are all setup, now we will move on to the next step of creating a segment for mixing.

Create Mixing Segment

In this section, we will create a process segment called **Mix Nuts**, associate this operation with the **Mixing Line 1** equipment, and then add materials references for the three input materials (**Almonds**, **Peanuts** and **Walnuts**), and the output material (**Mixed Nuts**).

- Select the **Segments & Operations** tab in the MES Manager and click on the  icon and select **Add Process Segment**.
- Enter **Mix Nuts** into the name field.




Refer to the [Creating Process Segment](#) in the Help Manual for more information on the segment fields and options

Add Materials Reference

The Material reference property defines the materials that will be consumed and created by this process segment.

- For each input material, Almonds, Peanuts and Walnuts, and also the output material of this process segment, Mixed Nuts, we will add a material reference. Use the table below to set the appropriate material reference property values. Not every property of the material reference needs to be set, except for what is defined in the table. Refer to [Operations Segment](#) for more details of the Material Reference property settings.

Property	Almonds	Peanuts	Walnuts	Mixed Nuts	Comment
Name	Almonds In	Peanuts In	Walnuts In	Mixed Nuts Out	
Use	In	In	In	Out	Defines the input and output materials from this process segment.
Auto Generate Lot	False	False	False	True	Input material lot #'s will be entered or selected. Output material will have a lot # generated for it.
Material Reference	Material Definition - Bulk Almonds	Material Definition - Bulk Peanuts	Material Definition - Bulk Walnuts	Material Definition - Mixed Nuts	Only allow input materials of specific types. Only mixed nuts can be created.

Property	Almonds	Peanuts	Walnuts	Mixed Nuts	Comment
Lot Equipment Reference	Storage Unit - Almond Silo	Storage Unit - Peanut Silo	Storage Unit - Walnut Silo	Equipment Class - Mixed Nut Bins <div>  If you are using 2.9.1 SP2 version of MES, please set the Equipment to <any equipment>. </div>	Restrict incoming materials to their respective silos. Output material can only go to storage bins.
Lot Number Source	Auto	Auto	Auto	Auto	For incoming materials, use the lot id of the selected material. For output material, a lot id will be automatically generated because of Auto Generate Lot setting.
Quantity Source	Manual	Manual	Manual	Link Combine	For incoming materials, quantity consumed by this process will be manually entered. For output material, Mixed Nuts, quantity will be all quantity sources labled 'Total' combined.
Quantity Source Link	Total	Total	Total	Total	

Property	Almonds	Peanuts	Walnuts	Mixed Nuts	Comment
Rate Period	None	None	None	None	
Units	lbs	lbs	lbs	lbs	

Add Equipment Reference

The Equipment Reference property defines where this process segment can be executed.

- Use the table entries below to set up the Equipment reference properties.

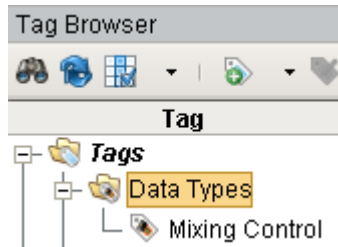
Property	Value	Comment
Name	Mix Station	This is the name that we can reference this equipment resource by in script.
Equipment Reference	Line - Mixing Line 1	This is the Production Item (equipment) that the mix segment can be run at.

- Hit the **Save** button.
- Select Yes when asked **Create operation for this process segment?**

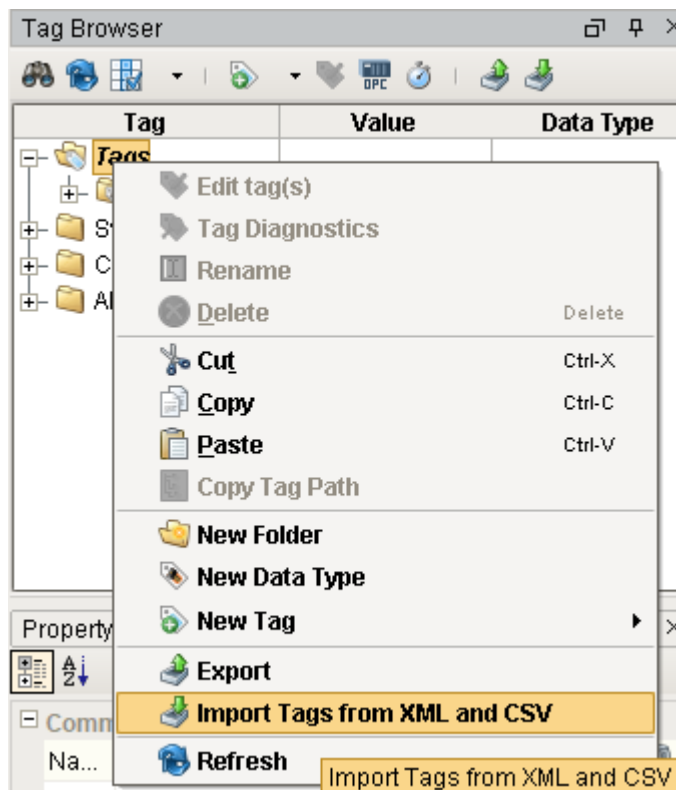
Configure Mixing UDT

In this section we'll use a tag UDT ([User Defined Type](#)) to interface to the Mixer PLC simulator we setup earlier. We'll add script to it that will control the starting and stopping of the **Mix Nuts** operation as well as starting the line simulator and obtaining production counts. This UDT can be used for any number of mixing lines that follow the same logic. The next section will cover creating an instance of this UDT to control MixingLine1. For more help, see [Creating User Defined Types](#).

-  Download the **Mixing Control UDT.xml** and **import** this file by right-clicking on the **Tags** folder in the Tag Browser in the designer.



You will now see the **Mixing Control** tag UDT in the **Data Types** folder under the root **Tags** folder in the Tag Browser. The tag UDT has three Data Type Parameters that we can set when we create an instance of this tag UDT. These parameters allow us to re-use this UDT by simply changing the parameters when we create an instance of the UDT to point to specific plc tags. The OPC tags then use the **Device** and **plc** parameter values to point to the right plc tags.



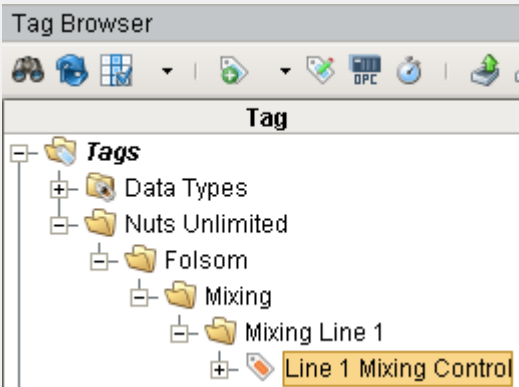
The **Running** tag of the Mixing Control UDT contains a tag change event script that will start and stop the production simulator as well as the **Mix Nuts** operation.

Configure Mixing Tags

In the previous section, we created a UDT which we'll now use to create an instance that will control the mixing operation for Mixing Line 1. The members of the UDT will be linked to the Simulator created earlier.

Use the following steps to create an instance of the **Mixing Control UDT** to control Mixing Line 1.

- In the **Tag Browser**, create the following new folder structure
Nuts
Unlimited\Folsom\Mixing\Mixing Line 1
- Create an instance of the **Mixing Control UDT** by right-clicking on the **Mixing Line 1** folder and using the menu items as shown.

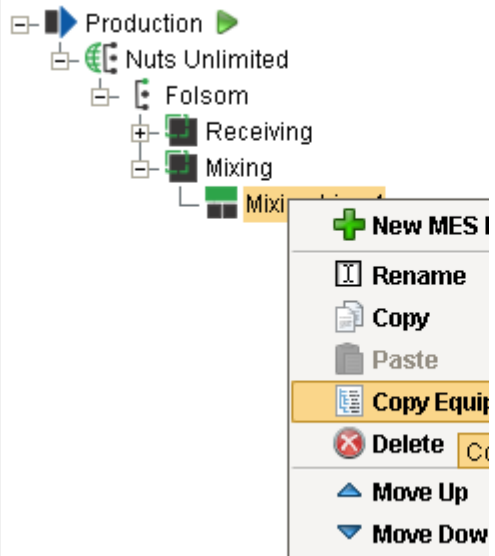
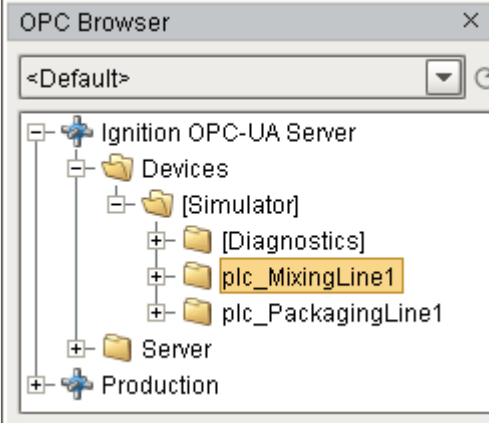


- Set the **Data Type Properties** as follows:

Field	Value	Description

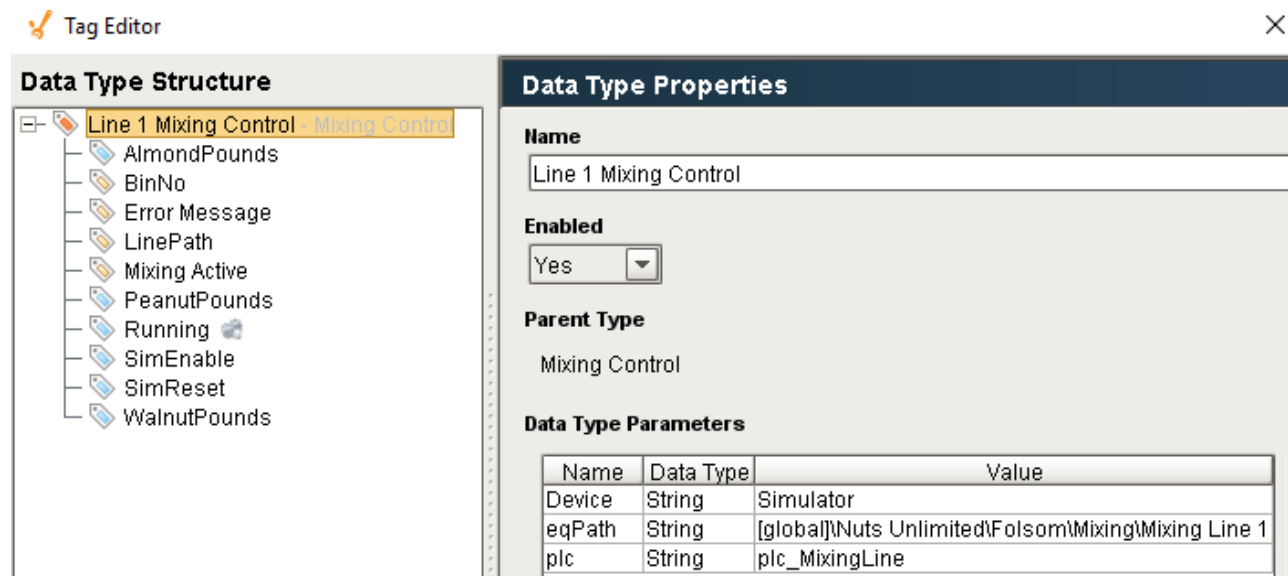
Field	Value	Description													
Name	Line 1 Mixing Control		<div><div>Data Type Parameters</div><table><tr><th>Name</th><th>Data Type</th><th></th></tr><tr><td>Device</td><td>String</td><td>Simulator</td></tr><tr><td>eqPath</td><td>String</td><td>[global]\Nuts Unlimited</td></tr><tr><td>plc</td><td>String</td><td>plc_MixingLine1</td></tr></table></div>	Name	Data Type		Device	String	Simulator	eqPath	String	[global]\Nuts Unlimited	plc	String	plc_MixingLine1
Name	Data Type														
Device	String	Simulator													
eqPath	String	[global]\Nuts Unlimited													
plc	String	plc_MixingLine1													
<div>Data Type Parameters</div>															
Device	Simulator	<p>Set this to the name you gave the Production Simulator in the OPC-UA Devices section. If you named your simulator OPC device to something other than Simulator when you setup the simulator earlier, then you will set the <i>Device</i> parameter to whatever you called it.</p>	<div><div>Devices</div><table><tr><th>Name</th><th>Type</th></tr><tr><td>Simulator</td><td>Production</td></tr></table><div>→ Create new Device...</div></div>	Name	Type	Simulator	Production								
Name	Type														
Simulator	Production														

Field	Value	Description
eqPath	[global]\Nuts Unlimited\Folsom\Mixing\Mixing Line 1	Set this to the equipment path of Mixing Line 1 (right click on Mixing Line 1 in the production model and choose 'Copy equipment path').
plc	plc_MixingLine1	This is used in the OPC Item Path indirect binding for the OPC tags

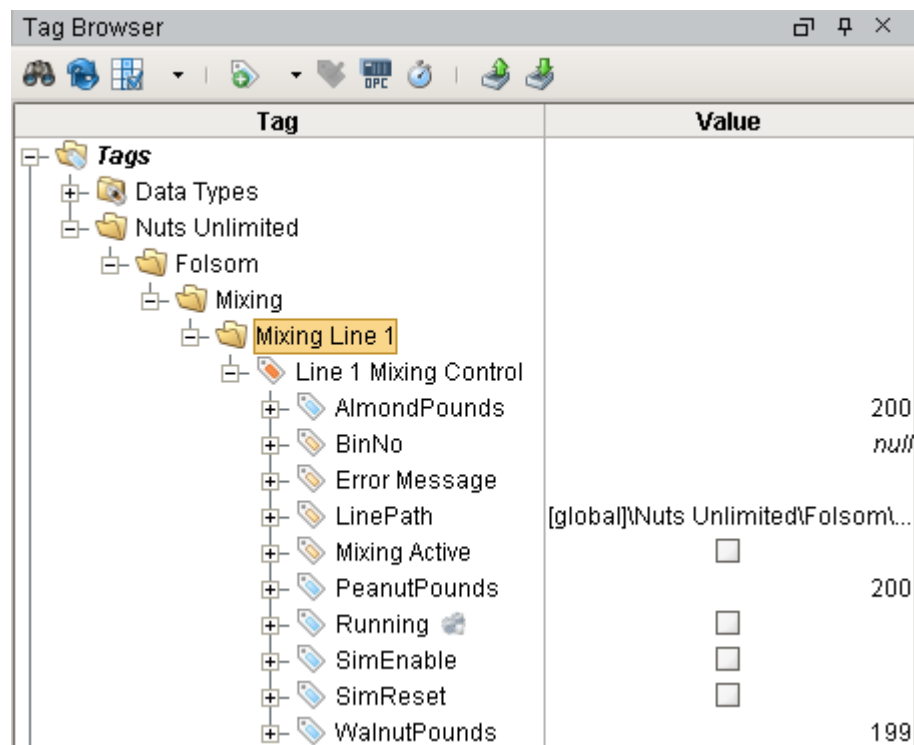



- Save by clicking **OK**.



Your tag UDT should now be setup and displaying data from the simulator (as shown in the image).

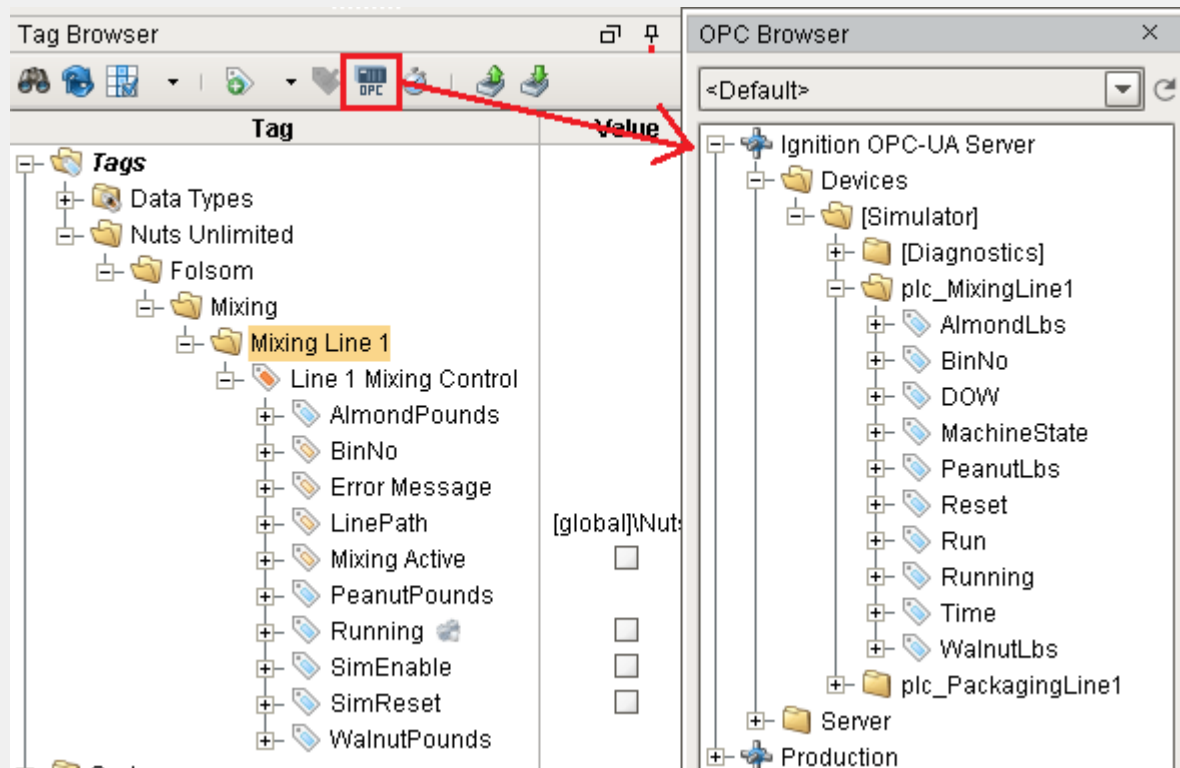


The nice part with using a UDT here is we didn't have to create any script for this specific instance for Line 1 Mixing Control. Dozens of instances for different mixing lines can be created without scripting them individually. If you have tens or hundreds of lines, this is a very effective way of rolling out and maintaining an application.



Problems?

If you see  on any of the tags, it's likely that your path to the Simulator OPC tags is not correct. To troubleshoot, click on the  button in the Tag Browser pane to bring up the OPC Browser pane and drag one of the simulator tags back over to the Tag Browser pane. Now compare the OPC path between that tag and the faulted UDT tag.



Configure Mixing Screen

We will now create a screen that will allow us to start the Mix Nuts operation and monitoring what is happening. This window will use the process segment, UDT and tags that have been created so far.

- Open the **Nut Mixing** window in the Mixing folder.
- Right-click on the **Line 1 Mixing Control** UDT and choose **Copy Tag Path**. Select the *tagUDTPath* custom property of the **Nut Mixing** root container and paste in the tag path.
- For each silo on the Mixing screen, copy the associated equipment path from the production model to the *EquipmentPath* template property. i.e. [global]\Nuts Unlimited\Folsom\Receiving\Nut Storage Silos\Almond Silo to Almond Silo.
- Add an **MES Object Selector** component from the Production component palette on to the screen below the Total label and enable the **Include Equipment Objects** property.

This will provide us with a dropdown we can use to select the storage bins (supplemental equipment).

For the simulator to run and change tag values, it must be enabled and reset.

- Add a **button** with the text **Mix** to start the mixing operation simulation.
- Add the following script to the button **action - actionPerformed** event to set the SimEnable and SimReset tags as shown.

Code Snippet

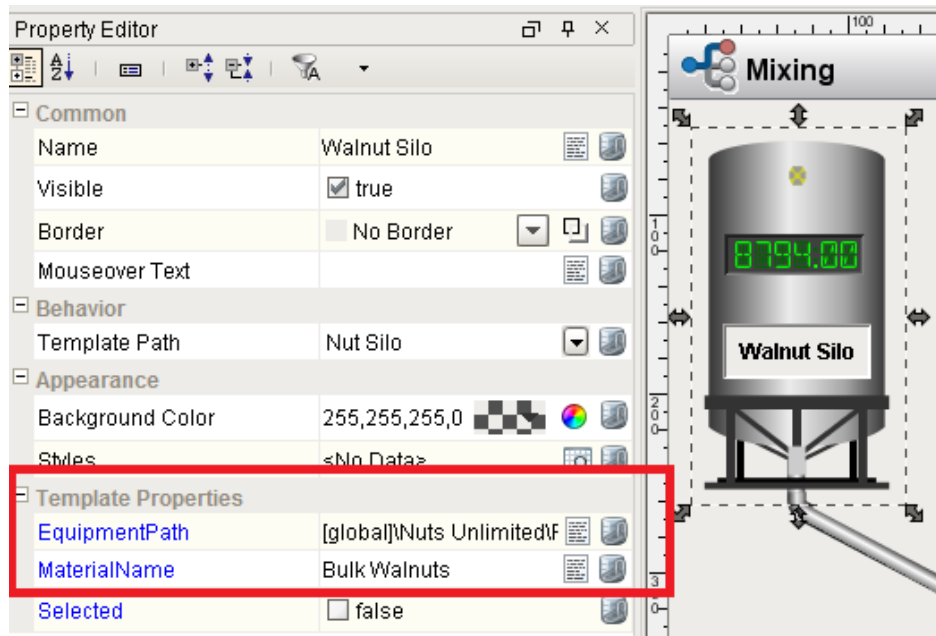
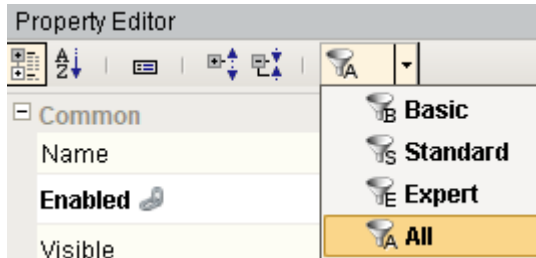
```
tagUDTPath = event.source.parent.tagUDTPath
system.tag.write(tagUDTPath + "/Error Message", "")
if event.source.parent.getComponent('MES Object Selector').
selectedName == '':
    system.gui.messageBox("Please select a storage bin to put the
mixed nuts in")
else:
    BinNo = event.source.parent.getComponent('MES Object Selector'
).selectedName
    system.tag.write(tagUDTPath + "/BinNo", BinNo)
    system.tag.write(tagUDTPath + "/SimEnable", 1)
    system.tag.write(tagUDTPath + "/SimReset", 1)
```

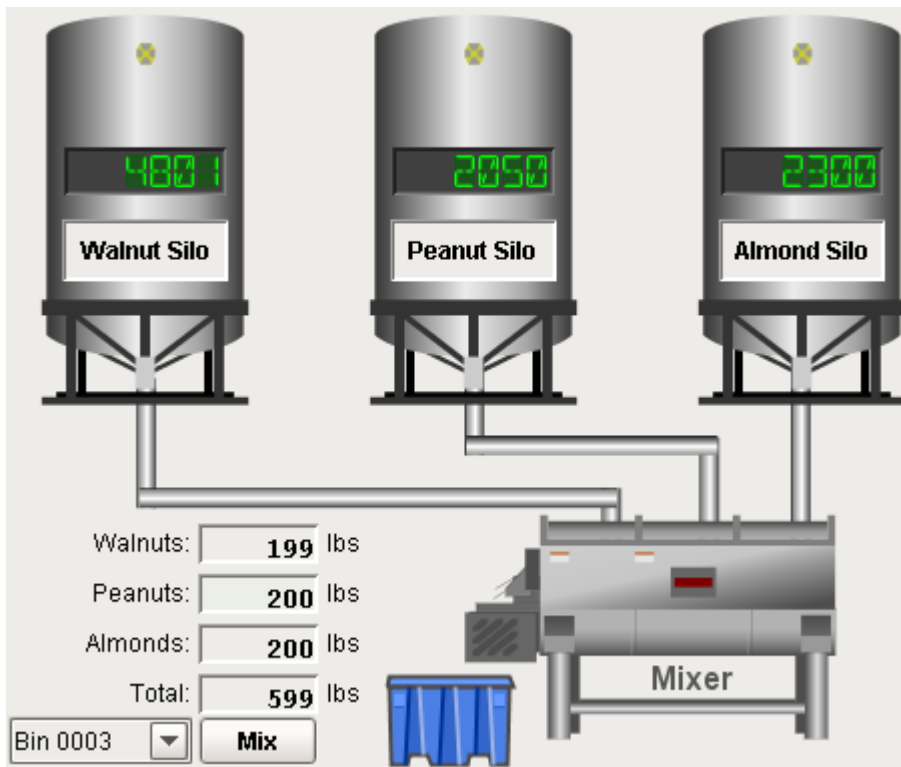
- In the Property Editor pane, bind the **Enabled** property of the button to the expression....

```
!tag({Root Container.tagUDTPath} + "/SimEnable")
```

- Set the expression **Overlay Opt-Out** to True.

- i** If you don't see the **Enabled** property, you'll need to change the filter to **ALL** on the Property Editor pane.





We'll now add a Table to the window to display the inventory in all of the mixing nut bins.

- Add a **Power Table** component to the Nut Mixing window and choose **Function** for the Data Property binding type.
- Select **Equipment WIP** binding function.
- Type in **Mixed Nut Bins** in the Equipment Path parameter. This is the equipment class that contains the bins used to store the mixed nuts.
- Right-click on the Table and use the **Table Customizer** to hide all columns except the **LotNumber**, **MaterialName**, **InQuantity**, **OutQuantity**, **NetQuantity**, **Units** and **LocationName** columns.

We can now test the Nut Mixing screen in the next section.


LotNumber	MaterialNa...	InQuantity	OutQuantity	NetQuantity	Units	Location ...
REJ-0002	Mixed Nuts	250	0	250		Reject Bin
0000000022	Mixed Nuts	599	0	599		Bin 0003
MN-0005	Mixed Nuts	3,750	0	3,750		Bin 0001

Binding Functions

For more info, goto [Trace Binding Functions Help](#).

Test the Mixing Operation

We are now ready to test that we have setup everything to allow us to mix some nuts and create a new mixed nut product.

- Go into Run mode by pressing **F5** or .
- Select an empty bin using the MES Object Selector and click the **Mix** button which will start our PLC simulator.
- Once a mixing operation has completed, mix again for each bin.

Keep an eye on the tag UDT member tags. You should see the **SimEnable** tag go true. After a few seconds the **Running** tag should go true. If the tag change event script worked, the **Mixing Active** tag should go true. If it doesn't go true, check the gateway console to see if an error was thrown by the Nut Mixing logger. We are also writing any errors to the **Error Message** tag and that is bound to an error message label on the screen.

If we have done everything correctly, the simulation will start to mix nuts and use the bin we selected to put the mixed nuts in. When we defined the bins, we set the **Lot Handling Mode** to **Single Lot** which will prevent 2 lots being added to the same bin, so make sure you select an empty bin. The Mixed Nut Inventory table will populate as mixing is completed and bins are filled.

LotNumber	MaterialNa...	InQuantity	OutQuantity	NetQuantity	Units	Location N...
0000000001	Mixed Nuts	599	0	599		Bin 0001
0000000002	Mixed Nuts	600	0	600		Bin 0002
0000000003	Mixed Nuts	601	0	601		Bin 0003

Problems?

If the simulation does not work, we'll have to debug to find out what is not working. If you don't see any nuts being mixed, check that your UDT tags are being set correctly to start the simulator.

Because we started the operation segment inside a tag change event in the UDT we created for Mixing Line 1, we won't see an error in our designer or the client. The Tag Change events are executed on the gateway and if you recall, we have the try/except clause around the seg.begin() and seg.end() that creates a logger called Nut Mixing. Any error messages will be logged to the gateway console with the logger called "Nut Mixing" and we are also writing any errors to the **Error Message** tag and that is bound to an error message label on the screen.

If you get the error....

E Nut Mixing	18Apr2017 11:00:19	Material property named (Mixed Nuts Out) is not defined.
---------------------	--------------------	--

...check the configuration of material definitions and material references in the Mix nuts process segment in the MES Object Editor.

If you get the error....

E Nut Mixing	14Mar2017 10:45:25	The specified material reference (Peanuts In) is missing a required lot selection.
---------------------	--------------------	--

...your Received Nut Silo(s) are empty. Go back to the Unloading screen and receive some nuts that can be used by the mixing operation

If you get the error...

E Nut Mixing	20Apr2017 09:34:11	The specified material (Mixed Nuts Out) is setup to save to location (Bin 0001), but the location is configured to hold a single lot and it already has an available lot.
---------------------	--------------------	---

...your storage bins are full. You'll have to consume the material in the packaging operation we'll create next or create a 'scrap material' operation to empty those bins. We show how to create a scrap material process segment in the [Mixing Test with the MES Material Selector](#) section.

Tracing the Mixing Operation

We now have more data that we can view in the Trace Graph.

- Open the **Trace Graph** window located in the *Analysis* folder.

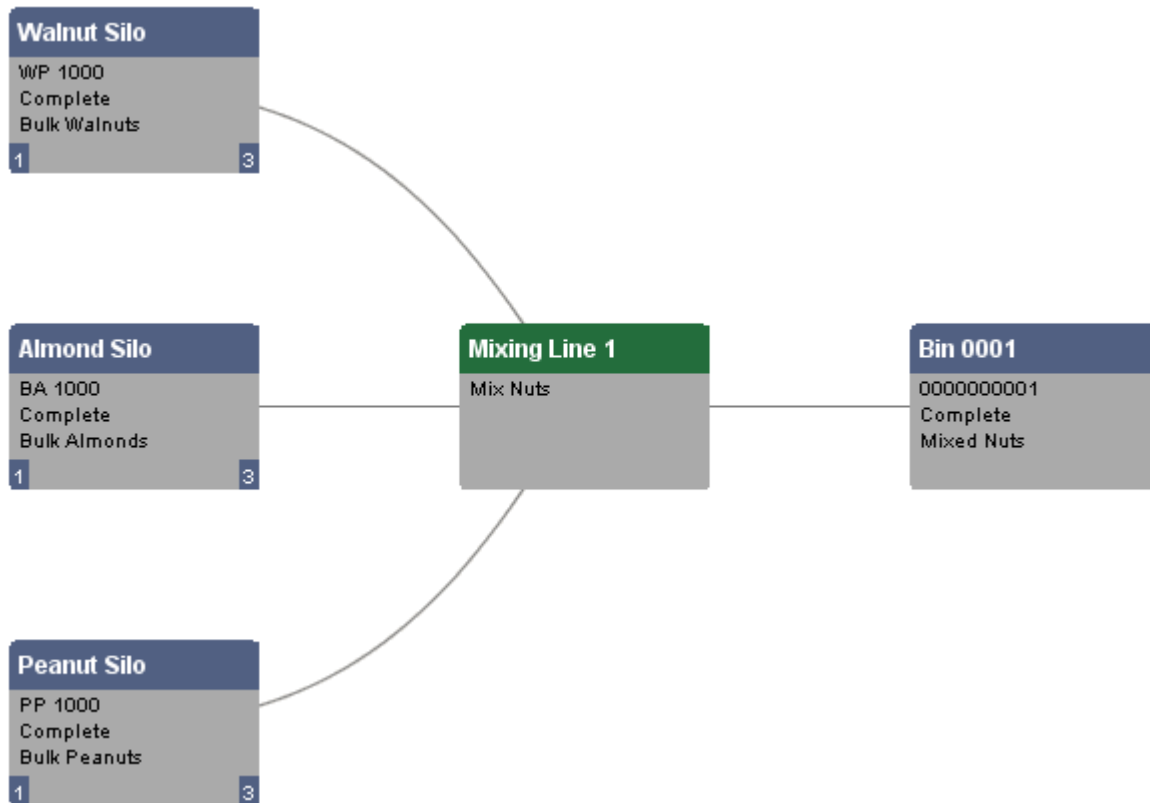
Select one of the lot numbers either used or created in the **Unload Nuts** operation to see the results. We see a simple trace graph showing the mixing process. Each box on the trace graph is called a node. The Mix Nuts node (green) represents the segment (production task) that was done. The Almond Silo node (blue) represents the material lot as a result of the Nut Unloading segment.

The colors of the different node types is configurable and can be modified by changing the Node configuration dataset of the Trace graph object. See [MES Trace Graph Help](#) for more details.

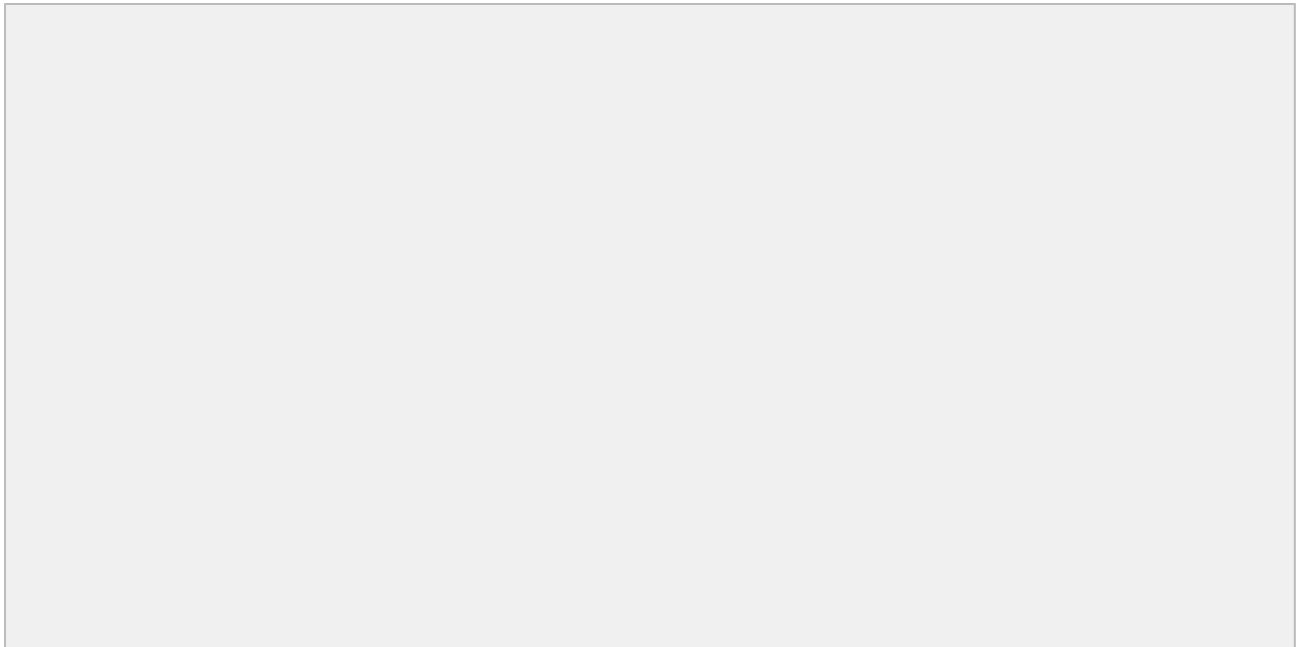
You can click on any segment node (green) to view the segment details. For example, click on the **Nut mixing** node and select **Show Details**. A popup will open with the details.

You can click on the material lot node (blue) to view trace information for that Lot No. You can view any raw material and any finished good and anything along the way. You can also click on the **Table** tab to view the data in tabular form.

Each node can be configured to add custom user menu items. You can use this to extend the trace graph functionality to bring back any data associated with that lot or process that was not captured as part of Track & Trace. An example would be to bring back OEE or SPC data or tag process history data. For more information on creating custom user menu items, see the `UserMenuItemClicked` eventHandler in the [MES Trace Graph Help](#) for more details.



At this point, we will add a template to the mixing screen to show a trace graph. We already saw the trace in the Trace Graph window, but this exercise will show how we can embed the trace component onto the same screen using templates and we'll use this later on to create windows within windows.



- Drag the **WIW /Tracer** template onto the Mixing Screen.

The screenshot displays the Sepasoft software interface with several panels:

- Project Browser:** Shows a tree structure with 'WIW' expanded, containing 'Tracer' and 'WIW Template'.
- Tag Browser:** Shows a tree structure with 'Tags' expanded, containing 'Data Types', 'Nuts Unlimited', 'Mixing', 'Mixing Line 1', 'Line 1 Mixing Control', 'AlmondPounds', 'BinNo', and 'Error Message'.
- Property Editor:** Shows properties for 'Nut Mixing' under 'Behavior' and 'Appearance'.
- Mixing Screen:** A diagram showing three silos (Walnut Silo, Peanut Silo, Almond Silo) connected to a 'Mixer'. Below the silos, a table shows the following data:

Material	Weight (lbs)
Walnuts	199
Peanuts	200
Almonds	200
Total	599

 A 'Bin 003' dropdown and a 'Mix' button are also visible.

A red arrow points from the 'Tracer' template in the Project Browser to the 'Mixing Line 1' node in the diagram.

- Add the following script to the Power Table `propertyChange` event:

```
if event.propertyName == 'selectedRow':
    row = event.newValue
    data = event.source.data

    if row != -1:
        lotNumber = data.getValueAt(row, "LotNumber")
    else:
        lotNumber = ''

    inputParams = '{"lotNumber":"' + lotNumber + '"'
    event.source.parent.getComponent('Tracer').inputParams
```

- **Save** your changes

You can now click on any of the lots created in the power table and the lot number will be passed to the template that contains the trace graph and operation info panel. Hover over the segment node to see information about the operation and lots.

Using the MES Material Selector

⚠ Skip past this part of the tutorial. We have an issue with the MES Material Selector component and the <any equipment> class that the developers are currently working on.

Before we move on, we'll try out the [MES Segment Selector](#) and [MES Material Selector](#) components for executing a mixing operation instead of the tag UDT we created earlier. This will simply demonstrate another way that we can run operation segments.

We have already provided a template for you called **Segment Control**. The Segment Control template contains a [MES Segment Selector](#), [MES Material Selector](#) and four buttons. You will need to add the scripting and binding as detailed below to finish it off before we can use it on the Nut Mixing screen.

- Open the **Segment Control** template in the designer.
- Bind the Equipment Path property of the **MES Segment Selector** to the template property *eqPath*.
- Change the **Mode** property of the **MES Segment Selector** to **Definition**.
- Right click on the **Start** button and select the **Scripting** menu item. Add the following script to the Action/ActionPerformed event.

```
event.source.parent.getComponent('MES Segment Selector').  
beginSegment()
```

- Add the following script to the **Update** button ActionPerformed event:

```
event.source.parent.getComponent('MES Segment Selector').  
updateSegment()
```

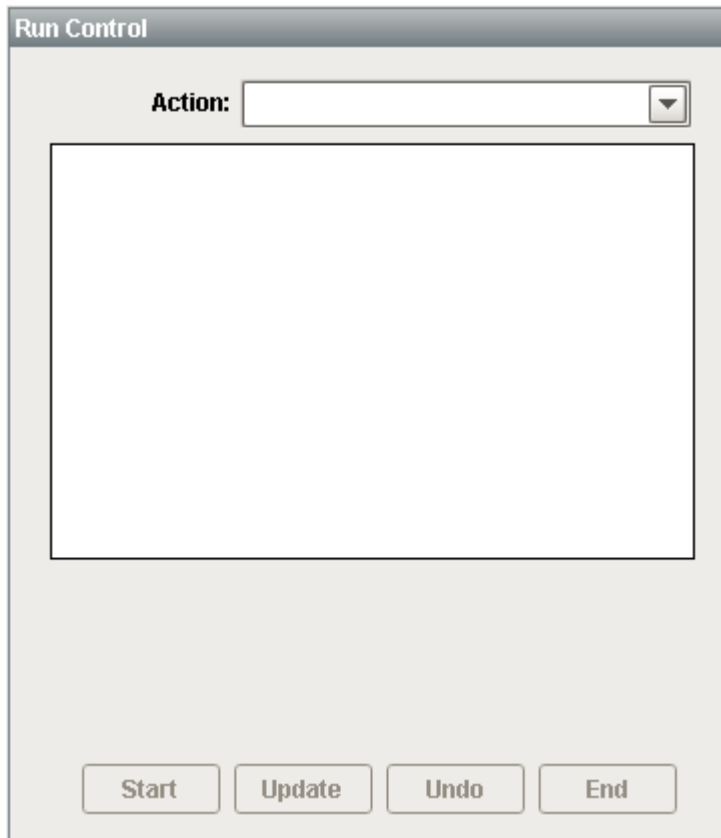
- Add the following script to the **Undo** button ActionPerformed event:

```
event.source.parent.getComponent('MES Segment Selector').  
undoChanges()
```

- Add the following script to the **End** button ActionPerformed event:

```
event.source.parent.getComponent('MES Segment Selector').  
endSegment()
```

- Set the **Enabled** property binding of the **Start** button to **Property** binding type and bind it to the **Can Begin Segment** property of the MES Segment Selector.
- Set the **Enabled** property binding of the **Update** button to **Property** binding type and bind it to the **Can Update Segment** property of the MES Segment Selector.
- Select the **Undo** button and set the **Enabled** property binding to **Property** binding type and bind it to the **Can Undo Segment** property of the MES Segment Selector.
- Select the **End** button and set the **Enabled** property binding to **Property** binding type and bind it to the **Can End Segment** property of the MES Segment Selector.
- **Save** your changes to the **Segment Control** template.

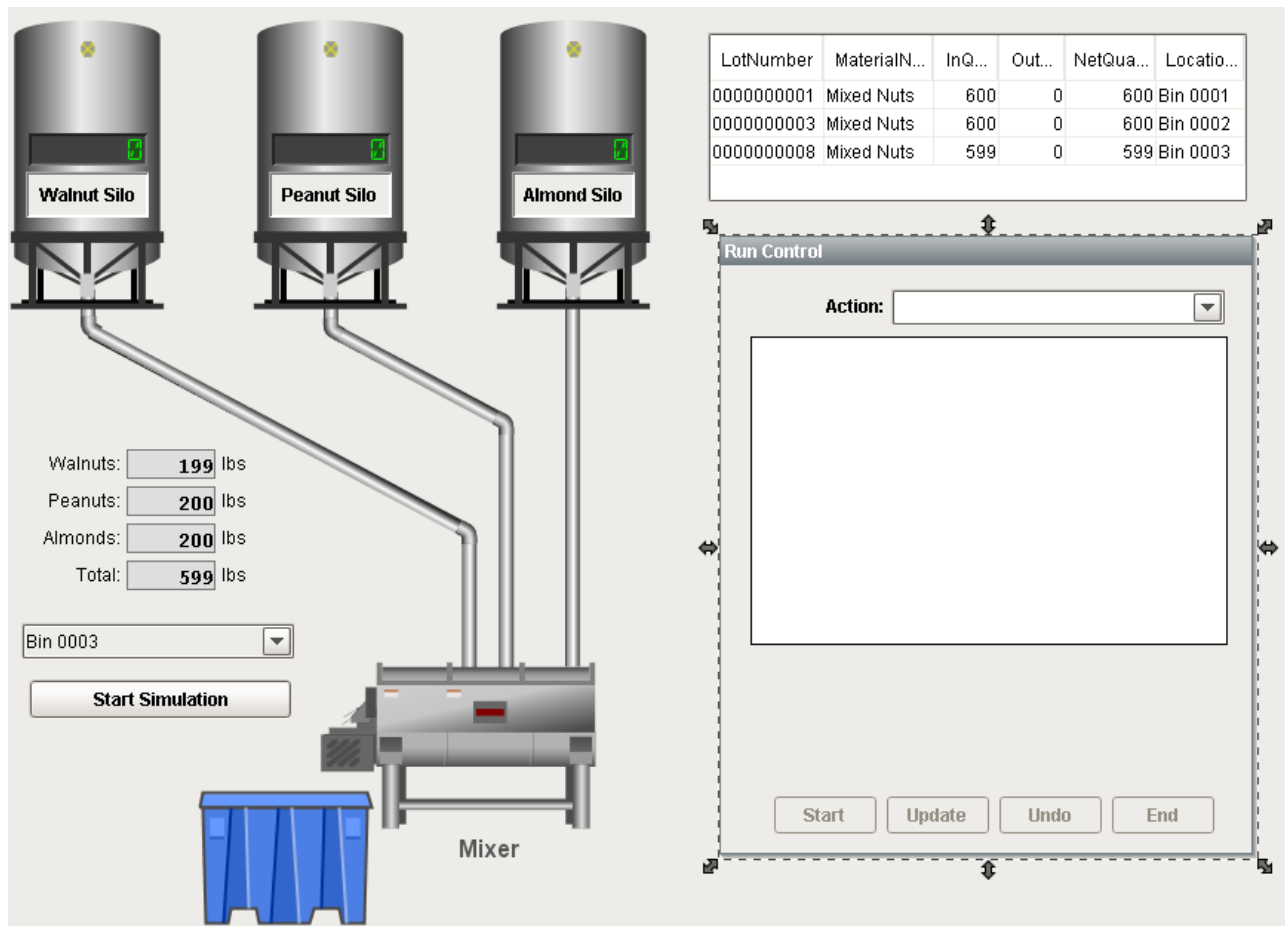


- i** The [MES Segment Selector](#) component has a number of functions that can be called on to start and end operation definitions and operation segments. The functions do not show up in the property selector, only properties, so you'll have to know what the functions are. The component help lists these. You can also add a button and call `print dir(event.source.parent.getComponent('MES Segment Selector'))` to get a listing of all constructors, properties and functions available to this and any type of component.

For more help see [MES Segment Selector](#)

Now that the template is complete, we can use it on our Nut Mixing window.

- Drag the **Segment Control** template onto the Nut Mixing window.
- Right-click on **Mixing Line 1** in the production model and select **Copy Equipment Path**.
- On the Nut Mixing screen, select the *eqPath* property of the **Segment Control** template and copy the equipment path to the **Equipment Path** property.



Mixing Test with the MES Material Selector

⚠ Skip past this part of the tutorial. We have an issue with the MES Material Selector component and the <any equipment> class that the developers are currently working on.

Before we can run a mixing operation, we will have to clear one of the bins otherwise we won't have any place to put our mixed nuts. As we haven't yet created a packing operation to consume material out of the bins, we will create a process segment that will allow us to scrap any material from any location.

- If you don't have equipment class named **<any equipment>** and material class named **<any material>**, then create those classes. Note, they are created automatically the first time you use the OEE Material Manager.
- Create a process segment called **Scrap Material**.

- Add an equipment reference called **Scrap Equipment** and set the Equipment Reference to **Equipment Class, <any equipment>**.
- Set up the following properties for the Material Reference:

Property	Value	Description
Name	Any Material	Name we can use to reference this material in scripting
Use	In	This material is consumed by this processed
Material Reference	Material Class - <any material>	Any material that belongs to this class can be consumed
Lot Equipment Reference	Equipment Class - Mixed Nut Bins	Any material at equipment that belongs to this class can be consumed. Note If we made the Lot Equipment Reference <any equipment>, then this operation could be used to scrap any material anywhere. In the current release the MES Material Selector doesn't handle <any equipment>, but this would work with scripting.
Lot Number Source	Auto	Will use the lot number of the material
Quantity Source	Available Lot Quantity	All available material will be consumed
Rate Period	None	

Now let's use this **Scrap Material** operation segment to clear one of the bins. In this case we'll use the **Sample Operation Control** screen in the Trace Sample screens folder. This screen is very useful for testing and debugging a Track & Trace implementation. It allows you to select and test any process segment you have created. You should use this screen quite a bit during a project implementation.

- Select the **Nut Unloading** equipment in the MES Object Selector component.
- Select the **Scrap Material** operation definition in the Operation Selector component and click the **Begin Operation** button.
- Select the **Scrap Material** operation segment in the Segment Selector component.
- Select the Bin you want to clear in the Material Selector component and click the **Execute Segment** button.

You should now have an empty bin that we can use to test out the Material Selector template we added to our Nut Mixing screen.

- In the Segment Control template that we dragged onto the Nut mixing screen, select the Action **Mix Nuts**. Set a quantity for Almonds, Peanuts and Walnuts, click the **Start** button and then the **End** button.

If all went well, you should have filled your empty storage bin with a new batch of mixed nuts.

Create a Scrap Material Operation

Let's create one more operation called **Scrap Material**. We'll use this anytime we want to clear inventory out from a location.

- Create a process segment called **Scrap Material**.
- Add an equipment reference called **Scrap Equipment** and set the Equipment Reference to **Equipment Class, <any equipment>**. If the **<any equipment>** class doesn't exist, create it.
- Set up the following properties for the Material Reference:

Property	Value	Description
Name	Material In	Name we can use to reference this material in scripting
Use	In	This material is consumed by this processed
Material Reference	Material Class - <any material>	Any material that belongs to this class can be consumed. If the <any material> class doesn't exist, create it.
Lot Equipment Reference	Equipment Class - <any equipment>	Any material anywhere can be consumed. If the <any equipment> class doesn't exist, create it.
Lot Number Source	Manual	Will use the lot number of the material
Quantity Source	Available Lot Quantity	All available material will be consumed
Rate Period	None	

- **Save** the object and select **Yes** when asked to create an operations definition for this segment.

Now let's use this **Scrap Material** operation segment to clear one of the bins.

- Add a button to the Mixing Screen and change the text to **Scrap Nuts**.
- Add the following script to the actionPerformed event of the button:

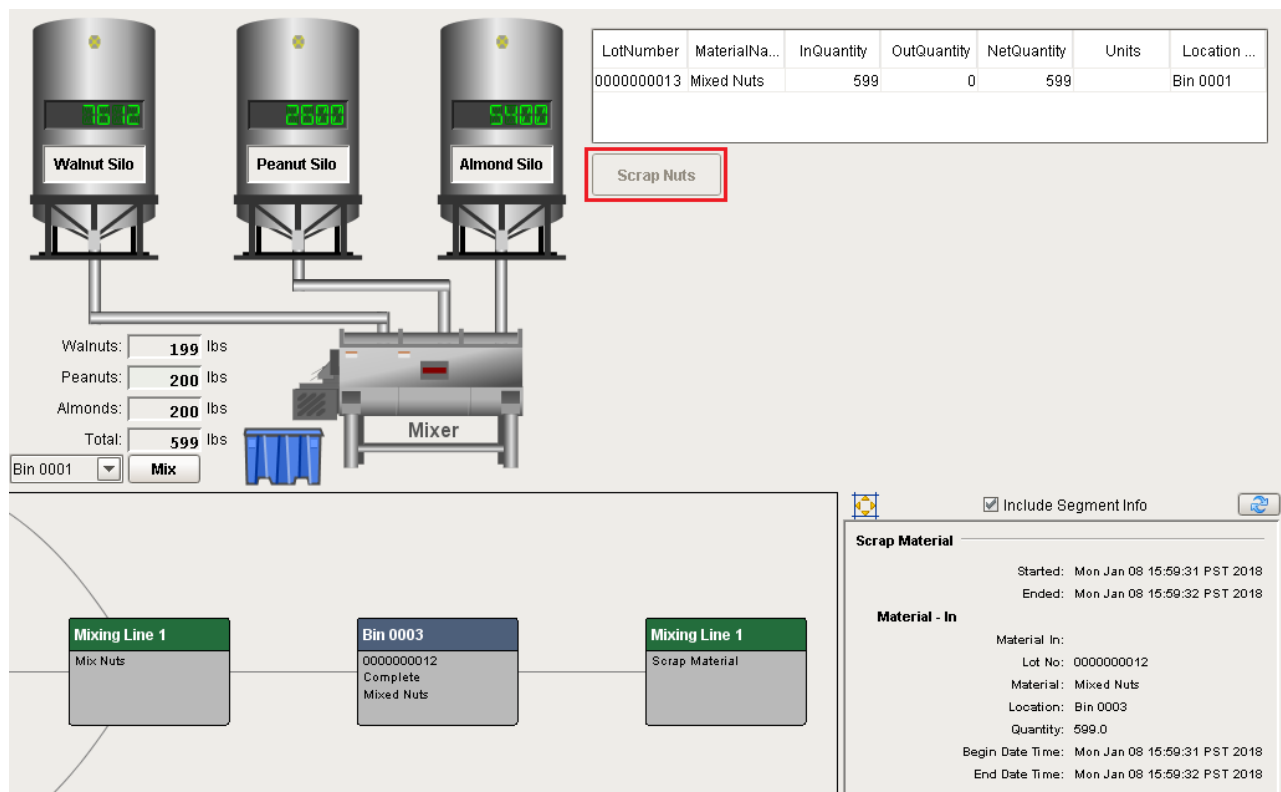
```
comp = event.source.parent.getComponent('Power Table')
data = comp.data
selectedRow = comp.selectedRow
lotNumber = data.getValueAt(selectedRow, "LotNumber")
lotSequenceNumber = data.getValueAt(selectedRow, "LotSequence") #Setting this to -1 will always return the most recently created lot instance

#Let's find out how much of this material we have
matObj = system.mes.loadMaterialLot(lotNumber, lotSequenceNumber, True)
netQty = matObj.getLotInventory().getNetQuantity()

#Now we'll create an instance of the Scrap Material Operation
eqPath = '[global]\Nuts Unlimited\Folsom\Mixing\Mixing Line 1'
operation = 'Scrap Material'
autoAssignOptions = False
#If there may be more than one active operation on this equipment then we need to call
oper = system.mes.createOperation(operation, eqPath, autoAssignOptions)
oper.begin()
seg = oper.createSegment(operation)

#Let's put together the information about the material lot we will consume and thereby scrap
materialPropertyName = 'Material In'
qty = netQty
seg.setMaterial(materialPropertyName, lotNumber, lotSequenceNumber, qty)
seg.execute()
#Let's force the table to immediately update itself
system.db.refresh(comp, 'data')
```

- Press **F5** to go into preview mode, select a lot of mixed nuts in the power table and press the **Scrap Nuts** button.



3.2.5 Create Inventory Operations

Before we move onto OEE, we'll create some more generic operations that will allow us to perform some common inventory tasks. These will be nice-to-have operations for any project, but we'll use them here to really become familiar with the scripting side of executing operations.

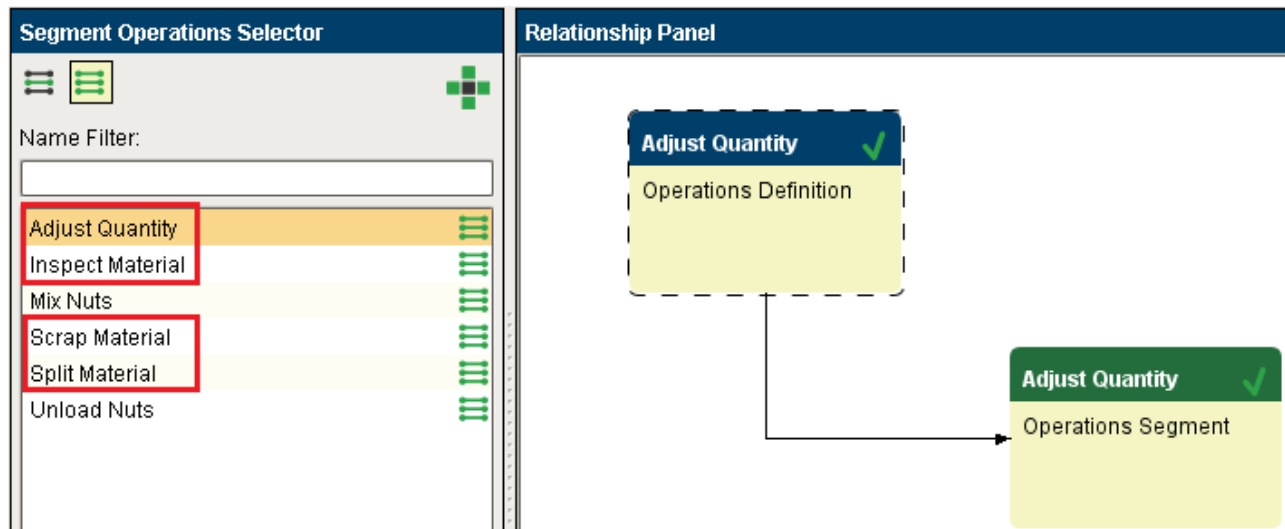
📌 What You'll Learn In This Section

- Importing MES Objects
- Using the Power Table *onRowsDropped* extension function
- Using Client Tags in your project to share data between screens
- Using generic operations to create common functions
- How Track & Trace can be used to create a plant floor inventory management system

- Import the following operations using the MES Object Exporter screen MES Object Export Operations Definition - Inventory Functions.xml.

Refer to the kb [Exporting MES Objects Between Gateways](#) for more information on how to import MES Objects.

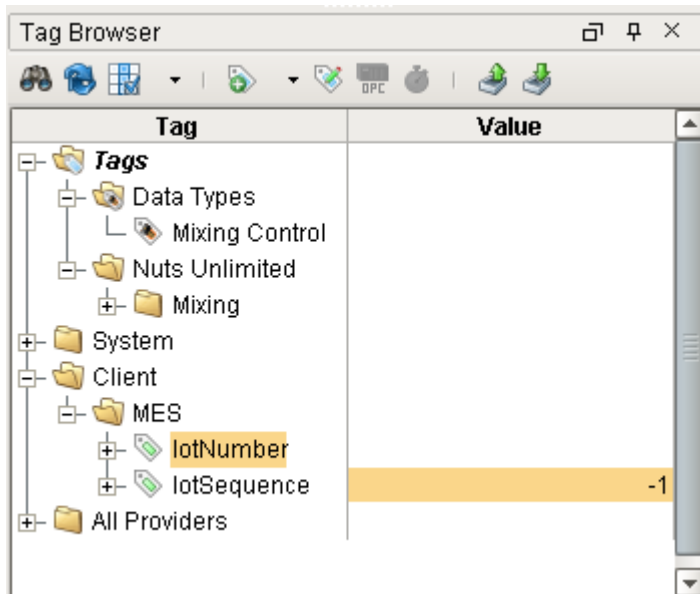
- Verify the following new operations exist in the MES Object Editor component



We're going to add three templates to this screen that will allow us to view the inventory in our bins, perform some inventory adjustments on the lots at those bins, and view those operations. Before we can do that, we will create a couple of client tags that will allow us to pass the material lot information between the templates. We could do this with a custom property on the root container or a script function, but client tags are a great way of passing parameter values between windows. Consider using them to set a date range or select a production line or area. With client tags, you no longer need to add a date range or MES Object Selector components to every window that a user that might to look at.

- Create a Client Tag folder called **MES** with the following clients tags:

Tag	DataType
lotNumber	String
lotSequence	Integer



- Create a new Windows Folder called **Inventory** and a new main window called **Inventory Functions**.
- Drag three Inventory Table templates from the **Trace** folder onto the screen.
- For each template, set the template parameters to the following:

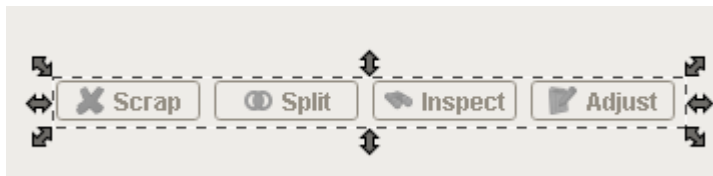
Parameter	Inventory Table	Inventory Table 1	Inventory Table 2
eqPath	Bin 0001	Bin 0002	Bin 0003
lotNumber	Bind to [Client]MES/lotNumber Check box for "Bidirectional"	Bind to [Client]MES/lotNumber	Bind to [Client]MES
lotSequence	Bind to [Client]MES /lotSequence Check box for "Bidirectional"	Bind to [Client]MES /lotSequence	Bind to [Client]MES /lotSequence
opPath	[global]\Nuts Unlimited\Folsom\Mixing\Mixing Line 1	[global]\Nuts Unlimited\Folsom\Mixing\Mixing Line 1	[global]\Nuts Unlimited\Folsom\W

Bin 0001				
Lot #	Material	Qty	Units	
0000000019	Mixed Nuts	600		

Template Properties	
eqPath	Bin 0001
lotNumber	0000000017
lotSequence	3
lotStatus	
opPath	[global]\Nuts Unlimited\Folsom\Mixing\Mixing Line 1

- Drag an Inventory Functions template from the **Trace** folder onto the screen.
- Set the Inventory Functions template parameters to the following:

Template Property	Value	Description
equipPath	[global]\Nuts Unlimited\Folsom\Mixing\Mixing Line 1	Equipment Path that the operation will run against. This must be a fixed piece of equipment.
lotNumber	Bind to [Client]MES/lotNumber Check box for "Bidirectional"	Material lot number that will be passed to the operation.
lotSequence	Bind to [Client]MES /lotSequence Check box for "Bidirectional"	Lot sequence that will be passed to the operation.

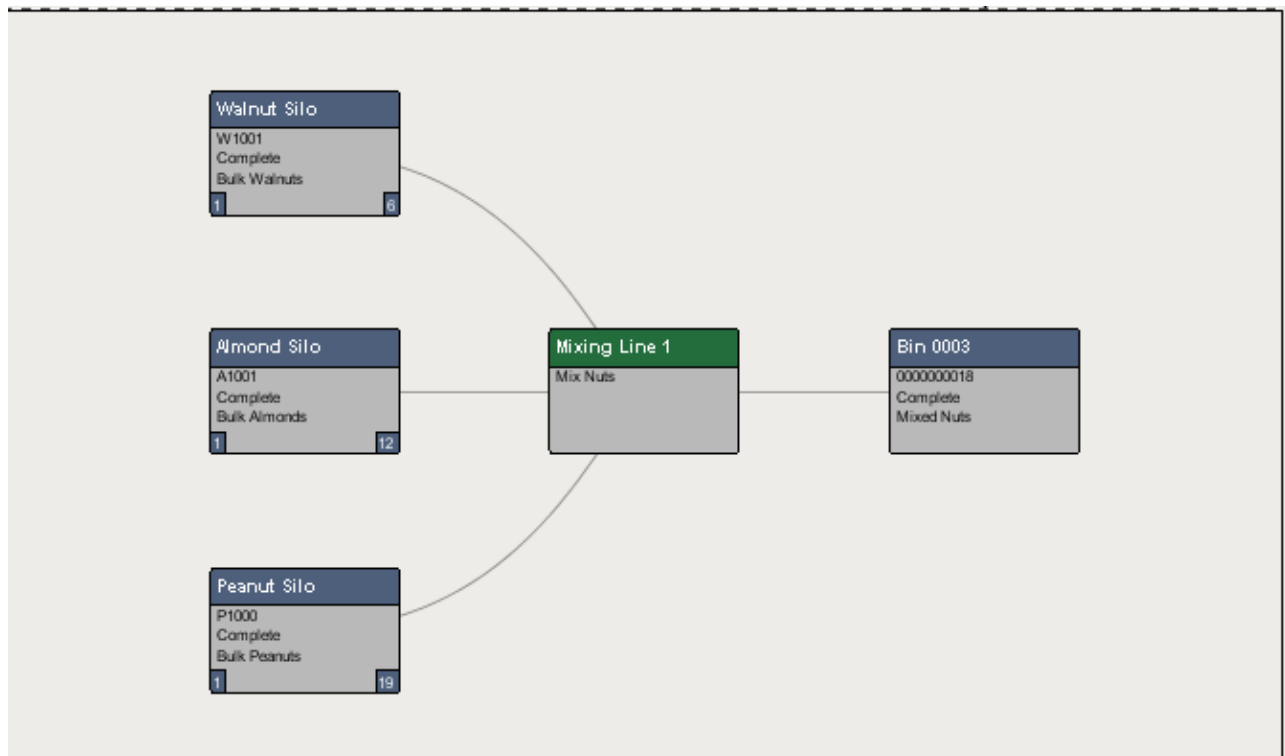


Template Properties	
equipPath	[global]\Nuts Unlimited\Folsom\W
lotNumber	
lotSequence	-1

- Drag an Tracer template from the **WIW** folder onto the screen.
- Set the **inputParameters** property of the Tracer template to the following expression:
`"{'lotNumber'=''} + {[Client]MES/lotNumber} + ''"`

If all went well, you should be able to click on a lot in any of the Inventory tables to select it. The client tag will then contain the *lotNumber* and *lotSequence* for that lot. This value is passed to the Inventory functions template allowing you to **Scrap, Split, Inspect or Adjust** the quantity of that lot. We also bound the *inputParameter* property of the Tracer template to the client *lotNumber* tag allowing us to see the operations performed that created and consumed from that lot.

In this example, we passed **Bin 0001** to the *eqPath* of the Inventory Table. We passed the name of the equipment because this is supplemental equipment. If we want to view the inventory of fixed equipment such as the Nut Silos, we would pass the Equipment Path i.e. *[global]Wuts Unlimited\Folsom\Receiving\Nut Storage Silos\Almond Silo*.



Template Properties	
BGColor	238,236,232
inputParameters	{'lotNumber'='00000000'}

We can also move inventory between locations by clicking on a lot in the table and dragging it to another table. This uses the *onRowsDropped* extension function of the Power Table component in which we are calling the **Move Parts** operation.

[Click here to see the onRowsDropped extension function script...](#)



```

def onrowsDropped(self, sourceTable, rows, rowData,
dropIndexLocation):
    # convert the Ignition dataset to a Python dataset
    pythonDataset = system.dataset.toPyDataSet(rowData)
    for row in pythonDataset:
        #set the Material, location, lot number and quantity
        matDef = row['MaterialName']
        lotID = row['LotNumber']
        lotSequence = row['LotSequence']
        qty = row['NetQuantity']
        fromEquipUUID = row['Location UUID']
        eqPathFrom = system.mes.loadMESObject(fromEquipUUID).
getEquipmentPath()
        #Set the Equipment Path to move To
        eqPathTo = self.parent.eqPath
        if eqPathFrom != eqPathTo:
            #get the segment name
            segName = 'Move Parts'
            opPath = self.parent.opPath
            #Create a segment for the current operation response
            seg = system.mes.createSegment(segName, opPath, False)
            #Set material
            seg.setMaterial('Material In', lotID, lotSequence, qty)
            seg.setMaterial('Material Out', matDef, eqPathTo, lotID,
qty)

            #Execute the segment (Move the material)
            seg.execute()
            self.refreshData() #Update the data binding to reflect
what has changed

```

3.2.6 Copying and Deriving MES Objects

Track & Trace provides us with a number of ways to create MES Objects for resources, class and operations, both through the [MES Object Editor](#) as well as scripting. But we can also create objects in different ways too, by creating New , by copying, and by deriving. In this section, we will explore the difference between copying and deriving and then create a derived object that we'll use to show how the number and type of input and output materials consumed and created can be determined on the fly through scripting.

✔ What You'll Learn In This Section

- Copying versus Deriving Objects.

Create an Any Operation Process Segment

- Create a process segment called **Any Operation** in the MES Object Editor.
- Add an equipment reference called **New Equipment** and set the Equipment Reference to **Equipment Class, <any equipment>**.
- Add a Material Complex Property and set up the following properties:

Property	Value	Description
Name	Material In	Name we can use to reference this material in scripting
Optional	True	This is an optional Material
Use	In	This material is consumed by this processed
Material Reference	Material Class - <any material>	Any material that belongs to this class can be consumed.
Lot Equipment Reference	Equipment Class - <any equipment>	Any material anywhere can be consumed.
Lot Number Source	Manual	Will we pass the lot number in scripting
Quantity Source	Manual	Will we pass the lot quantity in scripting
Rate Period	None	

- Add another Material Complex Property and set up the following properties:

Property	Value	Description
Name	Material Out	Name we can use to reference this material in scripting
Optional	True	This is an optional Material
Use	Out	This material is consumed by this processed
Auto Generate Lot	True	Out materials always need to have this set to True
Material Reference	Material Class - <any material>	Any material that belongs to this class can be consumed.
Lot Equipment Reference	Equipment Class - <any equipment>	Any material anywhere can be consumed.
Lot Number Source	Manual	Will we pass the lot number in scripting
Quantity Source	Manual	Will we pass the lot quantity in scripting
Rate Period	None	

- **Save** the object and select **No** when asked to create an operations definition for this segment.

Process Segment, Any Operation

Save

Core Properties

Name	Any Operation
Description	
End Operation When Complete	<input checked="" type="checkbox"/>
Segment Recipe Name	<no recipe selected>

Custom Properties

none defined	
--------------	--

Material

Material In	In, Material Class, <any material>
Material Out	Out, Material Class, <any material>

Equipment

New Equipment	Equipment Class, <any equipment>
---------------	----------------------------------

Personnel

none defined	
--------------	--

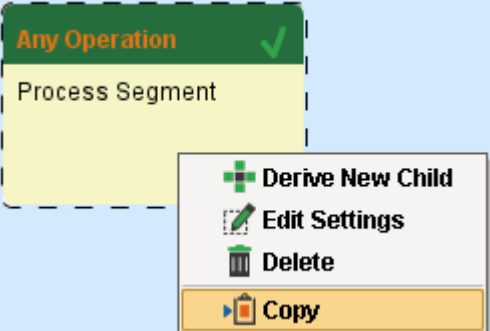

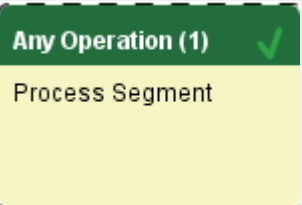
Supplemental Equipment

none defined	
--------------	--

Production Settings

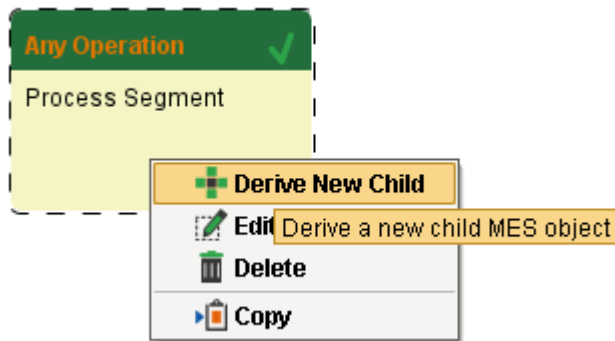
none defined	
--------------	--

Create the Any Operation Process Segment

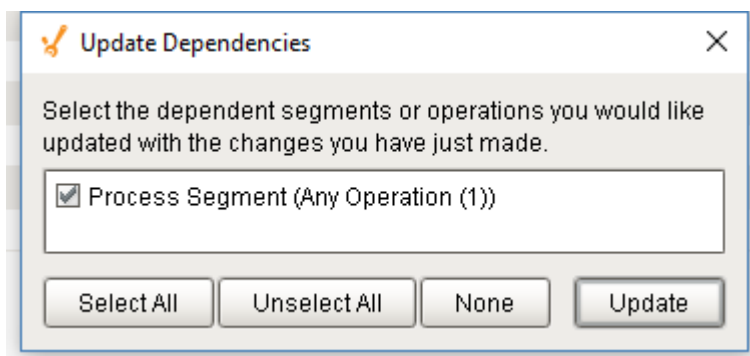
<ul style="list-style-type: none"> Right click on the Any Operation process segment and select Copy. 	
<ul style="list-style-type: none"> Right click outside of the Any Operation process segment and select Paste. 	
<p>This creates an exact copy of the Any Operation process segment that contains the exact settings that we defined for the Any Operation process segment. We can now make changes to the Any Operation process segment and those changes will not affect the Any Operation (1) process segment.</p> <p>We are not going to use this new segment though so we can go ahead and delete it. It was just to show a quick way to new segments that contain similar settings.</p>	
<ul style="list-style-type: none"> Right click on the Any Operation (1) process segment and select Delete. 	

Derive a Child from the Any Operation Process Segment

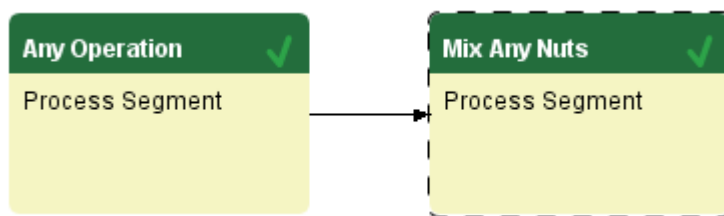
<ul style="list-style-type: none"> Right click on the Any Operation process segment and select Derive New Child.



We have now created a new process segment called **Any Operation (1)** that is derived from the **Any Operation** process segment. All the same settings exist in both segments, but this time any changes that made to the parent object will be updated in the child object. The arrow between segments shows that this child object is derived from the parent object. If you change the description field in the parent object and select **Update** when prompted, the dependencies between objects will be updated. This object derivation and dependencies are the same for when an operations segment is automatically created for a process segment.



- Right click on the **Any Operation (1)** process segment and change its name to **Mix Any Nuts**.
- Click **Save** and select **Yes** when prompted to create the operation segment for this process segment.



We have already created an operation called **Mixed Nuts**, so why are we creating this new one? Well the **Mixed Nuts** operation shows the full extent of how production control can be implemented in a manufacturing operation, by placing control over who can perform this operation, what type of material lots can be consumed, from where and what type of new material lot is created is created and where. When bulk lots are consumed from the silos, the lot number is automatically assigned because we set the lot handling mode of the nut silos to FIFO, and the output quantity is automatically calculated for us by the infeed quantities. This type of built-in production control is important and useful when we are using the standard Track & Trace components such as the [MES Material Selector](#) component to create a user entry screen. But often times, the operation execution will be implemented using standard ignition components and scripting, operation execution may be fired by plc tag change events and we may want to simply record that an operation took place rather than validate if it is allowed to take place.



We will now use the new **Mix Any Nuts** operation to show how we determine exactly what material lots were consumed in an operation and what lots were created and where through scripting.

3.2.7 Operation Execution Through Scripting

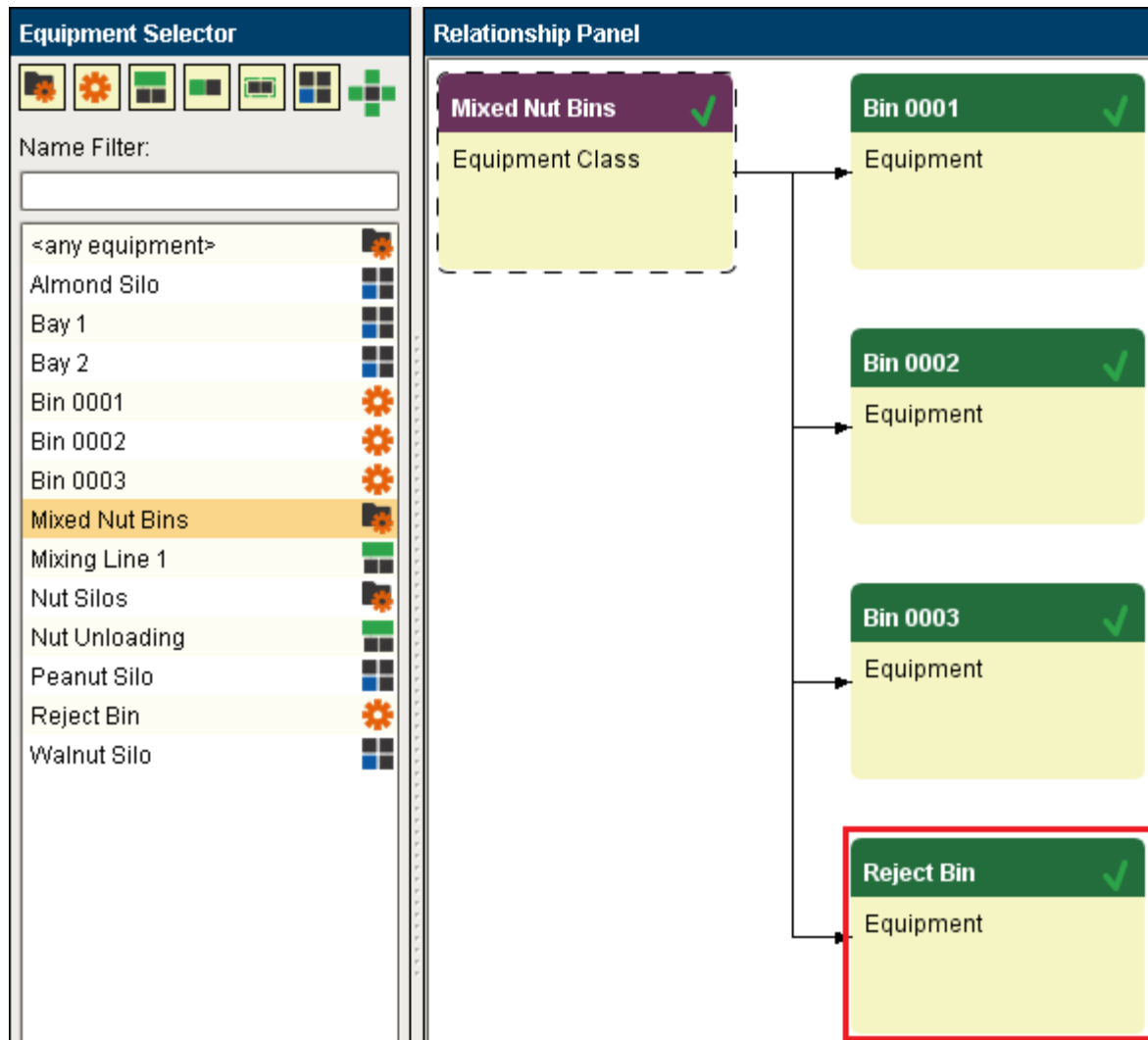
In this section, we will import a pre-built screen that shows how we can use scripting with the **Mix Any Nuts** operation created in the last section to control what lots are consumed and created.

✔ What You'll Learn In This Section

- Consuming input materials and creating output materials on the fly.
- Using component extension functions (Active Operation button script).
- Starting simultaneous operations on a line.
- Using a process segment with a single optional input and output material property to consume and create many different input and output lots.
- Avoiding an operation from automatically assigning unwanted resources to a response segment.
- Using `seg.update()` and reloading the segment object.
- Clearing a selection on the MES Object Selector and then forcing it to update to reflect newly created objects.


- In the MES Object Editor screen, select  **Equipment**.
- Add a new piece of equipment called **Reject Bin** by clicking .
- Set the Lot Handling mode to **Random Lot**.
- Add it to the **Mixed Nut Bins** class


As we mix nuts, we can choose if the material lots being created go to one of the Mixed Nut Bins (Bin 0001 - 0003) or to our newly created *Reject Bin*. In our Trace Graph component, we have *Reject Bin* configured to be shown with a red header background to make the rejected nuts stand out a bit more. We could as easily create a new material definition called *Reject Nuts* or set the lot status to *Rejected*. There are many ways to do this and how you choose to do it will be based on your process.






- Import the [Generic Mixing Operation.proj](#).

This screen is complete and you should be able to use it to mix some nuts. Make sure you have some bulk nuts in your silos. If your silos are empty, go back to the **Nut Unloading** screen and receive some more nuts into each of the silos.

 There are a couple of containers on the screen that you can ignore for now.

Enter Filter Screen Size	<input type="text" value="10"/>	mm	Enter Mix Speed	<input type="text" value="20"/>	rpm	
--------------------------	---------------------------------	----	-----------------	---------------------------------	-----	---

Select Mixing Operator	<input type="text" value="Jacobs, Rodney"/>			
------------------------	---	---	---	---

We will be using these in the next sections when we add custom properties to our mixing operation and want to track operators.

Select Mixing Line

Mixing Line 1

Start Mix Operation

Get Active Operations

Line

Type

Name

Mixing Line 1	OperationsResponse	Mix Any Nuts
Mixing Line 1	Response Segment	Mix Any Nuts

Select Material(s) to Add

LotNumber	MaterialName	NetQuantity	Units	Location Name
A1005	Bulk Almonds	500 lbs		Almond Silo
P1005	Bulk Peanuts	0 lbs		Peanut Silo
P1006	Bulk Peanuts	250 lbs		Peanut Silo

Add Material(s)

Select Mixing Operator Jacobs, Rodney

Select Material Out Mixed Nuts

Bin 0001

Enter Quantity 3,750

Enter Lot Number MN-0005

Create Lot(s) Out

End Operation

LotNumber	MaterialName	NetQuantity	Units	Location Name
MN-0005	Mixed Nuts	3,750		Bin 0001
REJ-0002	Mixed Nuts	250		Reject Bin

Enter Filter Screen Size 0 mm

Enter Mix Speed 0 rpm

Scrap Nuts

Peanut Silo

P1005

Active

Bulk Peanuts

Walnut Silo

W1003

Complete

Bulk Walnuts

Almond Silo

A1004

Complete

Bulk Almonds

Mixing Line 1

Mix Any Nuts

Bin 0001

MN-0005

Active

Mixed Nuts

Reject Bin

REJ-0002

Complete

Mixed Nuts

MN-0005

Include Seg Info

Lot No: P1005

Material: Bulk Peanuts

Location: Peanut Silo

Quantity: 1000.0

Begin Date Time: Fri Jan 12 08:23:53 PST 2018

End Date Time: -

Material - Out

Material Out:

Lot No: REJ-0002

Material: Mixed Nuts

Location: Reject Bin

Quantity: 250.0

Begin Date Time: Fri Jan 12 08:22:38 PST 2018

End Date Time: Fri Jan 12 08:25:08 PST 2018

Material Out-1: Active

Lot No: MN-0005

Material: Mixed Nuts

Location: Bin 0001

Quantity: 3750.0

Begin Date Time: Fri Jan 12 08:25:08 PST 2018

End Date Time: -



Selected Node Type - ResponseSegment

9321267d-6145-44ef-a21c-2418e9c119ce

Step

Description

© 2017 Sepasoft, Inc.

Step	Description
<ul style="list-style-type: none"> Select Mixing Line 1 	<p>If <i>Mixing Line 1</i> doesn't show up, press the  button. This button has a script that will clear the current selection and then force itself to refresh its data from the gateway. When the MES Object Selector is first instantiated on the screen, it pulls data from the gateway. If you then add equipment, this selector will not reflect that change until it has <code>.update()</code> called on it.</p> <p>Click here to view the script...</p> <pre data-bbox="727 762 1382 926">event.source.parent.getComponent('MES Object Selector Line'). clearSelection() event.source.parent.getComponent('MES Object Selector Line').update()</pre>
<ul style="list-style-type: none"> Click  Get Active Operations button 	<p>The table to the right will show any active operations currently underway. We have scripting in our buttons that will allow simultaneous operations to occur on the mixing line, but you can right-click on the table to end or abort any active operations if you'd like.</p> <p>Click here to view the script...</p> <p>The button simply calls an extension function we have created on the table component itself. We use an extension function here, because we want to call this function not only from this button, but also the End Operation button.</p> <pre data-bbox="727 1539 1382 1808">def getActiveOperations(self, eqPath): filter = system.mes.object.filter .createFilter() # filter.setMESObjectTypeName ('All Equipment') filter.setPrimaryMESObjectPath (sitePath)</pre>

Step	Description
	<pre> list = system.mes. searchMESObjects(filter) hdr = ['Line','Path', 'Type','Name', 'UUID'] data = [] for item in list: name = item.getName() mesObject = item. getMESObject() eqPath = mesObject. getEquipmentPath() opList = system.mes. getCurrentOperations(eqPath) for opItem in opList: data.append([name, eqPath, opItem. getMESObjectType(),opItem. getName(), opItem.getUUID()]) segList = system.mes. getCurrentSegments(eqPath) if segList.size() != 0: for segItem in segList: data.append([name, eqPath, str(segItem. getMESObjectType()), segItem. getName(), segItem. getMESObjectUUID()]) else: data.append([name, eqPath, '', '', '']) data = system.dataset.toDataSet (hdr,data) data = system.dataset.sort(data, "Name", False) self.parent.getComponent('pTableS egments').data = data </pre>
<ul style="list-style-type: none"> Click <div data-bbox="315 1667 626 1734" data-label="Image"> </div> <p>button</p>	<p>Click here to view the script...</p> <div data-bbox="699 1703 1430 1801" data-label="Text"> <pre>#Create an instance of the Operation</pre> </div>

Step	Description
	<pre> eqPath = event.source.parent. getComponent('MES Object Selector Line').equipmentItemPath operationName = 'Mix Any Nuts' autoAssignOptions = False #We don't want the operation to automatically grab resources as will determine what will be used #If there may be more than one active operation on this equipment then we need to call oper = system.mes.createOperation (operationName, eqPath, autoAssignOptions) oper.begin() #Begin the Operations Response #If we use the two lines below to start the segment, we will be required to provide a qty for the Material in property... #seg = oper.createSegment(operation) #seg.begin() #Begin the Response Segment #..instead, we will use the following script so that we don't error out on qty for Material in seg = system.mes. createSegmentForOperation(oper. getUUID(), operationName, autoAssignOptions) seg.begin() #The rest is just controls on this screen event.source.parent. OperationInProgress = True event.source.parent.getComponent('num Total').intValue = 0 event.source.parent.getComponent('txt LotNumber').text = '' #Update the table with active operations event.source.parent.getComponent('pTa bleSegments').getActiveOperations (eqPath) </pre>

- Select Input Lots and click



for each lot that you wish to add

Select as many lots from the silo as you want. You can do this using the Shift or CTRL key to select multiple lots, or you can select a single lot.


[Click here to view the script...](#)

```
eqPath = event.source.parent.  
getComponent('MES Object Selector  
Line').equipmentItemPath  
oper = system.mes.getCurrentOperation  
(eqPath)  
operName = oper.getName()  
if oper is not None:  
    seg = oper.getActiveSegment  
(operName)  
    segUUID = seg.getUUID()  
    data = event.source.parent.  
getComponent('Power Table').data  
    rowList = event.source.parent.  
getComponent('Power Table').  
getSelectedRows()  
    totalQty = event.source.parent.  
getComponent('numTotal').intValue  
    for row in rowList:  
        matDefName = data.getValueAt  
(row, 'MaterialName')  
        lotNumber = data.getValueAt  
(row, 'LotNumber')  
        seqNo = data.getValueAt(row, '  
LotSequence')  
        qty = data.getValueAt(row, 'Ne  
tQuantity')  
        print "Setting Material In  
to " + lotNumber + " Seq # " + str(se  
qNo) + " Qty " + str(qty)  
        seg.setMaterial('Material In'  
, lotNumber, seqNo, qty)  
        seg.update()  
        seg = system.mes.  
loadMESObject(segUUID) #This is  
important, we need to reload the  
segment before performing another .  
update()
```

Step	Description
	<pre> totalQty = totalQty + qty event.source.parent.getComponent('numTotal').intValue = int(totalQty) event.source.parent.getComponent('Tracer').inputParameters = '' inputParameters = "{ 'lotNumber' = ' " + lotNumber + "' }" event.source.parent.getComponent('Tracer').inputParameters = inputParameters else: system.gui.messageBox("No operation is currently running on " + eqPath) </pre>
<ul style="list-style-type: none"> • Select Mixed Nuts for the type of material lot to create 	<p>The MES Object Selector has a filter to only show this type. You may want to allow other types and that's ok.</p>
<ul style="list-style-type: none"> • Select the Location that the Material Lot will go to 	<p>Again this MES Object Selector is filtered to only show equipment in the Mixed Nut Bins. You may to change that for your implementation.</p> <p>If your <i>Reject Bin</i> equipment doesn't show up, press the  button. This button has a script that will clear the current selection and then force itself to refresh its data from the gateway. When the MES Object Selector is first instantiated on the screen, it pulls data from the gateway. If you add equipment, this selector will not reflect that change until it has .update() called on it.</p> <p>Click here to view the script...</p> <pre> event.source.parent.getComponent('MES Object Selector Bins'). clearSelection() event.source.parent.getComponent('MES Object Selector Bins').update() </pre>

Step	Description
<ul style="list-style-type: none"> Enter Quantity 	<p>The Quantity is automatically calculated from the input lots selected, but you will probably want to split up the value between the different lots out.</p>
<ul style="list-style-type: none"> Enter a Lot Number 	<p>We could have defined this to be auto-generated in the process segment, but we set it to Manual so we'll have to pass it here. We have a check in the script behind the Create Lot(s) Out button to check for duplicate lot numbers.</p>
<ul style="list-style-type: none"> Click <div data-bbox="316 682 620 735">  Create Lot(s) Out </div> button 	<p>You can go back and keep adding as many different material out lots as you want.</p> <p>Click here to view the script...</p> <pre data-bbox="727 856 1383 1816"> eqPath = event.source.parent. getComponent('MES Object Selector Line').equipmentItemPath oper = system.mes.getCurrentOperation (eqPath) operName = oper.getName() if oper is not None: matName = event.source.parent. getComponent('MES Object Selector Material').selectedName location = event.source.parent. getComponent('MES Object Selector Bins').selectedName lotNumber = event.source.parent. getComponent('txtLotNumber').text #Make sure this lot number doesn't already exist try: system.mes. getLotInventoryByLot(lotNumber) system.gui.messageBox("The lot number " + lotNumber + " has already been used. Please select another") except: seg = oper.getActiveSegment (operName) segUUID = seg.getUUID()</pre>

Step	Description
	<pre> qty = event.source.parent. getComponent('numTotal').intValue print "Setting Material Out to " + lotNumber + " " + matName + " Qty " + str(qty) seg.setMaterial('Material Out', matName, location, lotNumber, qty) seg.update() seg = system.mes. loadMESObject(segUUID) #This is important, we need to reload the segment before performing another . update() event.source.parent. getComponent('Tracer'). inputParameters = '' inputParameters = "{ 'lotNumbe r'='" + lotNumber + "'" }" event.source.parent. getComponent('Tracer'). inputParameters = inputParameters else: system.gui.messageBox("No operation is currently running on " + eqPath) </pre>

Step	Description
<ul style="list-style-type: none"> Click the  button 	<p>Click here to view the script...</p> <pre> eqPath = event.source.parent. getComponent('MES Object Selector Line').equipmentItemPath oper = system.mes.getCurrentOperation (eqPath) if oper is not None: #End segments segNameList = oper. getActiveSegmentNames() for segName in segNameList: seg = oper.getActiveSegment (segName) seg.end() else: system.gui.messageBox("No operation is currently running on " + eqPath) </pre>

Take some time to look at the scripts we are using here, from creating the operation, to adding materials and ending the run. This set of scripts should serve you well in your own implementation. The way we create operations and assign materials follows some set steps that if missed will cause problems. By following the scripts we have laid out here, you can avoid some of the common mistakes we see happen which include:

- Not setting `AutoAssignOption` to `False` - Unwanted material lots are consumed.
- Creating an operation segment directly, not from an Operation - Material lots are automatically consumed and you cannot start simultaneous operations.
- Calling `seg.update()` repeatedly but not reloading the segment object - `.update()` is only occurring on the segment object at the Gateway, segment object in client memory has not been updated.
- MES Object Selector component not showing newly created equipment - Need to call `.update()` on component after object has been created.

3.2.8 Using Custom Properties

In this section, we'll learn about how custom properties can be used to store any type of production data on any MES Object.

✔ What You'll Learn In This Section

- Do's and Don'ts of using custom properties

Custom properties can be added to any MES Object using the [MES Object Editor](#) or through scripting using the object . [addCustomProperty\(\)](#) functions. How to use custom properties and where they reside is dependent upon the application, the purpose and how you will want to access those values for analysis.

Custom properties could be added to a:

- **Work Order** objects to hold meta data perhaps about which specific raw material lots must be used against a work order, or process specs.
- **Material Definition** objects that store perhaps internal as well as external ID's for that type of material or custom properties could be added at a Class level and then every object within that class will derive those properties. In this case it is important to note that the custom property will exist at the child object, but the value will not. See [Setting Custom Properties on Material Definitions](#) for more information.
- **Material Lots** to track process data about that lot, but a word of caution here. Every time a material lot is used by an operation, a new instance of the material lot object is created. It may have the same lot number, but internally the lot sequence number of the lot object has been incremented. This would require all custom property values to be copied over to the new lot object, and now we have a situation where we are storing a lot of duplicate data on multiple objects. For material lots, it is recommended that only data specific to that instance of a lot object is maintained as a custom property. If the data is a process setting of the operation, then store that data as a custom property of the operation. In the trace graph we can access that process setting on the response segment of the operation that the lot was used or created from.

If you have a large amount of meta data that you want to store, then you may want to consider creating a custom table to hold that data and use the UUID or lot number of the lot object as a reference to access the data. Custom properties do add an overhead. Whenever you access an MES Object, the custom properties of that object will also be loaded.

Finally, consider the type of analysis you will want to do with data that you are considering using custom properties for. You may be better served using the recipe module to store recipe parameters, the SPC module for spec and actual process parameters, the OEE module and additional factors for production related data, the tag historian for time-related process process data, or custom table or web service interfaces to access data from other systems.

For the practical, we will add two custom properties to the **Mix Any Nuts** operation that will allow us to track the filter screen size used and the rotation speed on the mixer.

- In the MES Object Editor screen, edit the **Mix Any Nuts** process segment and add two custom properties, **Filter Screen Size** and **Mix Speed**.
- Set up both custom property values as shown in the image on the right.
- Update the **Mix Any Nuts** operations segment when prompted.

Process Segment, Mix Any Nuts ✓ Save

Core Properties

Custom Properties


Filter Screen Size no initial value mm

Name	Filter Screen Size
Value	
Units	mm
DataType	Int4
Description	Hole size of filter screen
Production Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Custom Properties	

Mix Speed no initial value rpm

Name	Mix Speed
Value	
Units	rpm
DataType	Int4
Description	Armature speed
Production Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Custom Properties	

- In the **Generic Mixing Operation** screen, Start a new mixing operation and add a material lot in.
- In the trace graph for the select lot, hover over the **Mix Any Nuts** node. In the Operations info panel to the right, you should see the custom properties.

Enter Filter Screen Size	<input type="text" value="10"/>	mm	Enter Mix Speed	<input type="text" value="20"/>	rpm	
--------------------------	---------------------------------	----	-----------------	---------------------------------	-----	---

- Use the container to enter values for both custom properties. When you hover over the **Mix Any Nuts** node now, you should see the updated values.
- **End** the operation

All the work here is done in the  `actionPerformed` script.

[Click here to view script...](#)

```

eqPath = event.source.parent.parent.getComponent('MES Object
Selector Line').equipmentItemPath
oper = system.mes.getCurrentOperation(eqPath)
operName = oper.getName()
if oper is not None:
    #Let's set up the key value pairs for the custom properties in a
python dictionary
    filterSize = event.source.parent.getComponent('numSize').intValue
    mixSpeed = event.source.parent.getComponent('numRPM').intValue
    cpFilter = "Filter Screen Size"
    cpMixSpeed = "Mix Speed"
    cpDict = {cpFilter:filterSize, cpMixSpeed: mixSpeed}
    #Get the Active segment
    seg = oper.getActiveSegment(operName)
    #Let's check that the custom properties exist for the response
segment
    cpList = seg.getCustomProperties() #This function returns an
MESObjectCollection object
    for key in cpDict:
        if cpList.containsKey(key):
            seg.setPropertyValue(key, cpDict[key])
        else:
            system.gui.messageBox("Custom Property " + key + "
doesn't exist for this response segment. Available Custom Properties
are " + str(cpList))
            import sys
            sys.exit(0)
    system.mes.saveMESObject(seg) #Don't forget to save
    print seg.getAllCustomProperties() #This function returns an
arrayList object

```

Enter Filter Screen Size mm Enter Mix Speed rpm

A1004 ☒ Include Seg Info

Mixing Line 1

Mix Any Nuts

Mix Any Nuts

Started: Thu Jan 11 16:05:24 PST 2018
Ended: -

Filter Screen Size: 10
Mix Speed: 20

Material - In

Material In: (unused)
Material In-1: Active
Lot No: A1004
Material: Bulk Almonds
Location: Almond Silo
Quantity: 1000.0
Begin Date Time: Thu Jan 11 16:05:35 PST 2018
End Date Time: -

3.2.9 Tracking Personnel

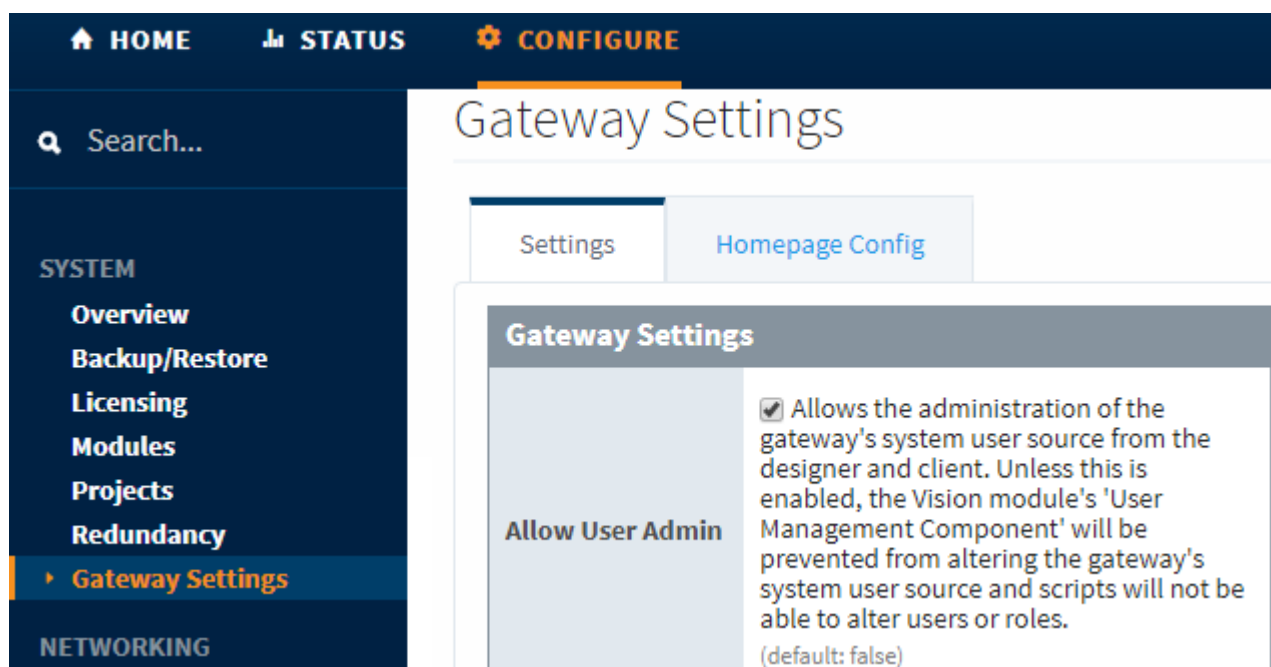
In this section, we'll learn about creating users, adding them to personnel classes, and then tracking operations by operator.

👍 What You'll Learn In This Section

- Difference between Ignition roles and Personnel Classes.
- Production Control based on certified operators.
- Tracking operations by operator.

We are going to create some users using the Ignition **User Management** component. Before we can do that, we need to change a Gateway Setting to allow the designer and client access to modify the gateway's system user source.

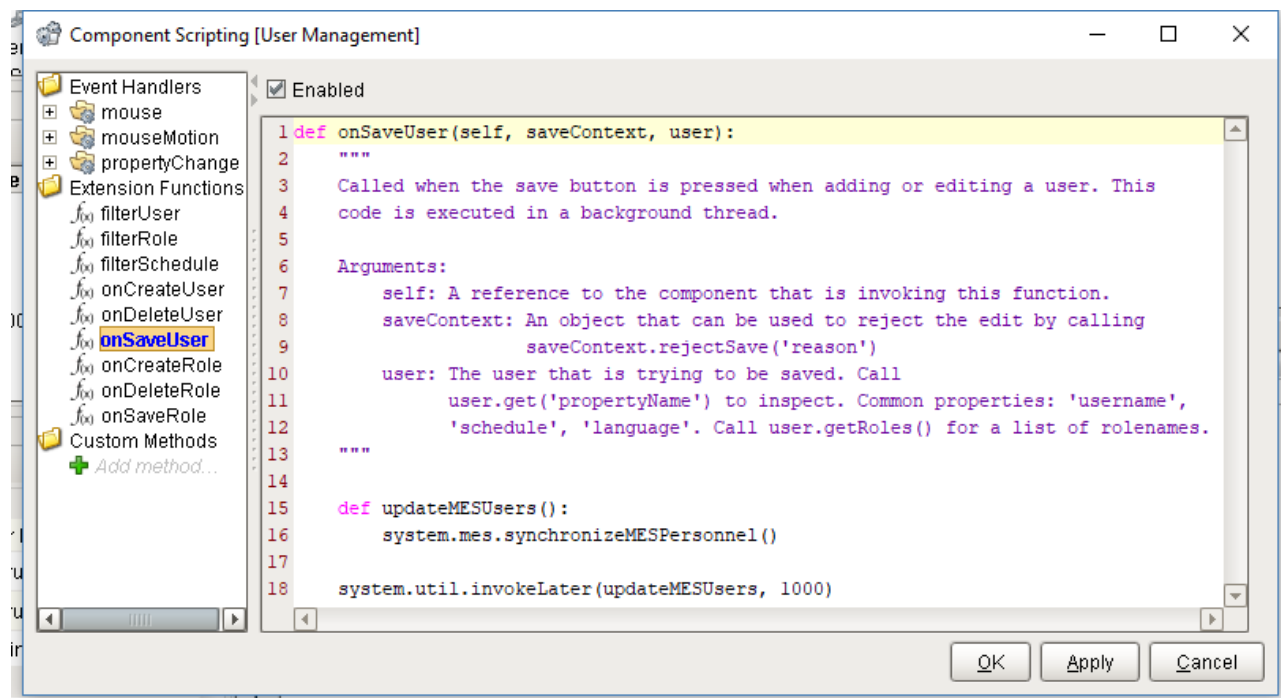
- On the Gateway under **Configuration**, select the **Gateway Settings** page and enable the **Allow User Admin** option.
- Create a new Main Window called **User Manager** under the **Administration** folder.
- Drag the **User Management** component from the **Admin** palette onto the new window.
- **Save** your changes.



MES Personnel objects are automatically created for any users added to any of the Ignition User Sources. However the synchronization of users only occurs once every hour and may not happen immediately. We can force this by calling the `system.mes.synchronizeMESPersonnel()` script function.

- Add the following script to the **onSaveUser** extension function of the **User Management** component.

```
def updateMESUsers():  
    system.mes.synchronizeMESPersonnel()  
  
system.util.invokeLater(updateMESUsers, 1000)
```



- Go into preview mode by pressing **F5** and add some users.

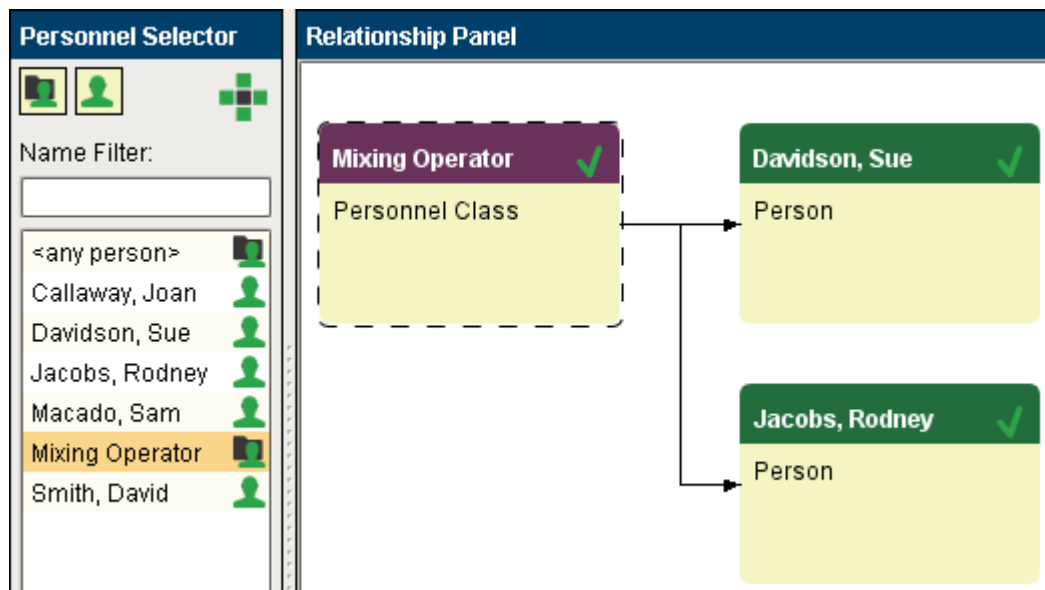
We have created the following...

Users				
Users				
Username	Name	Roles	Contact Info	Schedule
admin		Administrator		Always
dsmith	David Smith			Always
jcallaway	Joan Callaway			Always
rjacobs	Rodney Jacobs			Always
smacado	Sam Macado			Always
sdavidson	Sue Davidson			Always

In the MES Object Editor screen, you should now see all the users you created in the User Management screen. You may need to click on Material tab and then back to Personnel tab to make it refresh.

Now we will make a couple of Personnel Classes that will hold only those personnel that are certified to perform certain operations. If the generic <any person> doesn't exist, go ahead and create that too.

- Add a new Personnel Class called <any person> if it doesn't already exist.
- Add a new Personnel Class called **Mixing Operator**.
- Drag some of the users you created into the **Mixing Operator** class.



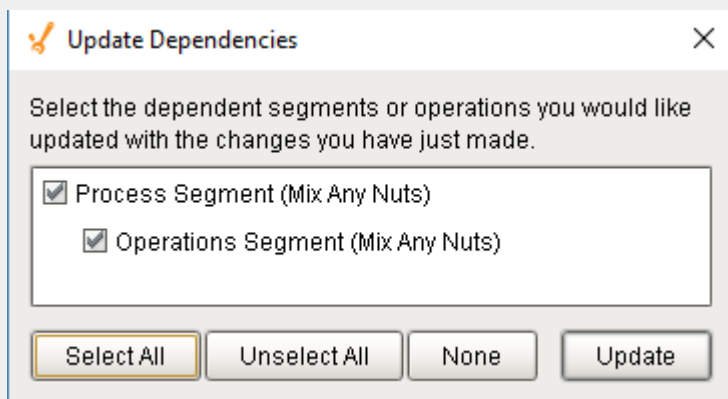
Difference Between Roles and Personnel Classes

When we added users using the Ignition User Management component, we didn't create or associate our users to any roles. Roles are used by Ignition as part of the security implementation as to who access to do what, from logging into a project, to accessing a screen or pressing a button. Every component has a security setting where this can be defined.

Personnel Classes, on the other hand, group users together that can then be associated to operations. For your implementation, ignition security roles and personnel classes may be one and the same, and you could simply use script to create and add users to a personnel class whenever they are given a certain role. Production control over who can start an operation can also be a mixture of ignition security roles and membership of a personnel class. You may simply use Track & Trace to track who the operator was and ignition security roles to control who can perform certain operations.

We need to go back and modify our process segment to add a Person Complex Property. We could add it to the **Mix Any Nuts** process segment, but instead we'll add it to our generic **Any Operation** process segment, that way we'll see how changes made can be pushed back to derived objects.

- In the **MES Object Editor** screen, right-click on the **Any Operation** process segment and select **Edit Settings**.
- Under **Personnel**, add a new complex property called **Person**.
- Set **Optional** to *True* and **Personnel Reference** to *Personnel Class, <any person>*
- When prompted to Update Dependencies, **Select All** and then **Update**.



We could also have made the personnel reference of the **Mix Any Nuts** process segment be to the **Mixing Operator** class. That way only members of the Mixing Operator class would be allowed to execute this operation. But we are setting all the operation parameters by scripting and we'll enforce production control there.

Process Segment, Any Operation

✓ Save

Core Properties

Name	Any Operation
Description	Generic Operation
End Operation When Complete	<input checked="" type="checkbox"/>
Segment Recipe Name	<no recipe selected> ▼

Custom Properties

none defined	
--------------	--

Material

Material In	In, Material Class, <any material>
Material Out	Out, Material Class, <any material>

Equipment

New Equipment	Equipment Class, <any equipment>
---------------	----------------------------------

Personnel

Person	Personnel Class, <any person>
--------	-------------------------------

- In the **Generic Mixing Operation** screen, Start a new mixing operation and add a material lot in.
- In the trace graph for the select lot, hover over the **Mix Any Nuts** node. In the Operations info panel to the right, you should see a **Personnel** section now.

- Use the

Select Mixing Operator Jacobs, Rodney ▼

 container to select an operator. When you hover over the **Mix Any Nuts** node now, you should see the updated values.
- Go ahead and select another operator if you'd like.

You can have multiple personnel associated with an operation. If you have a single person complex property the timestamp of when the person was operating the line will be recorded. If you have multiple people working on a line at the same time, you may want to have additional person complex properties for each team member, or perhaps use a custom property to track which team was operating the line.

- **End the operation**

<div>Mixing Line 1</div> <div>Mix Any Nuts</div>	Location: Peanut Silo
	Quantity: 1000.0
	Begin Date Time: Fri Jan 12 08:00:50 PST 2018
	End Date Time: -
	Material - Out
	Material Out: Active
	Lot No: MN1002
	Material: Mixed Nuts
	Location: Bin 0002
	Quantity: 1000.0
Begin Date Time: Fri Jan 12 08:00:10 PST 2018	
End Date Time: -	
Personnel	
Person: (unused)	
Person: Davidson, Sue	
Begin Date Time: Fri Jan 12 08:00:16 PST 2018	
End Date Time: Fri Jan 12 08:01:10 PST 2018	
Person: Jacobs, Rodney (active)	
Begin Date Time: Fri Jan 12 08:01:10 PST 2018	
End Date Time: -	

All the work here is done in the *actionPerformed* script behind the  button.

[Click here to view script...](#)

```

import java.lang
eqPath = event.source.parent.parent.getComponent('MES Object
Selector Line').equipmentItemPath
oper = system.mes.getCurrentOperation(eqPath)
operName = oper.getName()
if oper is not None:
    seg = oper.getActiveSegment(operName)
    segUUID = seg.getUUID()
    personnelPropertyName = "Person"
    personName = event.source.parent.getComponent('MES Object
Selector Person').selectedName
    try:
        seg.setPersonnel(personnelPropertyName, personName)
        seg.update()
    except java.lang.Exception, e:
        system.gui.messageBox(e.getMessage())
        event.source.parent.getComponent('MES Object Selector Person').
clearSelection()
    #Force the Trace Graph to refresh
    tracer = event.source.parent.parent.getComponent('Tracer')
    inputParameters = tracer.inputParameters
    tracer.inputParameters = ''
    tracer.inputParameters = inputParameters
else:
    system.gui.messageBox("No operation is currently running on " +
eqPath)

```

3.3 Track and Trace Review

This concludes the Track & Trace portion of the online tutorial. In the next section we shall learn about the OEE 2.0 Module and continue the Nuts Unlimited project by adding a packaging line where we will schedule production runs, capture OEE metrics, and continue tracking the lots that are consumed by the packaging operation. During the OEE tutorial we will cover Work Order Management, Production Scheduling and Sub-Lots which are also available to T&T.

3.3.1 What We Covered

In summary, we have learnt the following so far:

- How to setup a gateway server for an MES application
- What the ISA-95 standard covers
- How to create a material flow and model the plant in the production model
- Configuring lot handling modes
- Using classes and definitions for grouping materials and equipment
- Creating process segments
- Accessing WIP inventory data
- Lot tracking using the Trace Graph
- Deriving objects
- Executing operation segments using scripting
- Using custom properties
- Using personnel resources in process segments

3.3.2 What We Didn't Cover

What we haven't covered or touched in great detail are:

- The MES scripting functions
- The MES Object model
- Production Routing
- How to Configure a Process Segment for Lot Blending
- Sub-lots (covered in next section)
- Scheduling (covered in next section)
- Creating custom lot numbers

Scheduling will be covered in the next section as part of OEE, but any operation can be scheduled. In fact operations can be linked into routes that can then be scheduled. We also only covered creating a process segment for which we automatically created a corresponding operation definition and operation segment, but operations definitions can have multiple operations segments in series or parallel.

3.3.3 Additional Resources

We can't cover everything in a tutorial, but we do have more resources available in the [MES Help manual](#) and [Sepasoft Knowledge Base](#) for the areas we didn't cover. The knowledge base articles provide real-world examples of how to extend the framework and provide a number of scripting examples.

» OEE 2.0

- › Features
- › Framework
- › Practical—Package Nuts
- › OEE 2.0 Review

4 OEE 2.0


Welcome to the OEE 2.0 Module! In this section, we will walk through some of the help documentation to familiarize ourselves with the OEE 2.0 module framework and features. Then we will continue building out the Nuts Unlimited project by adding a packaging line for the nuts, scheduling operations on it and capturing OEE and production data.

4.1 Features

4.2 Framework

4.3 Practical - Package Nuts

In this section we'll continue building our Nuts Unlimited project to add a nut packing line and capture OEE metrics on it. We will use the material lots created from the nut mixing section to pack them into containers before being stored in the warehouse. The operation will be controlled by an operator interface that we will create.

-  Download the [MES_2.0_Training_OEE_Base.proj](#) file.
- Select the **File->Import** menu in the Ignition designer and navigate to the folder where you downloaded the project file to and import the windows and templates.
- **Save** your changes.

4.3.1 Configure Packaging

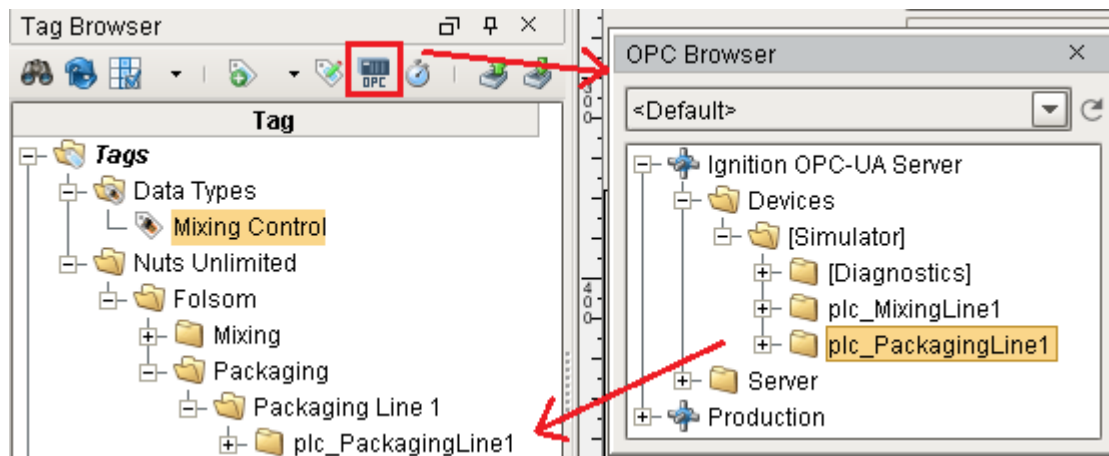
Although the OEE module uses the same framework and object model as Track & Trace, it also provides some additional components that we can use to configure the application rather than use the [MES Object Editor](#). We will need to configure the equipment, shifts and material definitions (product codes), so we'll see how to use these components in this section. Let's start by setting up the Packaging Line simulator so we have some tags to simulate equipment states and counts.

Setup The Packaging Line Simulator

- Create the following folder structure in the Tag Browser:
 - **Nuts Unlimited\Folsom\Packaging\Packaging Line 1**

We'll use this same folder structure for our equipment UDT tags. Creating a tag folder structure that matches with your Packaging Line equipment path will allow us to easily reference our UDT based on the selected equipment path when a user selects a line to view in a screen.







- Using the OPC Browser, drag the **plc_PackagingLine** folder from the Simulator Device into the **Packaging Line 1** folder.



Modeling the Packaging Line

The OEE module can handle complex process lines and work centers where there are multiple input and output streams and parallel process cells, but for this tutorial we'll create a simple packaging line with multiple cells in a contiguous sequential process.

- Add the following production items to the production model.

Icon	Production Item	Value	Comment
	Area	Packaging	
	Line	Packaging Line 1	
	Cell	Filler	Key Cell
	Cell	Checkweigher	
	Cell	Casepacker	
	Cell	Palletizer	


- Click on the **Packaging Line 1** production item and set the *Downtime Detection Mode* to Key Reason (Neighbor Priority) in the OEE 2.0 Downtime tab.
- **Save** your changes.

Create PLC Interface

As a best practice to make our OEE implementation scalable and to speed up implementation, we'll use a UDT to create a standard interface between the source of the line data (PLC) and the Production Model. In our case, the source of the data will come from our production simulator, but it could also come from user input.

114

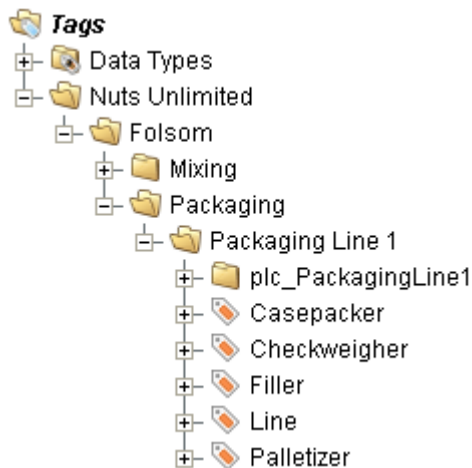
Instead of directly binding the plc tag to the production model, we will give ourselves the opportunity to perform some data validation and transformation on the data before passing it onto the Production Model for data capture and storage. It may be that our equipment status is based on the combined value of a number of plc tags or perhaps the count tags at the plc occasionally go negative due to ladder rung scanning or get set to zero if the modbus connection is lost. We can handle these scenarios by modifying the expression or the tag change event on the Input tag to set the value of the Output tag. We can handle peculiarities with specific equipment at the UDT level, allowing us to create standard screens that will work with all lines in all areas across sites.

-  Download the eqOEE.xml [here](#) and import it in the tag browser. You will need to right-click and save it to a file location.
- Import the tags into the Tag Browser.

Now we will create instances of the eqOEE UDT for the Packaging line and cells.

- Right-click on the **Nuts Unlimited\Folsom\Packaging\Packaging Line 1** folder and create a new tag using the **eqOEE** Data Type Instance for each cell in the line and also for the Line.

You should have the following tag structure now under Packaging in the tag browser.












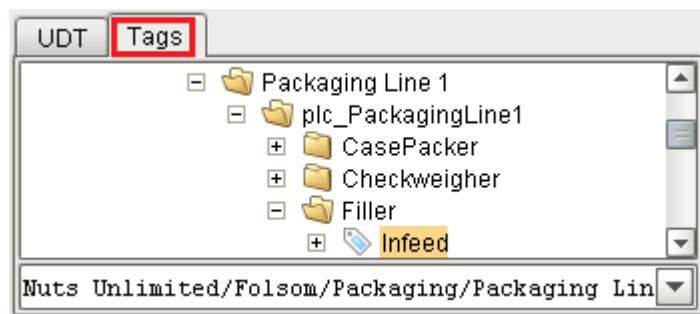
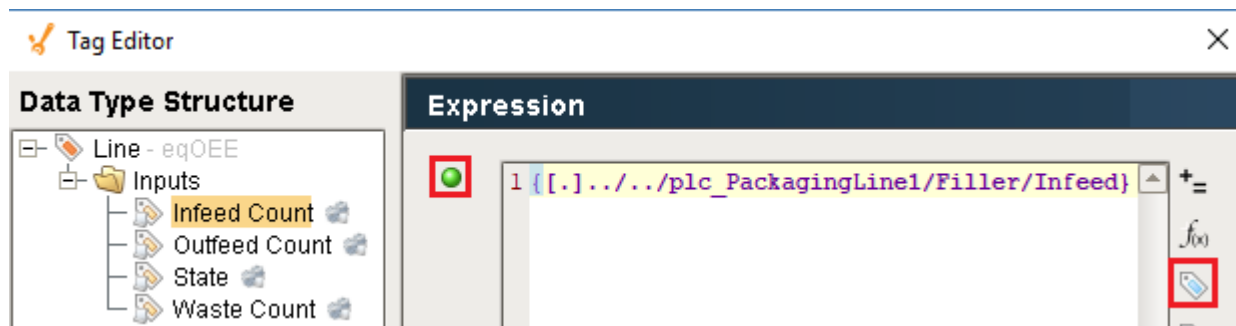
- ✓ Ignition provides scripting functions that allow you create any type of tag including tags of datatype UDT and also override inherited values as you create them. We've included a code snippet that creates the packaging line UDT instances for you, by going through the production model and creating instances for the line and each cell under the line. This code snippet is shown as an example and can be run, however the exported tags are provided further below if you want to fastTrak this section.

Setting up the Line Count Tags

We will now bind the count tags in the **Inputs** folder of the Line UDT instance to the respective simulator tags. Infeed count will come from the **Filler Infeed** tag, outfeed from the **CasePacker Outfeed** tag and waste count from **Checkweigher Waste** tag.

We are overriding the UDT expression as these bindings are specific to this instance. The tag change event on these tags will validate count changes before writing them to the memory tags in the Outputs folder. These output tags will be bound to the MES Counters for Packaging Line 1.

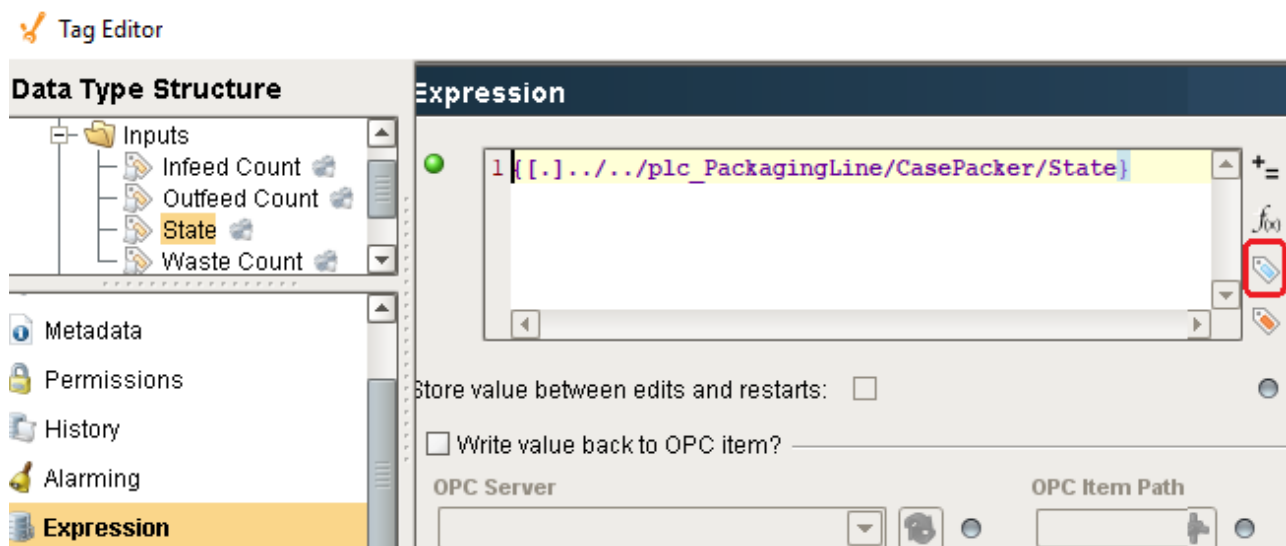
- Right click on the Line UDT instance and select **Edit Tag(s)**
- In the **Inputs** Folder...
 - Select **Infeed Count** and then **Expression**
 - Override the inherited parent expression by clicking on the grey dot  so that it becomes green 
 - Click on the  icon, select the **Tags** tab and navigate to *Nuts Unlimited /Packaging/Packaging Line 1/plc_PackagingLine1/Filler/Infeed*
- Select **Outfeed Count** and then **Expression**
 - Override the inherited parent expression by clicking on the grey dot  so that it becomes green 
 - Click on the  icon, select the **Tags** tab and navigate to *Nuts Unlimited /Packaging/Packaging Line 1/plc_PackagingLine1/Casepacker/Outfeed*
- Select **Waste Count** and then **Expression**
 - Override the inherited parent expression by clicking on the grey dot  so that it becomes green 
 - Click on the  icon, select the **Tags** tab and navigate to *Nuts Unlimited /Packaging/Packaging Line 1/plc_PackagingLine1/Checkweigher/Waste*
- Click **OK**



Setting Up The Equipment State Tags

We'll be using [Key Reason Detection](#) to determine overall line status from the state of each cell. With Key Reason, we select a primary cell on the line that as long as it is running, we will consider the line as running. When the primary cell goes down, the cell causing the primary cell to go down will be automatically determined. For our tutorial, that will be Filler cell.

We will now bind the simulator tags to the Inputs\Stats tag for each cell UDT tag.



- Right click on the Casepacker UDT instance and select **Edit Tag(s)**
- In the **Inputs** Folder...
 - Select **State** and then **Expression**
 - Override the inherited parent expression by clicking on the grey dot  so that it becomes green 
 - Click on the  icon, select the **Tags** tab and navigate to *Nuts Unlimited /Packaging/Packaging Line 1/plc_PackagingLine1/Casepacker/State*
- Click **OK**
- Right click on the Checkweigher UDT instance and select **Edit Tag(s)**
- In the **Inputs** Folder...
 - Select **State** and then **Expression**
 - Override the inherited parent expression by clicking on the grey dot  so that it becomes green 
 - Click on the  icon, select the **Tags** tab and navigate to the *Unlimited/Packaging /Packaging Line 1/plc_PackagingLine1/Checkweigher/State*
- Click **OK**
- Right click on the Filler UDT instance and select **Edit Tag(s)**
- In the **Inputs** Folder...
 - Select **State** and then **Expression**
 - Override the inherited parent expression by clicking on the grey dot  so that it becomes green 
 - Click on the  icon, select the **Tags** tab and navigate to *Nuts Unlimited /Packaging/Packaging Line 1/plc_PackagingLine1/Filler/State*
- Click **OK**
- Right click on the Palletizer UDT instance and select **Edit Tag(s)**
- In the **Inputs** Folder...
 - Select **State** and then **Expression**
 - Override the inherited parent expression by clicking on the grey dot  so that it becomes green 
 - Click on the  icon, select the **Tags** tab and navigate to *Nuts Unlimited /Packaging/Packaging Line 1/plc_PackagingLine1/Palletizer/State*
- Click **OK**

Connect the Production Model

We are only interested in production counts at the line level to obtain OEE Performance and Quality metrics, so we will only bring count tags in at the line level of the production model. Line state will be driven by the cell states, so we will bring in all equipment states.



You can have OEE calculated at the cell level too in which case you would bring in count tags at the cell level too. You also need to be licensed either with a site license or multiple machine licenses. Bringing in counts at the cell level will also provide cycle time analysis for the cells.

- Select the **Packaging Line 1** production item in the Production model and click on the **General Tab**.
 - Edit the **Material Out** counter in the MES Counters pane and change the SQL tag path to *Nuts Unlimited\Folsom\Packaging\Packaging Line 1\Line\Outputs\Outfeed Count* tag. Set the **Count Mode** to **Positive Change**.
- From the Tag Browser
 - Drag the *Nuts Unlimited/Packaging/Packaging Line 1\Line\Outputs\Infeed Count* tag to the MES counter pane, rename it to **Infeed Count**, set the **Counter Kind** to **Infeed** and the **Count Mode** to **Positive Change**.
 - Drag the *Nuts Unlimited/Packaging/Packaging Line 1\Line\Outputs\Waste Count* tag to the MES counter, rename it to **Waste Count** and set the **Counter Kind** to **Reject** and the **Count Mode** to **Positive Change**.

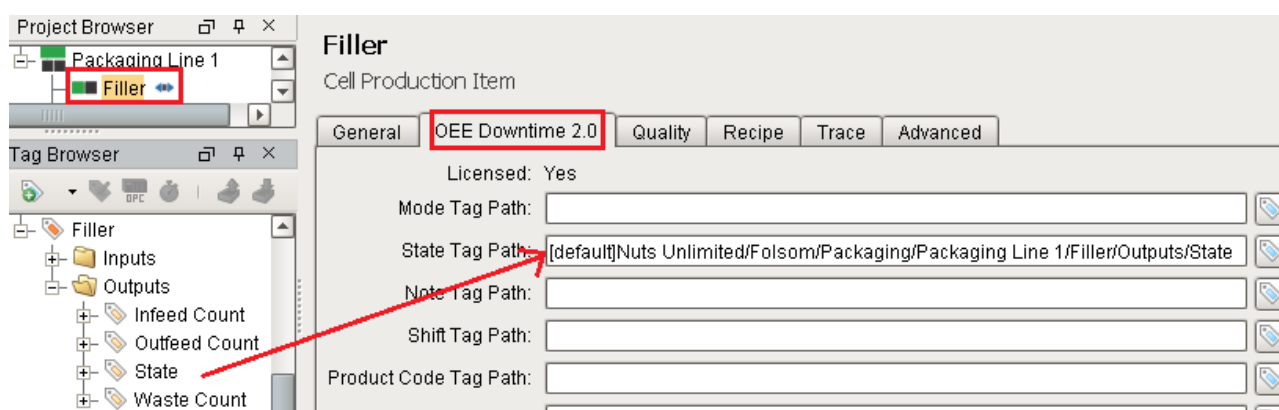
Your MES Counter pane should look as shown.

MES Counter:	Counter Name...	C...	Enabled	SQL Tag	Roll Over	Store...	Counter Kind	Count Mode
	Infeed Count		true	[default]Nuts Unlimited\Folsom\Packaging\Packaging Line 1\Line\Outputs\Infeed Count	32768	0	Infeed	Positive Change
	Material Out		true	[default]Nuts Unlimited\Folsom\Packaging\Packaging Line 1\Line\Outputs\Outfeed Count	32768	0	Outfeed	Positive Change
	Waste Count		true	[default]Nuts Unlimited\Folsom\Packaging\Packaging Line 1\Line\Outputs\Waste Count	32768	0	Reject	Positive Change

Cell Configuration

We will now configure the cells underneath the line to bring in the equipment state.


- Select the **OEE Downtime 2.0 Tab** in the Production model....
 - Drag the **Filler\Outputs\State** tag to the **Filler** cell **State Tag Path**
 - Drag the **Checkweigher\Outputs\State** tag to the **Checkweigher** cell **State Tag Path**
 - Drag the **Casepacker\Outputs\State** tag to the **Casepacker** cell **State Tag Path**
 - Drag the **Palletizer\Outputs\State** tag to the **Palletizer** cell **State Tag Path**
- **Save** your changes by clicking on the  button.



We have now bound our UDT expression state tags to the simulator and the UDT memory state tags to the production model. We don't have to do anything with the line state tag at the UDT or production model as we have set up the line to use **Key Reason**. However, if we wanted to override a line downtime event caused by a cell with a line reason, say **E-Stop Pulled**, we could bring a state tag for line reasons that are not specific to a cell.




Setup Equipment States

Let's now create a screen that we'll use to configure the Packaging Line, as well as any other production line for OEE runs. This screen will allow us to create the equipment states and modes for this line as well as configure which shifts are enabled.

- Under Windows, create a folder called **Configuration** and add a new Main Window called **Equipment Manager**.
- Drag the OEE Equipment Manager component from the OEE Downtime 2.0 component palette onto the window and stretch it to fill the window.
- **Save** your changes by clicking the  button.

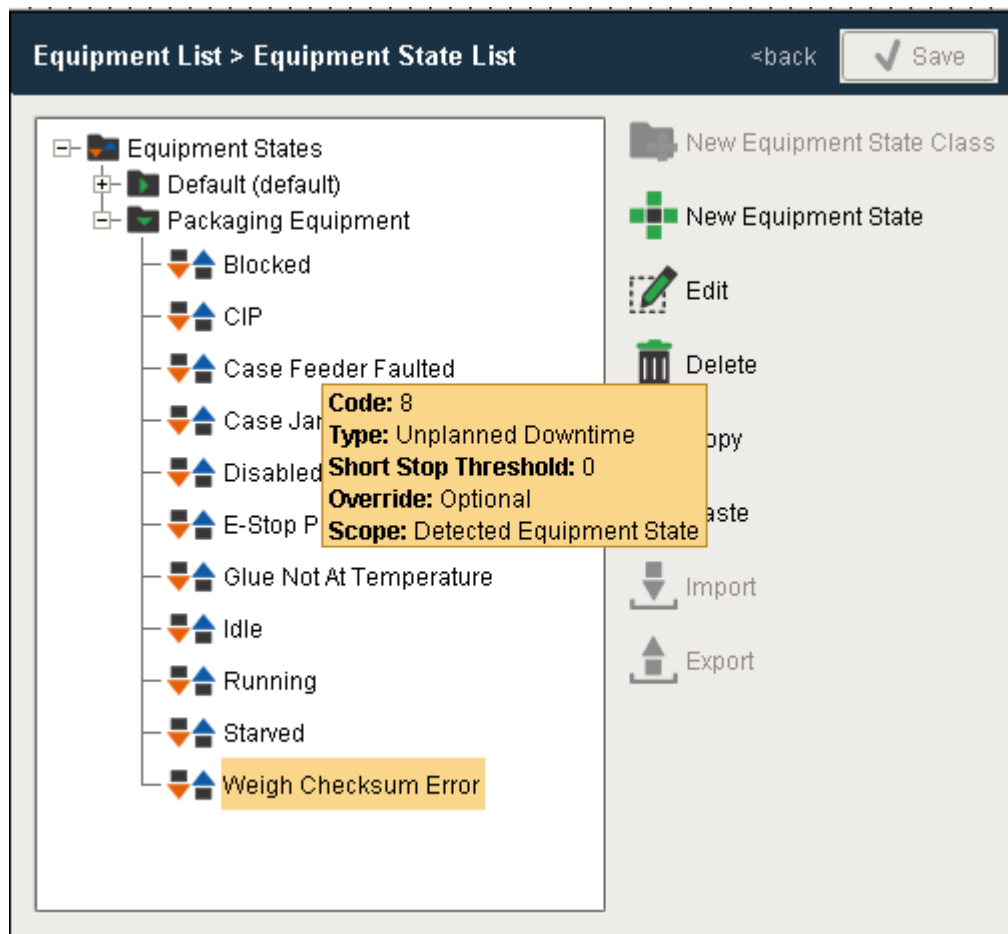
We are now done with the development side of the application of equipment configuration! All the rest can be done in the client as the [OEE Equipment Manager](#) component handles the configuration of modes and states. Have a look around in this component. We provide you with default Equipment Mode and State classes but you can also create custom modes and states and associate lines and cells to a specific class. You can also select the shifts associated with this line. We'll deal with shifts later. Refer to [Equipment Configuration](#) for more information.

The production simulator for the packaging line has specific states that we will need to add using our **Equipment Manager** window. We'll add the states right now.

- Run the **Equipment Management** window in preview mode (F5) in the designer.
- Click on **Packaging Line 1** and select the **Equipment State Class**  button.
- Click on **Equipment States** and select  to create a new Equipment State Class called **Packaging Equipment**.
- Select this new State Class and click  to start defining the states for equipment on the filler line as defined below:

State Name	Type	Code
Blocked	Blocked	4
CIP	Planned Downtime	3
Case Feeder Faulted	Unplanned Downtime	8
Case Jam	Unplanned Downtime	5
Disabled	Disabled	0
E-Stop Pulled	Unplanned Downtime	10
Glue Not At Temperature	Unplanned Downtime	9
Idle	Idle	2
Running	Running	1
Out Of Infeed Material	Unplanned Downtime	6
Weigh Checksum Error	Unplanned Downtime	7

- Save the new State Class
- Change the **Current Selection** for **Equipment State Class** for Packaging line 1 and all cells under Packaging Line 1 to use the new **Packaging Equipment** state class.



We just put all process cells states for the packaging line into a single State Class. In the real-world, we would probably have a state class for each cell, i.e. filler, checkweigher....




Configure Finished Goods

Let's create a screen that we'll use to configure the finished goods that will be created on our Packaging Line.

- Add a new Main Window called **Material Manager** in the **Configuration** folder.
- Drag the **OEE Material Manager** component from the **OEE Downtime 2.0** component palette onto the window.

With this window you can add, modify and delete material definitions and classes and associate which production lines this product can be produced on.

We are now done with the development side of the application for material configuration for OEE production lines! All the rest can be done in the client as the [OEE Material Manager](#) component handles the configuration of materials and lines.

- Go into preview mode by pressing **F5** or 
- Add a Material Class called **Finished Goods**.
- Add a Material Definition for **Mixed Nuts 16oz** and enable it   **Packaging Line 1** for Packaging Line 1.
- In the Production Settings pane, change the **Rate Period** to **Min.** and the **OEE Standard Rate** to 150.0.
- Add a Material Definition for **Mixed Nuts 8oz** and enable it for Packaging Line 1.
- In the Production Settings pane, change the **Rate Period** to **Min.** and the **OEE Standard Rate** to 150.0.

Material List > Edit Material Definition
<back
Save

Material Definition Properties

Name
Mixed Nuts 16oz

Description

Material Production Settings

Nuts Unlimited
Folsom
Receiving
Nut Unloading
Mixing
Mixing Line 1
Warehouse
Packaging
Packaging Line 1

Production Settings

Production Mode
Production

Rate Period
Min

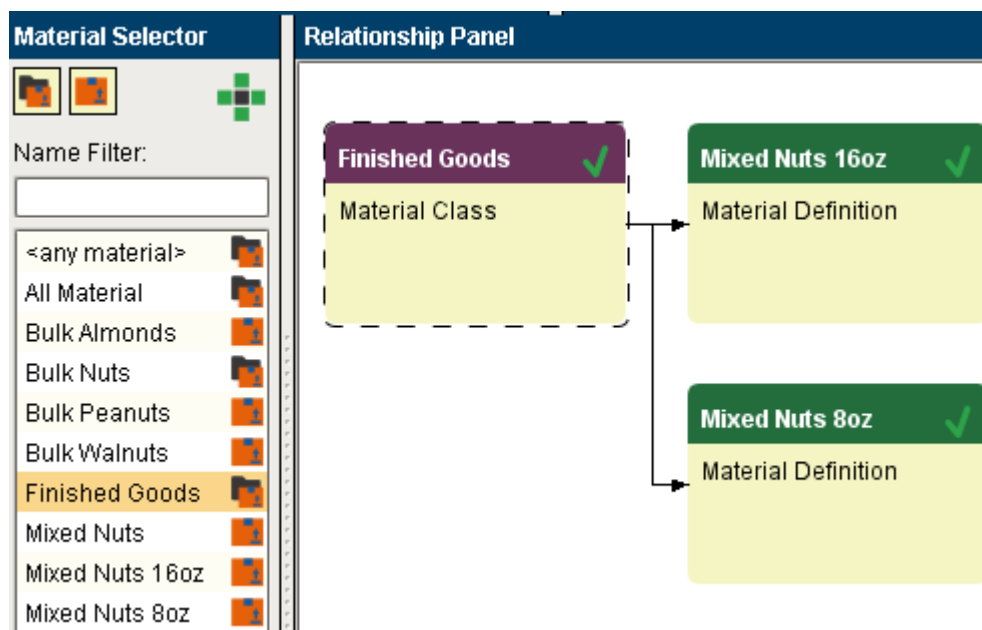
Schedule Rate
100.0

OEE Standard Rate
150.0

Infeed Count Equipment
Packaging Line 1

Infeed Count Scale
1.0

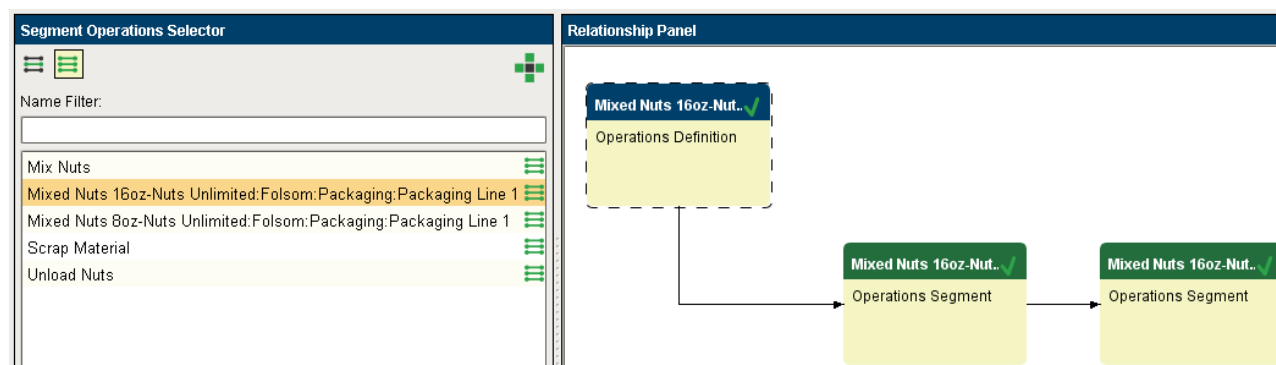
In the background, the [OEE Material Manager](#) component created the material class **Finished Goods** and two material definitions, **Mixed Nuts 8oz** and **Mixed Nuts 16oz**. These can be seen using the [MES Object Editor](#) component, and if we wanted to add custom properties, we could do so here. We could also create process segments that used these materials as input or output material references.



When we enabled Packaging Line 1 for these products in the [OEE Material Manager](#), in the background, the following operations definition and segments were automatically created for us.

- **Mixed Nuts 16oz-Nuts Unlimited:Folsom:Packaging:Packaging line 1**
- **Mixed Nuts 8oz-Nuts Unlimited:Folsom:Packaging:Packaging line 1**

The first segment under the Operations Definition is for the changeover operation, the second is for the production segment. Again if we wanted to add custom properties to associate data with these segments, we can do so here.



There are a number of material production settings that can be configured for when these products are produced on Packaging Line 1, however the default values will work just fine for us. Refer to [Product Definition Configuration](#) in the help for more information on using the Material Manager and the material production settings.

i Creating Material- Line Configuration through Scripting

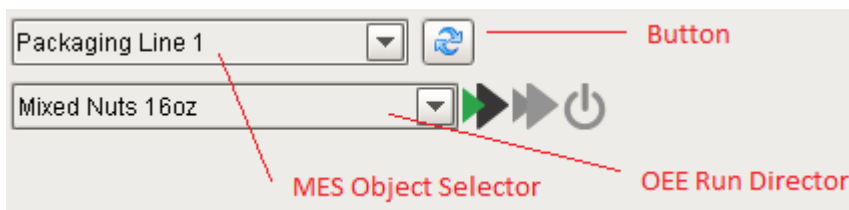
If you want to create the Material to Line configuration production settings through scripting, refer to [Creating Materials](#) in our online help manual.

Create Packaging Screen

We will now create a window that will allow us to control and monitor the nut packaging process and will require input from the operator. For now we will build this as a stand-alone OEE project with no lot tracking so that we can get familiar with the OEE 2.0 components such as the [OEE Run Director](#). Later on, we will add lot tracking so that we can trace the raw materials used by the packaging operation.

Add Run Control Components

We'll add some components that will allow us to select the packaging line and start production runs on it.



- Open the **Packaging\Packing** window that came as part of the base MES training project imported at the beginning of the tutorial.
- Add the following components of the type indicated so that they appear as shown in the image:

Component	Settings	Description
MES Object Selector	<ul style="list-style-type: none"> • Set property Include MES Line Objects to True. • Type Packaging into the Parent MES Object Name filter property. 	This will allow us to only select lines in the Packaging area.


Component	Settings	Description
Button	<ul style="list-style-type: none"> Clear the <i>textfield</i> and set the image path to <i>Builtin/icons/16/refresh.png</i>. Add the following script to the ActionPerformed event. <pre>event.source. parent. getComponent('MES Object Selector'). clearSelection() event.source. parent. getComponent('MES Object Selector'). update()</pre>	The button will allow us to clear the equipment path selected in the MES Object selector as well as force it to go back and reload MES Objects from the gateway (in case those objects have changed).
OEE Run Director	<ul style="list-style-type: none"> Set property Equipment Path binding to Root Container.MES Object Selector.equipmentItemPath. 	We'll use this component to start OEE runs on the selected line

Add Live Analysis Tags

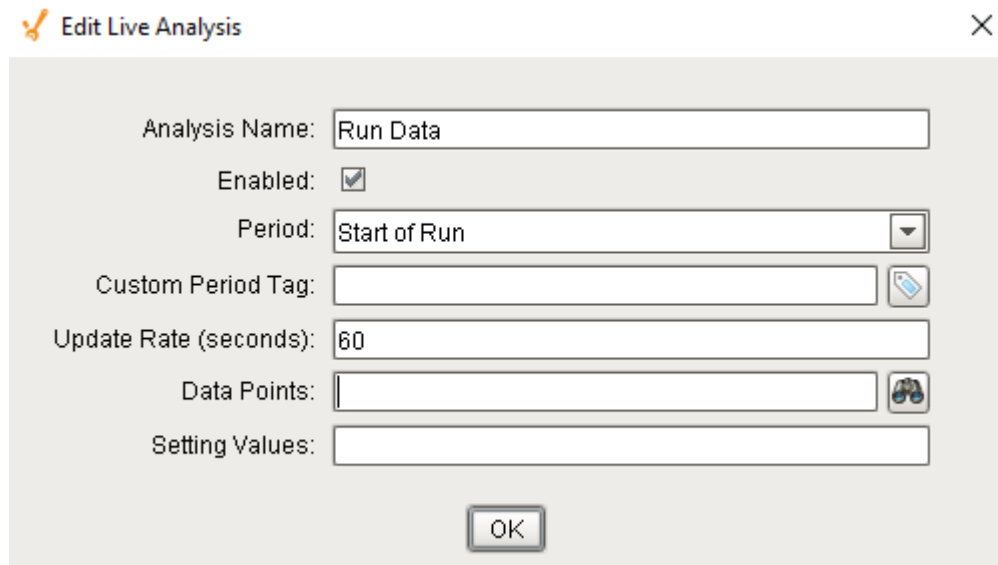
Before we start a production run on the packaging line, we're going to want to be able to see the real-time status for the line, production data and OEE metrics. To do this, we will use **Live Analysis** to expose real-time values through tags.

- Select **Packaging Line 1** in the Production Model and click on the **OEE 2.0 Downtime** tab.
- Right-click on the Live Analysis pane and select **New**.
- Set up the Live Analysis as follows:

Property	Value	Description
Analysis Name	Run Data	The name for the live analysis.
Enabled	True	Whether or not this live analysis tag is enabled
Period	Start of Run	Return values for given period.

Property	Value	Description
Data Points	OEE Infeed Count,OEE Outfeed Count,OEE Reject Count, Operation UUID,Product Code,Work Order,Equipment Mode Name,Equipment State Name,Equipment Mode Type, Equipment State Type,Line Downtime Reason,Line State Name,Line State Type,Line Downtime Equipment Name,OEE, OEE Availability,OEE Performance,Planned Downtime, Runtime,Unplanned Downtime,Elapsed Time,Line Schedule Count,Line Standard Count,Line Target Count,Schedule Rate, Standard Rate,OEE Quality,Shift,Rate Period,Infeed Standard Count,Infeed Units,Outfeed Units,Reject Units,Equipment Infeed Scale,Equipment Package Count,Equipment Reject Scale,Equipment Name	<p>Data points allows you to pick and choose the values you wish to access through tags. See the table below for the listing of available data points.</p> <div>  Tip You can copy the Data Points from the table above and paste them into the Live Analysis Data Points field or import the csv file from the FastTrak section. </div>

- **Save** and **Publish** your changes.




✂ Edit Live Analysis X


Analysis Name:

Enabled: ☒

Period:

Custom Period Tag: 

Update Rate (seconds):

Data Points: 

Setting Values:

OK

The Live Analysis automatically creates a new tag provider called **MES** that contains a tag for each selected datapoint in a folder structure that mirrors the equipment path with the analysis name appended on the end. We have created live analysis tags for the packaging line, but we could also do the same for cells and cell groups under the line. For more information refer to [Live Analysis](#) in the help documentation.

We can now drag these tags over onto our packaging screen and bind them to labels and other standard ignition components to display the values, but we earlier imported a ready made template that is already built, so we'll use this instead to speed up development.

- Add the following code to the shared Script library **MES**:

```
def getLiveAnalysisTagPath(eqPath):

    # Given an equipmentPath, this function will return the live
    analysis tag path
    # equipPath comes in the format of [global]
    \Enterprise\Site\Area\Line
    # which needs to be converted to [MES]Enterprise/Site/Area
    /Line/Live Analysis

    #print "getEquipmentStatusTagPath called for ", equipPath

    tagPath = eqPath
    if tagPath is not None:
        tagPath = tagPath.replace('\\', '/')
        if '[global]/' in (tagPath):
            tagPath = tagPath[len('[global]/'):]    #Remove
[global]\
            tagPath = '[MES]' + tagPath + '/Live Analysis'
    return tagPath
```

- Drag the **Line Run Data** template under Templates\OEE onto the packing screen.
- Add the following expression to the **LAPath** template property to pass the tag path to the Live Analysis

```
runScript("shared.MES.getLiveAnalysisTagPath",0,{Root Container.
MES Object Selector.equipmentItemPath}) + "/Run Data"
```

The global script shared.MES.getLiveAnalysisTagPath() takes an equipment path and returns a path to the live analysis tags for that equipment path. We will get the equipment path from the MES Object Selector on our screen.

132

+	Tags		
+	System		
+	Client		
-	All Providers		
+	default		
-	MES		
-	Nuts Unlimited		
-	Folsom		
-	Packaging		
-	Packaging Line 1		
-	Live Analysis		
-	Run Data		
+	Elapsed Time		1
+	Equipment Mode Name	NO MODE DATA	
+	Equipment Mode Type	Unknown	
+	Equipment State Name	Disabled	
+	Equipment State Type	Disabled	
+	Execution Time (ms)		21
+	From Time Stamp	2018-01-16 11:19...	
+	Line Downtime Equipment Name		
+	Line Downtime Reason		
+	Line Schedule Count		0
+	Line Standard Count		0
+	Line State Name	Disabled	
+	Line State Type	Disabled	
+	Line Target Count		0
+	OEE		1
+	OEE Availability		1
+	OEE Infeed Count		0
+	OEE Outfeed Count		0
+	OEE Performance		1
+	OEE Quality		1
+	OEE Reject Count		0
+	Operation UUID		
+	Planned Downtime		0
+	Product Code		
+	Rate Period		
+	Runtime		1
+	Schedule Rate		0
+	Shift		
+	Standard Rate		0
+	To Time Stamp	2018-01-16 11:20...	
+	Unplanned Downtime		0
+	Work Order		

Your screen should look the same as shown.

Packing

Line Mode NO MODE DATA	Line State -	Infeed 0	Outfeed 0	Waste 1	Elapsed 1.0 mins	Runtime 1.0 mins	Downtime 0.0 mins	Planned DT 0.0 mins	OEE 100.0%	A 100.0%	P 100.0%	Q 100.0%
----------------------------------	-----------------	-------------	--------------	------------	---------------------	---------------------	----------------------	------------------------	---------------	-------------	-------------	-------------

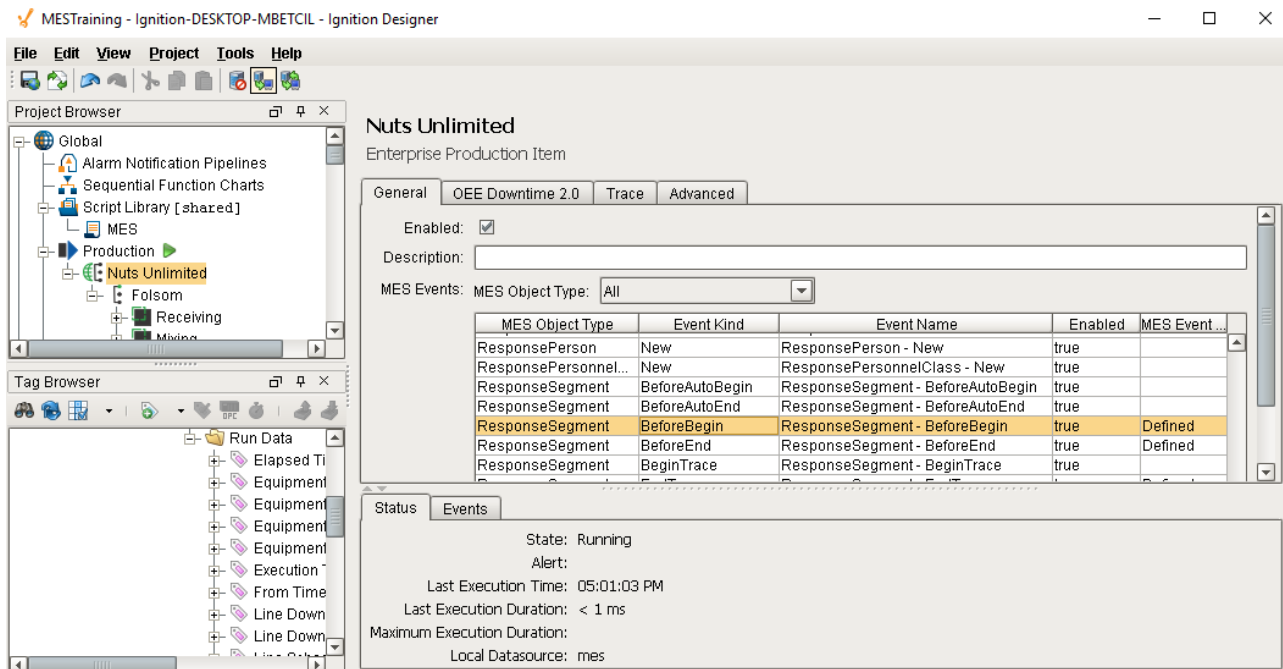
Work Order	Product Code	Std. Rate 0.0 per	Sch. Rate 0.0 per	Std. Count 0	Shift	Operation UUID
------------	--------------	----------------------	----------------------	-----------------	-------	----------------

Packaging Line 1

Mixed Nuts 16oz

Enable the Simulator

Before we test running a packaging operation, we will want to setup our production simulator so that it starts and stops when the packaging line is running. To do this we will write to the simulator Run and Reset tags when the packaging operation starts and stops via the MES Segment events.



MES Events Configuration Panel in the Production Model Designer

- Select the **Nuts Unlimited** item in the production model and choose the **General** tab. Right-click on the **ResponseSegment** object **BeforeBegin** event and select **Edit**.
- Add the following code to the MES Event script:

```
logger = system.util.getLogger('MyOEELogger')
obj = event.getMESObject()
segName = obj.getName()
eqName = obj.getEquipmentLink().getMESObject().getName()
logger.infoof("Response Segment Before Begin event for %s - %s", str(
eqName), str(segName))
if eqName == 'Packaging Line 1' and '_CO' not in (segName):
    logger.info("Starting Simulator")
    system.tag.write('[default]Nuts Unlimited/Folsom/Packaging
/Packaging Line 1/plc_PackagingLine1/Reset', 1)
    system.tag.write('[default]Nuts Unlimited/Folsom/Packaging
/Packaging Line 1/plc_PackagingLine1/Run', 1)
```

- Right-click on the **ResponseSegment** object **BeforeEnd** and select **Edit**.
- Add the following code to the MES Event script:

```
logger = system.util.getLogger('MyOEELogger')
obj = event.getMESObject()
segName = obj.getName()
eqName = obj.getEquipmentLink().getMESObject().getName()
logger.infoof("Response Segment Before End event for %s - %s", str(
eqName), str(segName))
if eqName == 'Packaging Line 1' and '_CO' not in (segName):
    logger.info("Stopping Simulator")
    system.tag.write('[default]Nuts Unlimited/Folsom/Packaging
/Packaging Line 1/plc_PackagingLine1/Reset', 0)
    system.tag.write('[default]Nuts Unlimited/Folsom/Packaging
/Packaging Line 1/plc_PackagingLine1/Run', 0)
```



Check that your tag path is the same. If not, copy and paste your tag paths.

- **Save** and **Publish** your changes.

So what exactly are we doing here? These scripts get executed whenever a response segment starts or ends. Remember, in ISA-95, a [response segment](#) object is created from an operations segment or request segment (scheduled) and is the actual running operation. In the script, we first create a logger, so we can check that these events are firing when we expect them to in the gateway console. We named the logger 'MyOEELogger' so that we can easily filter for just these messages. The event object has functions that allow us to access the response object and from here we can find out what equipment this event is from and what the name of the event is. Here, we are filtering out any operation with '_CO' so that we only start the simulator when the production response segment starts up and not the Changeover response segment. Refer to [MES Object Events](#) for more information.



The scripts we just copied to the MES events are hard-coded to expect the **Packaging Line 1** to be named exactly and the simulator tag paths to be as specified. If you have structured your tags or production items differently, you'll have to adjust accordingly.



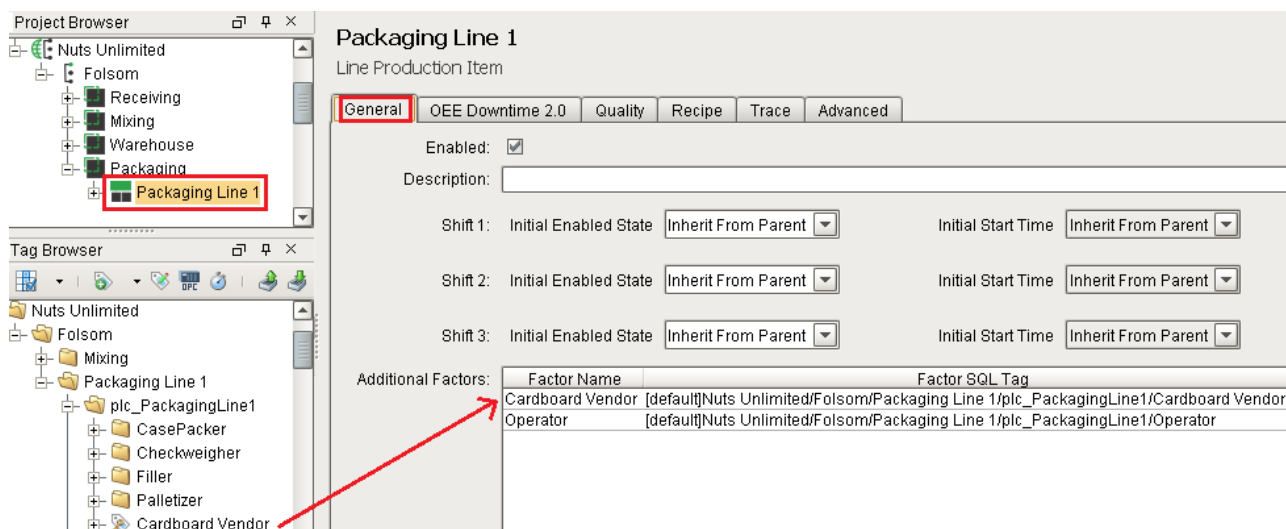
If your simulator does not automatically start when you start a run on the [Test Packaging](#) screen, don't worry, you'll be able to start it manually. We'll detail how to on the [Test Packaging](#) screen.

Adding Additional Factors

Additional factors are a powerful way of extending OEE production analysis capability beyond just counts and equipment status. Rather than just knowing what our Line OEE is by shift, product code or day, maybe we want to know if a certain operator or supplier of raw materials has an impact.

Let's quickly add a couple of additional factors to our Packaging line.


- In the Production model, select **Nuts Unlimited\Folsom\Packaging\Packaging Line 1** and open the General Tab.
- From the Tag Browser **Nuts Unlimited\Folsom\Packaging\Packaging Line 1\plc_PackagingLine1** folder...
 - Drag **Cardboard Vendor** tag onto the Additional Factors pane and change its name to **Cardboard Vendor**.
 - Drag **Operator** tag onto the Additional Factors pane and change its name to **Operator**.





That's it! The value of the Cardboard Vendor and Operator tag will be recorded and associated with the production data captured on Packaging Line 1. When we want to analysis production data, OEE metrics or causes of downtime, who was operating the line and who supplied the cardboard to the case packer cell will be available.

Configuring Shifts

The last thing we will do before we move onto testing our OEE implementation, is to configure the shifts for this line.

- Create a new window called **Shift Manager** under the **Configuration** folder.
- Drag the **Schedule Management** component from the **Admin** component palette on to the window.
- **Save** your changes.
- Go into preview mode (F5) and click .
- Create a shift called **Shift 1** and schedule it for **06:00-18:00** for **All Days**.
- **Save** it and then create another shift called **Shift 2** and schedule it for **18:00-06:00** for **All Days**, then **Save** it.










Schedules > Edit Schedule < back  Save

Name
 

Observe Holidays
☐ False

Description

Schedule

All days	<input checked="" type="checkbox"/>	<input type="text" value="06:00-18:00"/>	
Week days	<input type="checkbox"/>	<input type="text" value="0:00-24:00"/>	
Monday	<input type="checkbox"/>	<input type="text" value="0:00-24:00"/>	
Tuesday	<input type="checkbox"/>	<input type="text" value="0:00-24:00"/>	
Wednesday	<input type="checkbox"/>	<input type="text" value="0:00-24:00"/>	
Thursday	<input type="checkbox"/>	<input type="text" value="0:00-24:00"/>	
Friday	<input type="checkbox"/>	<input type="text" value="0:00-24:00"/>	
Saturday	<input type="checkbox"/>	<input type="text" value="0:00-24:00"/>	
Sunday	<input type="checkbox"/>	<input type="text" value="0:00-24:00"/>	

Repetition

Repeat / Alternate

Days/Weeks On

Days/Weeks Off

Starting At


Preview

< Previous Week of May 1, 2017 Next >

2017	Sunday, April 30	Monday, May 1	Tuesday, May 2	Wednesday, May 3	Thursday, May 4	Friday, May 5	Saturday, May 6
1 AM							
2 AM							
3 AM							
4 AM							
5 AM							
6 AM	6:00 AM - 6:00 PM	6:00 AM - 6:00 PM	6:00 AM - 6:00 PM	6:00 AM - 6:00 PM	6:00 AM - 6:00 PM	6:00 AM - 6:00 PM	6:00 AM - 6:00 PM

FastTrak

Want to FastTrak your training?

- Download [Shift Manager.proj](#) and import it into your project.
- Open the window called **Equipment Manager** under the **Configuration** folder.
- Select **Packaging Line 1** and select [change](#)  under **Equipment Schedule**.
- Enable **Shift 1** and **Shift 2** for this line.

Equipment List

- Nuts Unlimited
 - Folsom
 - Receiving
 - Nut Storage Silos
 - Nut Unloading
 - Mixing
 - Mixing Line 1
 - Warehouse
 - Finished Goods
 - Packaging
 - Packaging Line 1

Equipment Mode Class
Current Selection: Default [change](#)

Equipment State Class
Current Selection: Default [change](#)

Equipment Schedule
Current Selection: Shift 1, Shift 2 [change](#)

Whenever the shift changes as defined in the Schedule Management component, the name of the current shift will be recorded along with any production data for the packaging line and will be available to us through analysis. For more information, refer to [Shift Configuration](#) in the Help Manual.



⚠ It may take a few minutes for the newly configured shift values to show up in the MES Live Analysis tags. You can speed this up by disabling and re-enabling the production model. Adding Shifts to the production line automatically creates new MES tags.

- Tags
 - System
 - Client
 - All Providers
 - default
 - MES
 - Nuts Unlimited
 - Folsom
 - Packaging
 - Packaging Line 1
 - Live Analysis
 - Run Data
 - Shift
 - Current Shift
 - Production Day Begin Date
 - Shift Begin Date


| | Shift 1 |
|-----------------------|---------|
| 2017-10-03 6:00:00 AM | |
| 2017-10-03 6:00:00 AM | |

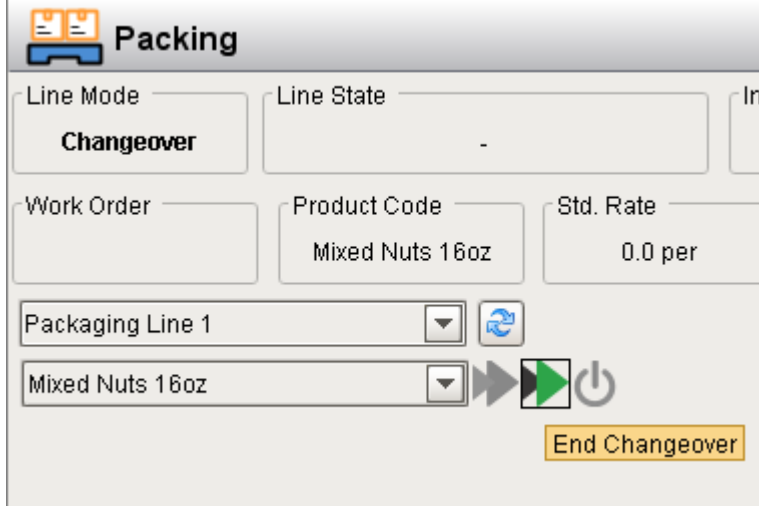
4.3.2 Test Packaging

Now we should be ready to actually run a packaging operation and create some mixed nut products.

- Make sure your Gateway is in read/write mode by selecting the  button.
- Open the **Packing** screen and select **Packaging Line 1** in the MES Object Selector.
- Select one of your Mixed Nuts products in the Run Director and click the  button.

If everything is setup correctly, the Run Director will have started your Mixed Nuts Operations Definition and the Changeover Operation Segment. The changeover segment will run for the default 60 seconds as defined in the [Material Production Settings](#) pane of the [OEE Material Manager](#), then the production operation segment will start up.

You can manually advance the packaging operation out of changeover into production by selecting the middle  button.



You should start seeing some production data showing mode, line state, product code, production counts and OEE metrics. Give it a few minutes for material to make its way through the packaging line at which point you should start to see production counts for Outfeed and Waste.

| Line Mode | Line State | Infeed | Outfeed | Waste | Elapsed | Runtime | Downtime | Planned DT | OEE | A | P | Q |
|------------|-------------------|---------------|---------------|------------|-----------|--------------------------------------|----------|------------|-------|-------|-------|-------|
| Production | Running (default) | 1394 | 104 | 216 | 21.2 mins | 16.2 mins | 2.0 mins | 3.0 mins | 43.2% | 88.9% | 57.4% | 84.5% |
| Work Order | Product Code | Std. Rate | Sch. Rate | Std. Count | Shift | Operation UUID | | | | | | |
| | Mixed Nuts 16oz | 150.0 per Min | 100.0 per Min | 3,180 | Shift 1 | d134d60f-d743-4e58-9e25-8c55fbcaf9e7 | | | | | | |

Downtime Table Editor

Let's add an [OEE Downtime Table](#) component to this window, so an operator can view and update line downtime events as they happen.

- Drag the OEE Downtime Table from the OEE Downtime 2.0 component palette onto the screen.
- Bind the equipment path property of the OEE Downtime Table to the *Root Container*. *MES Object Selector.equipmentItemPath*.
- Set the **Run Look Back Count** property of the OEE Downtime Table to 1.

| Begin | End | Cell | Reason | Duration | Note | Split | Occurrence ... |
|---------|---------|--------------|------------------------|----------|------|--------------------------|----------------|
| 4:56 PM | 4:56 PM | Filler | E-Stop Pulled | 0.02 | | <input type="checkbox"/> | 1 |
| 4:55 PM | 4:55 PM | Filler | Out of Infeed Material | 0.02 | | <input type="checkbox"/> | 1 |
| 4:48 PM | 4:49 PM | Palletizer | CIP | 1 | | <input type="checkbox"/> | 1 |
| 4:45 PM | 4:46 PM | Casepacker | Case Feeder Faulted | 1 | | <input type="checkbox"/> | 1 |
| 4:42 PM | 4:43 PM | Filler | CIP | 1 | | <input type="checkbox"/> | 1 |
| 4:37 PM | 4:39 PM | Checkweigher | Weigh Checksum Error | 2 | | <input type="checkbox"/> | 1 |
| 4:33 PM | 4:34 PM | Filler | Out of Infeed Material | 1 | | <input type="checkbox"/> | 1 |
| 4:30 PM | 4:31 PM | Casepacker | Case Jam | 1 | | <input type="checkbox"/> | 1 |

The OEE Downtime Table will display all line downtime events for a given time period as defined by the **Start Date** and **End Date** properties when *Run Look Back Count* is set to 0. By default *Run Look Back Count* is set to 1 which ignores the Start and End date parameters and only displays downtime for the current run. Play around with the [OEE Downtime Table](#) component. You can click on an event which will bring up a panel that allows you to change the downtime reason, split a downtime event into multiple events and add notes.

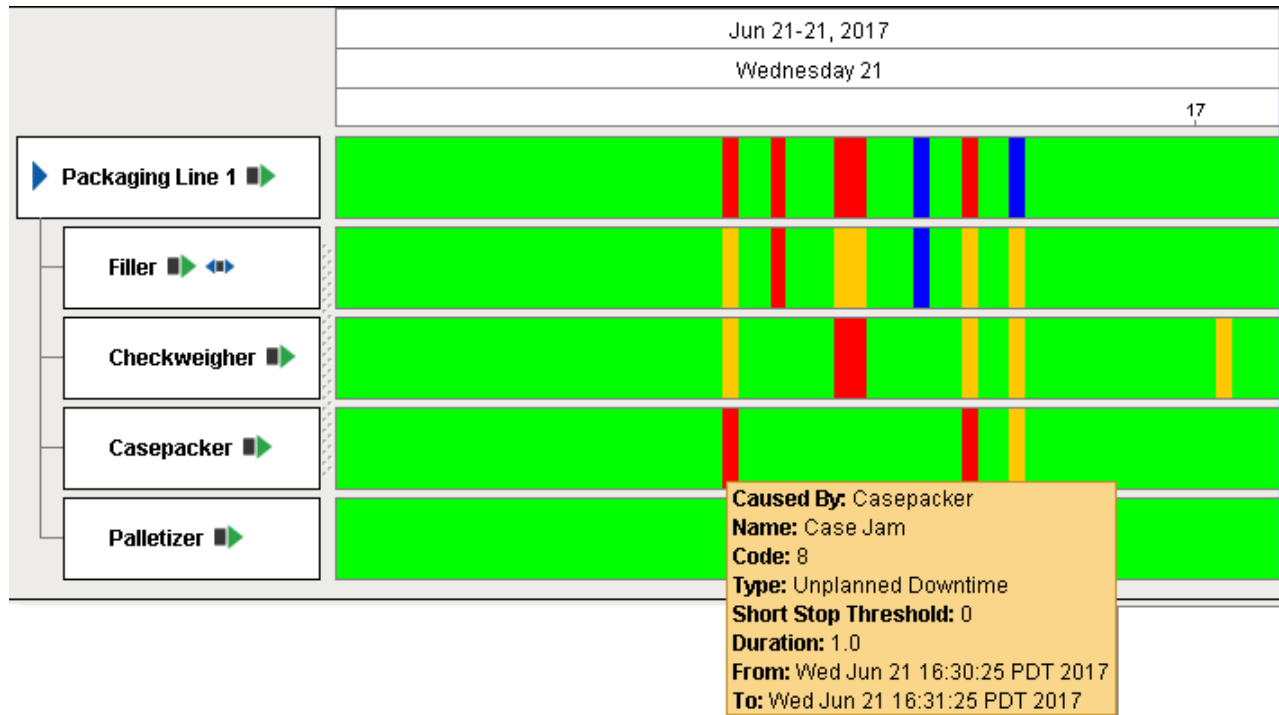
OEE Time Chart


We'll throw an [OEE Time Chart](#) component onto the screen underneath the [OEE Downtime Table](#). This will provide a visual of the states and modes of the cells underneath the line.

- Drag the OEE Time Chart from the OEE Downtime 2.0 component palette onto the screen.
- Bind the *linepath* property of the OEE Time Chart component to the *Root Container*. *MES Object Selector.equipmentItemPath*.
- Set the **Run Look Back Count** property of the OEE Downtime Table to 1.

142

In the runtime (F5), you can expand the view to show or minimize the cells under the line. Hovering over the state changes will provide a tooltip that details each state. There are plenty of configuration options available with this component and also extension functions that allow you to add a custom data field to the right where you can display product code or work order or anything you desire at the line and the cell level. Refer to the [OEE Time Chart](#) help for more information.



- End the Production run by clicking on the OEE Run Director  button.

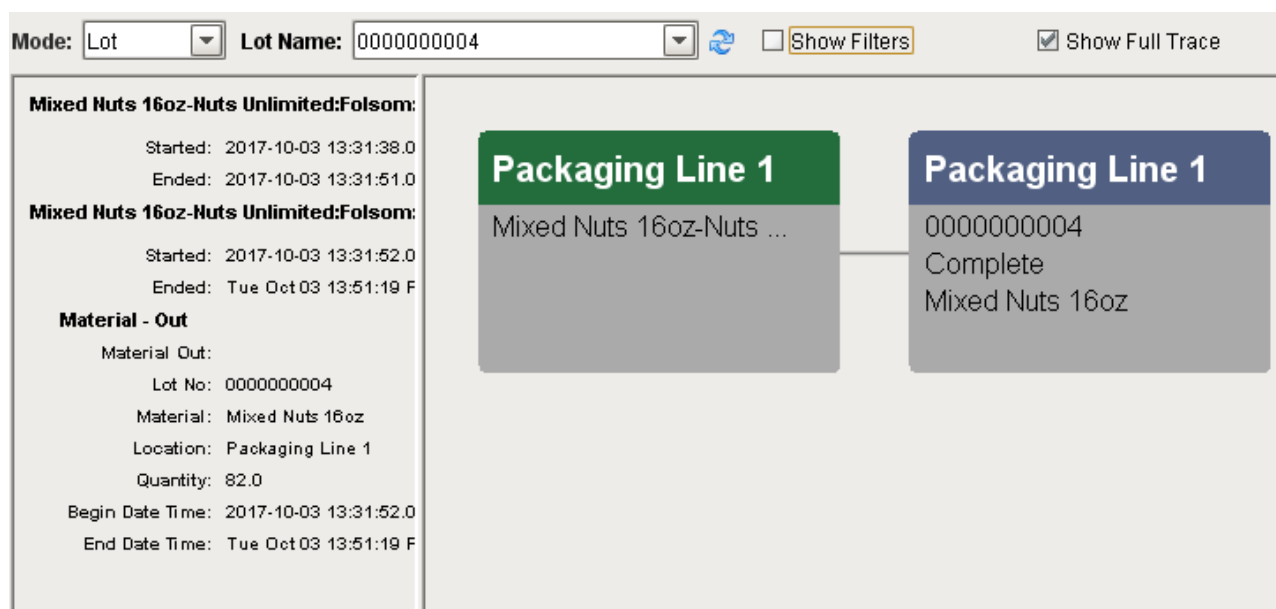
Tracing the Operation

Let's take a look at what was produced using the [MES Trace Graph](#). You might need to change the date range to see the lot that was just created. Select the last lot number used to package nuts to see the results. We should see a simple trace showing the packaging process and the resulting lot that was created. Each box on the trace graph is called a node. The Package Nuts node (green) represents the segment (production task) that was done. The blue node represents the finished goods material lot.



- The colors of the different node types are configurable and can be modified by changing the Node configuration dataset of the Trace graph object. See [MES Trace Graph Help](#) for more details.
- Each node can be configured to add custom user menu items. You can use this to extend the trace graph functionality to bring back any data associated with that lot or process that was not captured as part of Track & Trace. An example would be to bring back OEE or SPC data or tag process history data. For more information on creating custom user menu items, see the `UserMenuItemClicked` eventHandler in the [MES Trace Graph Help](#) for more details.

So what's missing from this trace? Well we don't know which lots of mixed nuts from the mixing operation were used in the packaging operation. If we are only implementing OEE to schedule production runs, track production numbers and provide OEE metrics, then we don't need to associate incoming lots to the packaging operation. But if we want to provide lot tracking and consume available lot quantities, then we need to let the packaging operation know which material lots are being consumed. In the next section, we'll extend the OEE implementation to add lot tracking.



The Trace Graph component comes with the Track & Trace module and will not be available if you are only using the OEE module.

4.3.3 Add Lot Tracking



This section of the tutorial uses components and scripting functions only available to the Track & Trace module to show how to add lot tracking to the current OEE implementation. If you do not want to extend your OEE implementation, you can bypass the next two sections by clicking [Scheduling Operations](#)

If we were just doing an OEE project, the configuration we have done so far would be sufficient to schedule production runs, control runs and capture OEE metrics and production data. But as part of this tutorial, we are also implementing lot tracking using the Track & Trace Module and we'll want to capture which lots of mix nuts from the mobile storage bins are consumed by the packaging operation to create the Mixed Nuts 8oz and 16oz products.

Fortunately, the material objects used by the OEE module are the same as those used by the Track & Trace module, so we can extend the OEE implementation fairly easily. The following steps will show how this is achieved.

Update Packaging Operation

The Material-Operations created by the [OEE Material Manager](#) component will now be modified to allow us to track incoming lots and we'll also add a reject lot to the output of the OEE operation.

In the last section we used the OEE Material Manager to create a material class **Finished Goods** and two material definitions, **Mixed Nuts 8oz** and **Mixed Nuts 16oz**. This class and material definitions can be seen using the MES Object Editor component, and if we wanted to add custom properties, we could do so here. We could also create process segments that used these materials as input or output material references.

When we enabled Packaging Line 1 for these products in the OEE Material Manager, in the background, the following operations definition and segments were automatically created for us.



- **Mixed Nuts 16oz-Nuts Unlimited:Folsom:Packaging:Packaging line 1**
- **Mixed Nuts 8oz-Nuts Unlimited:Folsom:Packaging:Packaging line 1**

The first segment under the Operations Definition is for the changeover operation, the second is for the production segment. Again if we wanted to add custom properties to associate data with these segments, we can do so here.

For our tutorial, we are going to add material references for the mixed nuts in the storage bins and rejected nuts to our production operation segment. We are going to modify the production operation segment rather than the OEE Production process segment, because the material references will be specific to the Mixed Nuts operation.

Modify the Mixed Nuts 8oz Packaging Operations Segment

We will just update the Mixed Nuts 8oz operation, so we can compare the standard 16oz operation to the lot tracking enabled 8oz operation.

- In the MES Object Editor screen, select  **Segments & Operations**.
- Right-click on the right hand operations segment for **Mixed Nuts 8oz-Nuts Unlimited: Folsom:Packaging:Packaging line 1** and select **Edit Settings**.
- Add a new material called **Mixed Nuts In** by clicking  and enter the following values:

| Property | Value | Comment |
|--------------------------------|---------------------------------|--|
| Name | Mixed Nuts In | Name we can use to access this from scripting |
| Use | In | Input material to this operation segment |
| Optional | True | |
| Material Reference | Material Definition, Mixed Nuts | Restrict the input material to Mixed Nuts |
| Lot Equipment Reference | Equipment Class, Mixed Nut Bins | Only accept material from the Mixed Nut Bins |
| Lot Number Source | Manual | We will supply the lot number of this material |
| Quantity Source | Manual | We will supply the quantity consumed |
| Units | lbs | |
| Rate Period | None | |

- Add a new material called **Rejected Mixed Nuts** by clicking  and enter the following values:

| Property | Value | Comment |
|--------------------------------|---------------------------------|---|
| Name | Rejected Mixed Nuts | Name we can use to access this from scripting |
| Use | Out | Input material to this operation segment |
| Optional | True | |
| Auto Generate Lot | True | |
| Material Reference | Material Definition, Mixed Nuts | Force waste material to the Rejected Mixed Nuts material definition |
| Lot Equipment Reference | Equipment, Reject Bin | Waste material will be stored at the line |
| Lot Number Source | Auto | Use the lot number of this material |
| Quantity Source | Manual | We will supply the quantity rejected |
| Units | lbs | |
| Rate Period | None | |

We also need to prevent the production operation segment from automatically starting when changeover is ended. The reason for this is that we do not want the Track & Trace module to auto assign material lots in to our production operation, so we need to create this response segment with **autoassignoptions** set to false through scripting.

- Under **Trigger Segment Begin** un-check the **Auto** checkbox
- **Save** the changes to the operations segment.

Operations Segment, Mixed Nuts 8oz-Nuts Unlimited:Folsom:Packaging:Packaging Line 1 ✓


| | | |
|---|-------------------------------|--|
| + | Station | Line, Packaging Line 1 |
| - | Personnel | |
| | none defined | |
| - | Supplemental Equipment | |
| | none defined | |
| - | Production Settings | |
| + | Packaging Line 1 | Line, Packaging Line 1 |
| + | Packaging Line 1:Casepacker | Line Cell, Casepacker |
| + | Packaging Line 1:Checkweigher | Line Cell, Checkweigher |
| + | Packaging Line 1:Filler | Line Cell, Filler |
| + | Packaging Line 1:Palletizer | Line Cell, Palletizer |
| - | Trigger Segment Begin | |
| - | DefaultBeginTrigger | Operations Segment, Mixed Nuts 8oz-Nuts Unlimited:Folsom:Packaging:Packaging Line 1_CO |
| | Name | DefaultBeginTrigger |
| | Mode | At Segment End |
| | Auto | <input type="checkbox"/> |

The last thing we need to do is to stop the Changeover segment from automatically ending after 60 seconds as we will be ending the changeover segment through scripting.

- Open up the **Material Manager** screen under Configuration and press **F5** to go into preview mode.
- Select the **Mixed Nuts 8oz** product and **Packaging Line 1**.
- Uncheck the checkbox for **Auto End Changover**.

Update Packaging Screen

Because we are now moving into the realm of this being a Track & Trace with OEE application, we need to replace some of the standard OEE components such as the Run Director. The Run Director simplifies the process of starting OEE operations but to be able to add material lots being consumed and created, we need to switch to standard Ignition components and system.mes scripting functions.

- **End** any OEE runs you have currently running.
-  Download and import **Packing - Lot Tracking.proj** into your project.

This screen is pre-built for you. Scripts in the actionPerformed event of the buttons will show you how to use scripting to start the OEE operation and end the Changeover segment. There is a trigger property that will automatically start the production segment when the changeover segment is ended. Once you are in production, you'll be able to select infeed lots and create waste lots for the operation.

Operational Control

Select Line: Packaging Line 1 Start CO End OperationsResponse Mixed Nuts 8oz-Nuts Unlimited:Folsom:Packagi...

Select Product: Mixed Nuts 8oz End CO Get Active Response Segment Mixed Nuts 8oz-Nuts Unlimited:Folsom:Packagi...

Infeed Material

Select Material(s) to Add

| LotNumber | MaterialName | NetQuantity | Units | Location Name |
|------------|--------------|-------------|-------|---------------|
| 0000000051 | Mixed Nuts | 500 lbs | | Bin 0001 |
| 0000000060 | Mixed Nuts | 499 lbs | | Bin 0002 |

Enter Quantity: 100 Add Material(s) Scrap Nuts

Rejected Goods

Select Material Out: Mixed Nuts Select Location Reject Bin Enter Quantity 10 Create Waste Lot

Finished Goods

Lot #: 0000000058 Add Sub-Lot SL003 View Sub-Lots

Downtime Run Chart Tracer

Diagram:

Details:

0000000059 Include Seg Info

Mixed Nuts 8oz-Nuts Unlimited:Folsom:Packaging:Packa...

Started: Fri Feb 02 12:39:09 PST 2018
Ended: Fri Feb 02 12:39:16 PST 2018

Mixed Nuts 8oz-Nuts Unlimited:Folsom:Packaging:Packa...

Started: Fri Feb 02 12:39:17 PST 2018
Ended: -

Material - In

Mixed Nuts In: (unused)
Mixed Nuts In-1:
Lot No: 0000000051
Material: Mixed Nuts
Location: Bin 0001
Quantity: 49.0
Begin Date Time: Fri Feb 02 12:40:29 PST 2018
End Date Time: Fri Feb 02 12:45:33 PST 2018
Mixed Nuts In-2: Active
Lot No: 0000000060
Material: Mixed Nuts

Selected Node Type - ResponseSegment 4a4cfd7-4fb8-422a-8017-597bc92581a4

- Add the following script to the shared.MES global scripts:

```
def getOperationName(eqPath):

    # Given an equipmentPath , this function will return the
    operation Name
    # equipPath comes in the format of [global]
    \Enterprise\Site\Area\Line
    # which needs to be converted to -Enterprise:Site:Area:Line

    print "getOperationName called for ", eqPath

    tagPath = eqPath
    if tagPath is not None:
        tagPath = tagPath.replace('\\', ':')
        if '[global]:' in (tagPath):
            tagPath = tagPath[len('[global]:'):]    #Remove
    [global]
        tagPath = '-' + tagPath
    return tagPath
```

This script is called by the **Start CO** button to build the operation definition name from the selected Line and material definition.

Set Finished Goods Lot Status to Available

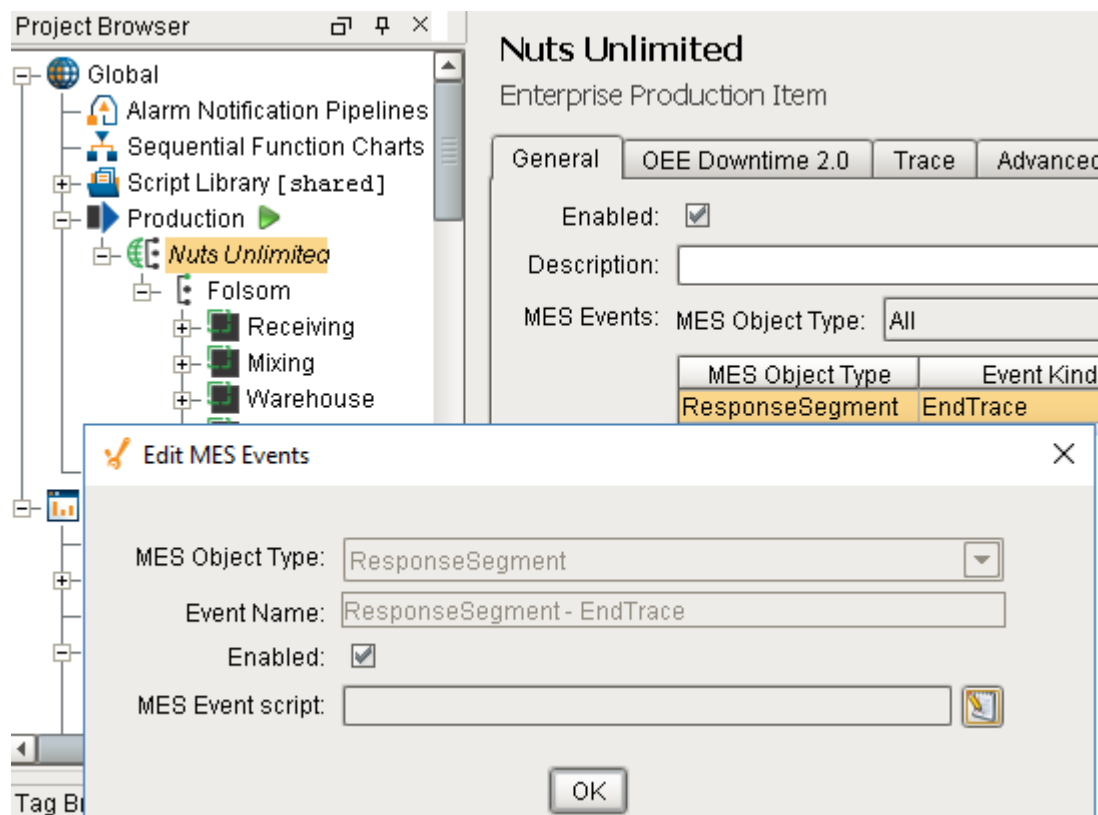
By default, any material lots produced by an OEE operation have their lot status set to **Used** because an OEE implementation by itself does not require lot tracking. as we are combining Lot tracking with OEE, we need to make sure that finished goods coming off the line are available to be consumed by any other operations we create such as **Ship Goods** or **Scrap Material** or **Rework**.

To do this, we will put some script into the MES Response Segment EndTrace event that will update the lot status to Available.

- Select the Enterprise production item **Nuts Unlimited** in the production model.
- On the General tab, right-click on the **Response Segment- End Trace** event.
- Add the following code to the MES Event script.

```
#When OEE 2.0 creates an outfeed object, its lot status is set to
USED.
#We need to set it to Available if we want to display it in the
Trace graph,
#use it in other segments or show it using the trace inventory
functions

logger = system.util.getLogger('MyOEELogger')
obj = event.getMESObject()
segName = obj.getName()
eqName = obj.getEquipmentLink().getMESObject().getName()
logger.infoof("Response Segment End Trace event fired created for %
s - %s", str(eqName), str(segName))
if eqName == 'Packaging Line 1' and ':' in (segName) and '_CO' not
in (segName):
    mat = obj.getMaterialLot('Material Out')
    logger.infoof("Setting lot status to Available for %s", str(mat
.getName()))
    mat.setPropertyValue('LotAvailabilityStatus', 'Available')
    system.mes.saveMESObject(mat)
```



Test Lot Tracking

We will now run the updated **Mixed Nuts 8oz** operation to see our lot tracking in action

- Open the **Packing With Lot Tracking** screen and press **F5** to go into preview mode.
- Select **Packaging Line 1**.
- Select the **Mixed Nuts 8oz** product and press the **Start CO** button to start the Changeover segment.
- Press the **End CO** button to end the changeover and start the production segment.

In the operation definition, the operation segments are sequential requiring the changeover operation to be run before the actual production operation can be run. Ending the changeover segment would normally automatically begin the production segment, but we disabled the trigger so that we can control how the production segment is created. The **End CO** button has script that will end the CO segment and then create and begin the Production segment.

152

The simulator will start to feed material into the packaging line. After a while finished goods will start to come out, along with some waste. We could have possibly tied the **Mixed Nuts In** quantity source to the Infeed count MES Counter and the **Reject Nuts Out** quantity source to the Waste Count MES counter, but we don't know how many nuts from each lot is being consumed, so we will handle that through scripting

- Select an infeed material lot from the power table, enter a quantity and press **Add Material**. If your table is empty, you'll need to go back and mix some more nuts.
- Go to **Rejected Goods**, select **Reject Bin**, enter a quantity and press **Create Waste Lot**.
- Press the **End** button when you are done with this packaging operation.

Operational Control
Select Line Packaging Line 1
Select Product Mixed Nuts 8oz

Start CO End
End CO Get Active

OperationsResponse Mixed Nuts 8oz-Nuts Unlimited:Folsom:Packagi...
Response Segment Mixed Nuts 8oz-Nuts Unlimited:Folsom:Packagi...

Infeed Material
Select Material(s) to Add

| LotNumber | MaterialName | NetQuantity | Units | Location Name |
|------------|--------------|-------------|-------|---------------|
| 0000000051 | Mixed Nuts | 500 lbs | | Bin 0001 |
| 0000000060 | Mixed Nuts | 499 lbs | | Bin 0002 |

Enter Quantity 100 Add Material(s) Scrap Nuts

Rejected Goods
Select Material Out Mixed Nuts Select Location Reject Bin Enter Quantity 10 Create Waste Lot

Finished Goods
Lot # 0000000058 Add Sub-Lot SL003 View Sub-Lots

Downtime Run Chart Tracer

Bin 0001

0000000051

Complete

Mixed Nuts

1

3

Bin 0002

0000000060

Active

Mixed Nuts

1

Packaging Line 1

Mixed Nuts 8oz-Nuts U...

Reject Bin

0000000059

Active

Mixed Nuts

Packaging Line 1

0000000058

Active

Mixed Nuts 8oz

0000000059

Include Seg Info

Mixed Nuts 8oz-Nuts Unlimited:Folsom:Packaging:Packa...

Started: Fri Feb 02 12:39:09 PST 2018

Ended: Fri Feb 02 12:39:16 PST 2018

Mixed Nuts 8oz-Nuts Unlimited:Folsom:Packaging:Packa...

Started: Fri Feb 02 12:39:17 PST 2018

Ended: -

Material - In

Mixed Nuts In: (unused)

Mixed Nuts In-1:

Lot No: 0000000051

Material: Mixed Nuts

Location: Bin 0001

Quantity: 49.0

Begin Date Time: Fri Feb 02 12:40:29 PST 2018

End Date Time: Fri Feb 02 12:45:33 PST 2018

Mixed Nuts In-2: Active

Lot No: 0000000060

Material: Mixed Nuts

Selected Node Type - ResponseSegment 4a4cfd7-4fb8-422a-8017-597bc92581a4

There is a box on this screen for adding sub-lots that is currently disabled. You can ignore this for now as we'll using it in the next section.

Trace the Material

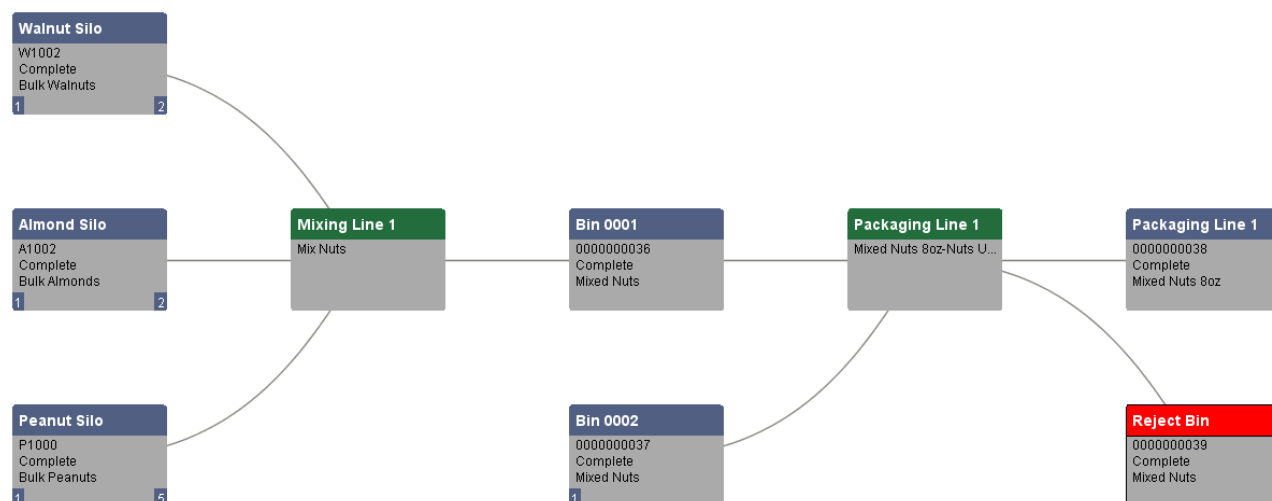
Our tracer template allows us to see the material lots consumed and created as part of this OEE operation, but let's take a look at the entire material flow in the Trace Graph window under Analysis.

- Open the **Trace Graph** window under Analysis and select the newly created lot. In our example, it is lot 0000000039.

We can see that two lots of Mixed Nuts were consumed to create the Mixed Nuts 8oz product and that we also created a waste lot that is in the Reject Bin.


- Select one of the infeed Mixed Nuts lots to see the entire material flow.

We can see the entire process, from the receiving of the Bulk nuts into the silos, to the mixing operation that created the Mixed Nuts, to the packaging operation that created the finished products.



At this point we have completed the lot tracking with OEE. We could go onto create a process segment that allows us to consume the finished goods. In this case, our process segment would simply have a Material In property and no Material Out property.

4.3.4 Add Sub Lot Tracking

 This section of the tutorial uses components and scripting functions only available to the Track & Trace module to show how to add lot tracking to the current OEE implementation. If you do not want to extend your OEE implementation, you can bypass this section by clicking [Scheduling Operations](#).

If we were just doing an OEE project, the configuration we have done so far would be sufficient to schedule production runs, control runs and capture OEE metrics and production data. But as part of this online tutorial, we are also implementing sub lot tracking using the Track & Trace Module. Sub lots allow us to create serialized lots of finished goods that will be associated with the finished good material lot.

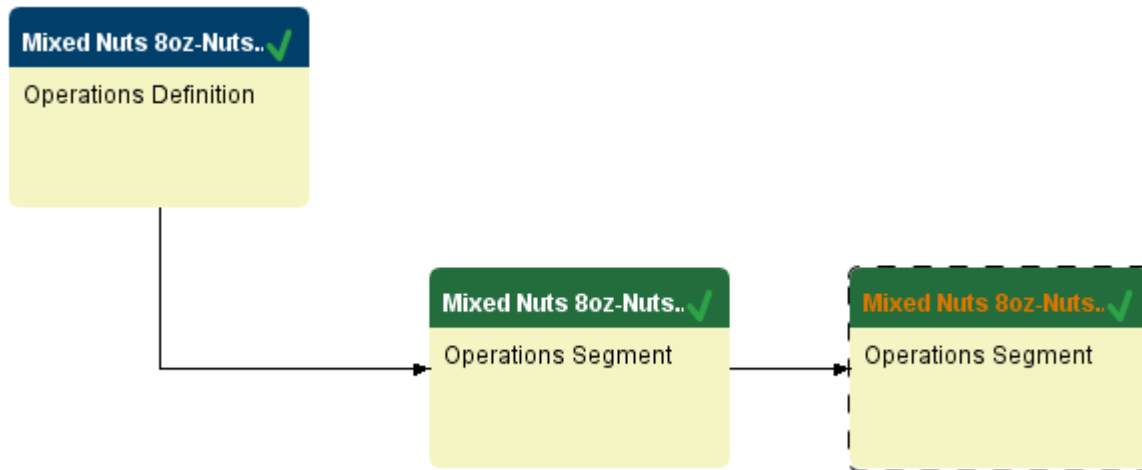
What Are Sub Lots?

As material lots are created on the packaging line, we simply update the count of how many items have been produced as part of the finished goods Mixed Nuts 8oz material lot. But if each item is serialized, then we would probably want to create an individual material lot for that item storing the serial number. In Track & Trace, we could simply execute the process segment each time a serialized lot is produced, but if 100's of parts are produced or a new part is created every second, the overhead with executing an operation segment each second and the ability to view 100's of material lots on the trace graph, would make this challenging. Sub lots allow us to create a serialized sub lot for each part that is a child of the finished goods Mixed Nuts 8oz material lot.

The first thing we need to do is let the operations segment for the OEE run know that sub lots are generated as part of this operation. We will do that in the MES Object Editor.

- End the current run.
- Open the **MES Object Editor** screen under **Admin**.
- Right-click on the right hand operations segment for Mixed Nuts 8oz-Nuts Unlimited: Folsom:Packaging:Packaging Line 1 operations definition and select **Edit**.
- Select the **Material Out** complex property and check the **Enable sublots** setting.

Our operations segment created by the OEE Material Manager is now configured to support sub lots for the Material Out.



Operations Segment, Mixed Nuts 8oz-Nuts Unlimited:Folsom:Packaging:Packaging Line 1 ✓ Save

Core Properties

| | |
|-----------------------------|---|
| Name | Mixed Nuts 8oz-Nuts Unlimited:Folsom:Packaging:Packaging Line 1 |
| Description | |
| End Operation When Complete | <input checked="" type="checkbox"/> |
| Segment Recipe Name | |


Custom Properties


| | |
|--------------|--|
| none defined | |
|--------------|--|

Material



Material Out



| | |
|-------------------------|--|
| Name | Out, Material Definition, Mixed Nuts 8oz |
| Optional | <input type="checkbox"/> |
| Production Selectable | <input type="checkbox"/> |
| Use | Out |
| Auto Generate Lot | <input checked="" type="checkbox"/> |
| Material Reference | Material Definition, Mixed Nuts 8oz |
| Lot Equipment Reference | Line, Packaging Line 1 |
| Enable Sublots | <input checked="" type="checkbox"/> |

- Open the **Packing with Lot Tracking** window under **Packaging**.
- Enable the **enableSublots** property on the *cntMaterial/Out* container.
- **Start** a new run for **Mixed Nuts 8oz** and then add sub-lots to the Finished Goods lot using the  button.

You can click the  **View Sub-Lots** to view the sub-lots that have been added to the finished goods lot.

Finished Goods

Lot # Add Sub-Lot   **View Sub-Lots**

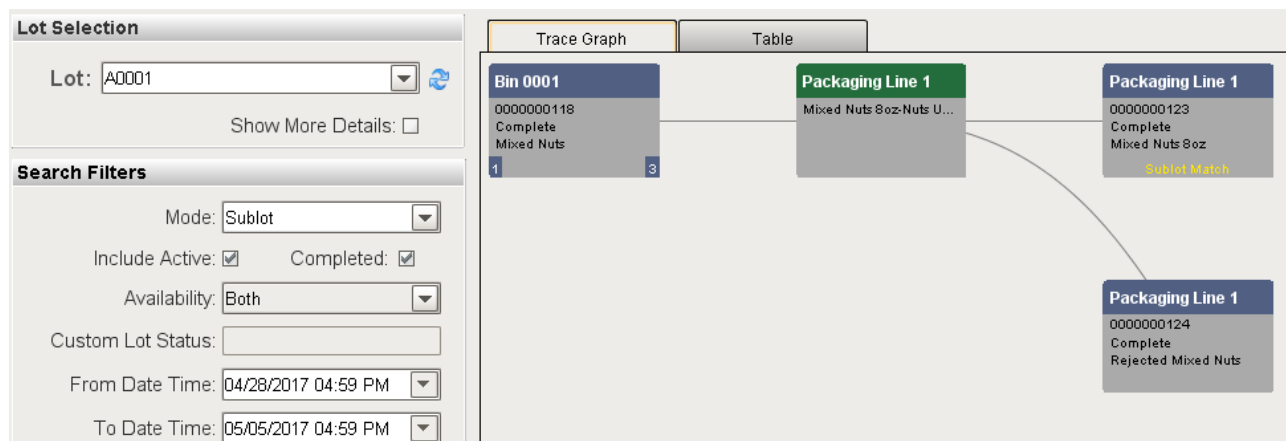
The script behind the  and  **View Sub-Lots** buttons shows how sub-lots can be added and retrieved.

Let's take a look on the trace graph to see our sublots.

- Open the **Trace Graph** window under **Analysis**.
- Change the Search Filter **Mode** to **Sublot** and set **Include Active Lots** to true.
- Enter the name of one of your sublots into the **Lot Selection** filter.

You may need to adjust the **To Date Time** filter to the future to see your sub lot.



The text **Sublot Match** will appear on the Mixed Nuts 8oz lot and we can see the trace of operations and raw material lots that made up this sublot.










4.3.5 Scheduling Operations

So far we have started unscheduled packaging operations using the OEE Run Director and MES Segment Selector components. In this section we will create some work orders using the [MES Work Order Table](#) component and use the scheduling components to schedule production runs.




Create a Work Order

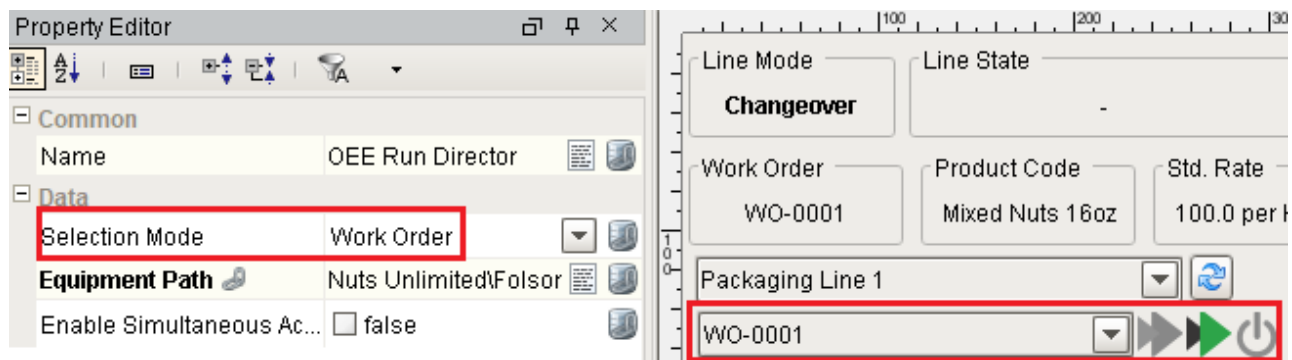
- **End** the current run.
- Create a folder called **Scheduling** and add a new main window called **Scheduler**.
- Drag the [MES Work Order Table](#) component from the Production component palette onto the window.
- Go into preview mode by pressing **F5** or clicking on the  button on the menu bar.
- Click the  Add button on the Work Order Table and create a work order for 100,000 units of the Mixed Nuts 16oz product. Set the due date to sometime this week.
- **Save** your changes.

We now have a work order that we can schedule using the MES Schedule View component.

| MES Work Order Table | | | | | | | | | |
|----------------------|-----------------|---------------------|------------|-----------|-------------|---------------|--------------------------|---|--------|
| Work Order | Material | Work Order Quantity | Schedul... | Actual... | Remainin... | Due Date | Closed |  | Add |
| WO-0001 | Mixed Nuts 16oz | 100,000 | 0 | 0 | 100,000 | Feb 3, 201... | <input type="checkbox"/> |  | Edit |
| | | | | | | | |  | Delete |
| | | | | | | | |  | Copy |
| | | | | | | | |  | Paste |
| | | | | | | | |  | Import |
| | | | | | | | |  | Export |

Start an unscheduled Work Order Run

- On the original Packing screen that you created with the OEE Run Director component, change the **Selection Mode** property of the OEE Run Director to **Work Order**.
- Go into preview mode by pressing **F5** or clicking on the  button on the menu bar.
- Select **Packaging Line 1** in the MES Object Selector and select the work order you created for this line.
- Start the operation using the  button.
- When you're done, end the work order run using the  button.

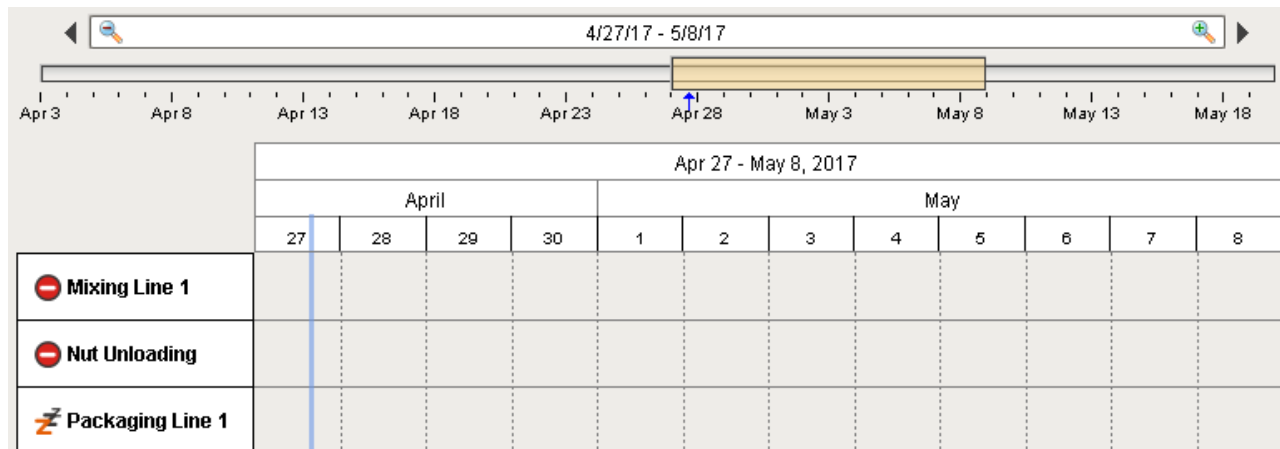



The OEE Run Director component allows us to start unscheduled production runs and un-scheduled work order runs. Now we will look at how we can schedule work order runs and production runs that don't have a work order.

Schedule Work Order Production Runs

- Open the Scheduler window you created earlier.
- Drag an MES Object Selector component onto the window and check the **Include MES Line Objects**.
- Drag the MES Schedule View component from the Production component palette onto the window and bind the **Equipment Path** property to *MES Object Selector.equipmentItemPath* property.
- Drag an Ignition Date Range component onto the window and bind the start and end date properties of the MES Schedule view component to the date range start and end date properties.

There are a couple of ways we can schedule work orders now. We could drag the work order from the work order table and drop it on the packaging line, or we could right-click on the Packaging line 1 in the MES Schedule View component and select **New Entry**.

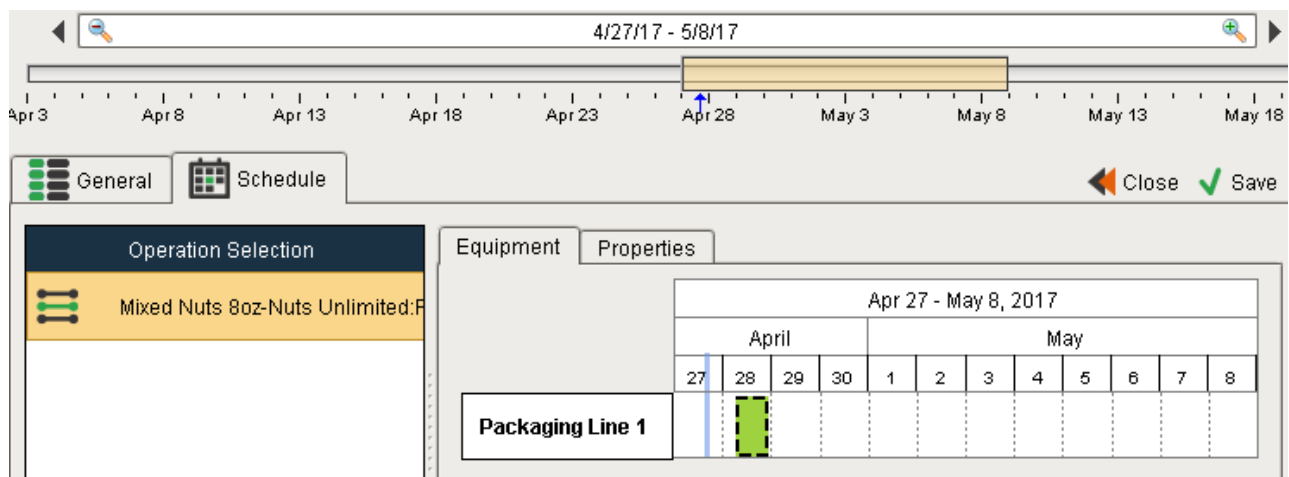


- Go into preview mode by pressing **F5** or clicking on the  button on the menu bar.
- Drag the work order we created from the work order table and drop it onto the Packaging Line 1.

*If this doesn't work, make sure you have the **Row Dragging Enabled** property of the MES Work Table set to True*

On the General Tab, you can adjust the production count or set the run to be based on a duration rather than a count. On the Schedule Tab, you can adjust when the run will start and end.

- **Save** your changes.




Schedule Non Work Order Production Runs



If you go back and take a look at the General Tab again, you'll see that you could select *None* for Work Order, and then select an operation. This allows you to schedule production runs without having to create a work order. We could also create process segments for **Maintenance** or **Cleaning** or any operation and schedule those activities using this component.

The following knowledge base article [Creating Maintenance Operations](#) shows how to create and schedule maintenance operations. For more information on Scheduling, refer to the [Detailed Production Scheduling](#) help.

Starting Scheduled Runs

- Drag an **MES Schedule Selector** component from the Production component palette onto the Scheduler window.
- Bind the **Equipment Path** property to *MES Object Selector.equipmentItemPathFilter* property.
- Bind the **Start** and **End** date properties of the MES Schedule view component to the date range start and end date properties.
- Go into preview mode by pressing **F5** or clicking on the  button on the menu bar.
- Right-click on the Scheduled Work Order run and select **Begin OEE Run**.
- When you are done, go ahead and end that run by right clicking on it and selecting **End OEE Production**.

| Description | ScheduledBegin | ScheduledEnd | ActualBegin | ActualEnd | State | PercentC... |
|-------------------|----------------------|-----------------------|-------------|-----------|---------------------|-------------|
| Mixed Nuts 8oz... | Apr 28, 2017 7:11 AM | Apr 28, 2017 12:31 AM | | | Manual - Incompl... | 0 |

 **Begin OEE Run**
 **End OEE Production**

The **MES Schedule Selector** component allows us to control any type of scheduled run, whether it is a work order, product code run or a maintenance or any other operation. This concludes Scheduling. In the next section we will build out some OEE analysis screens and reports.


4.3.6 Analysis and Reporting

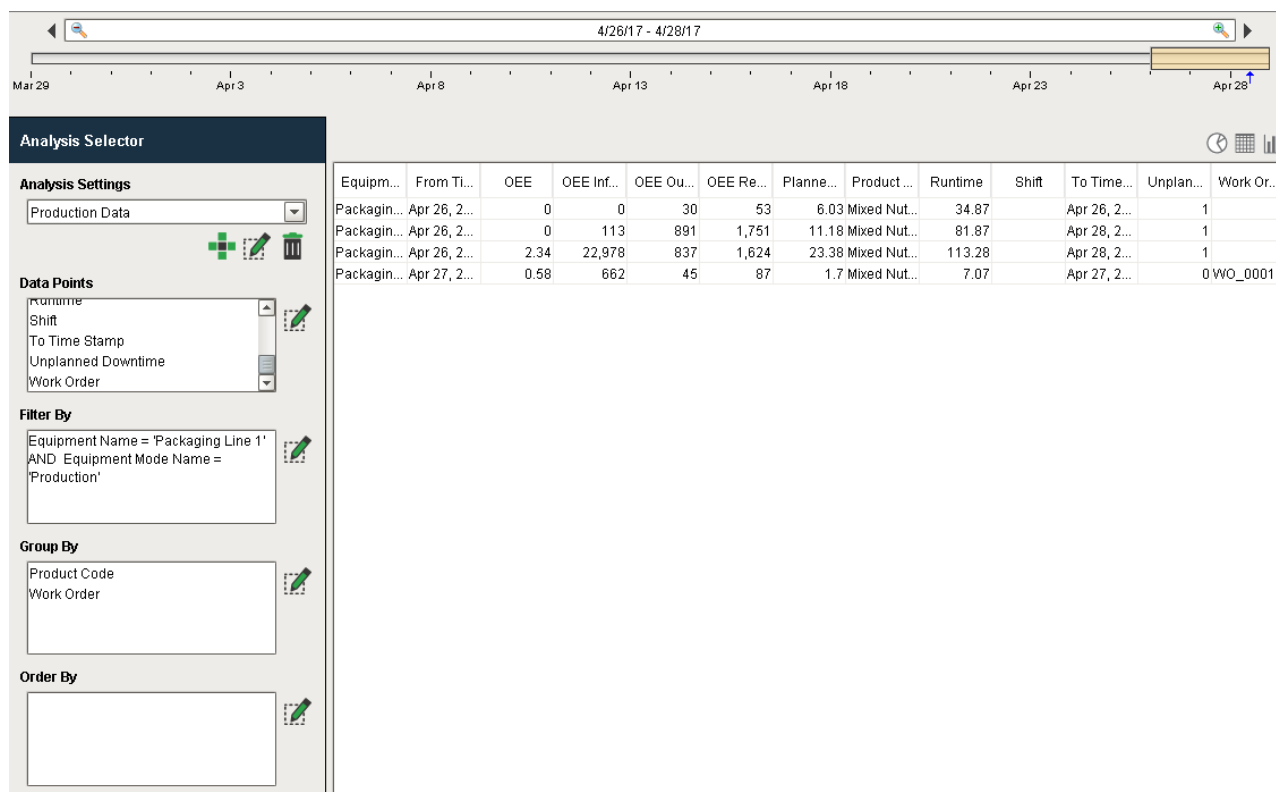
In this section, we will use the analysis components to create Stored Analysis Settings (SAS) that will allow us to access the data captured for our packaging operations. We will use the **MES Analysis Controller**, **MES Analysis Selector**, **Production Bar Chart** and **Production Pie Chart** as well as an Ignition Power Table component. We already have a screen built using these components that we can use to start creating our analysis with.

- Open the window **Impromptu Analysis** under Analysis.
- Go into preview mode by pressing **F5** or clicking on the



button on the menu bar.

- Click the  button to create a new Stored Analysis and call it **Production Data**.
- Add the following Data Points:
 - **Equipment Folder:** Equipment Name, Product Code, Work Order
 - **General:** Shift, From Time Stamp, To Time Stamp
 - **OEE:** OEE, OEE Infeed Count, OEE Outfeed Count, OEE Reject Count, Planned Downtime, Runtime, Unplanned Downtime
 - **Availability:** OEE Availability
 - **Performance:** OEE Performance
 - **Quality:** OEE Quality
- Add the following Filter By:
 - Equipment Name = 'Packaging Line 1' AND Equipment Mode Name = 'Production'
- Add the following Group By:
 - Product Code, Work Order, Shift



We have now created a Stored Analysis that will return production data for Packaging Line when the line mode was in PRODUCTION. We can call this Stored Analysis on any screen using the MES Analysis Controller and pointing it to this Stored Analysis or by using the [system.mes.analysis](#) scripting functions. We can also point to this Stored Analysis when creating a report by using it as a data source. Because we can now add security to who can access or modify this Stored Analysis, we can ensure that it does not get accidentally modified or deleted.

- ✓ We could have passed the name of the line to the 'Equipment Name' filter as a parameter and that would make this analysis more dynamic.

For more information on analysis, refer to [Analysis and Reporting](#) for more details.

i Exporting and Importing Analysis Settings

The **Production Data** analysis you just created is stored as an MES object. You can use the **MES Object Exporter** window to export and import these analysis settings between projects.

Create Production Data Screen

We will access the Stored Analysis Setting we just created called **Production Data** in a new window using the [MES Analysis Controller](#).

- Create a new window under **Analysis** called **Production Data**.
- Drag a **Date Range**, **MES Object Selector**, **MES Analysis Controller** and **Power Table** component onto the window.
- Bind the **Start Date** and **End Date** property of the MES Analysis Controller to the Date Range **Start Date** and **End Date** properties.
- Right-click on the MES Analysis Controller, choose **Customizers, Custom Analysis Settings** and select the **Production Data** Stored Analysis that we created earlier.
- Bind the Power Table data property to the **MES Analysis Controller** Ignition Dataset property.

We are able to pull data from Stored Analysis Settings using the MES Analysis Controller, but let's make this analysis a little more dynamic by passing the production line we are interested in getting data for, as a passed parameter to the Stored Analysis.

Dynamic Production Line Selection

- Open the **Impromptu Analysis** window under **Analysis**.
- Add a custom property called **eqPath** to the **MES Analysis Selector** component.
- In preview mode (F5), edit the **Filter By** expression:
 - Delete **Equipment Name = 'Packaging Line 1'**.
 - Browse to **Equipment / Equipment Path** in the Filter Pane, select the **=** Operator and select the **eqPath** in the Parameter dropdown. **Add** the new expression.

*In the expression pane, you should now see **Equipment Path = @eqPath AND Equipment Mode Name = 'Production'***

- Add a **MES Object Selector** component to the **Impromptu Analysis** window and enable **Include MES Line Objects**.
- Bind the MES Analysis Selector **eqPath** custom property to the MES Object Selector **Equipment Item Path** property.

You should now be able to select Packaging Line 1 in the MES Object Selector and see results in the power table.

The screenshot shows the Impromptu Analysis window. At the top is a timeline from April 1 to May 1, 2017. Below the timeline is the **Analysis Selector** component. A dropdown menu shows 'Packaging Line 1' selected. To the right of the dropdown is a table with production data.

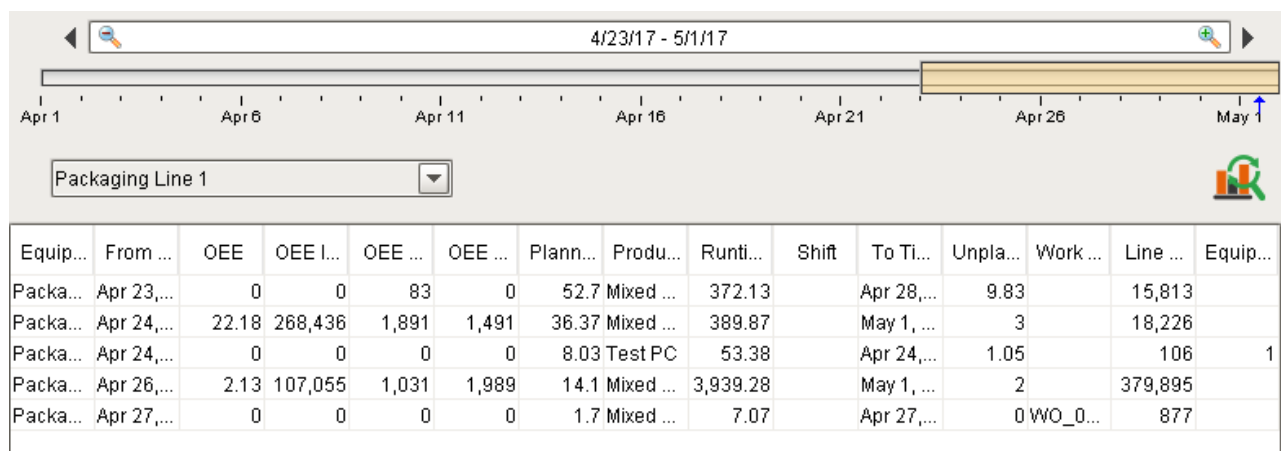
| Equip... | From T... | OEE | OEE Inf... | OEE O... | OEE R... | Planne... | Produc... | Runtime | Shift | To Tim... | Unplan... | Work O... | Line T... |
|------------|--------------|--------|------------|----------|----------|-----------------|-----------|---------|-------|-------------|-----------|-----------|-----------|
| Packagi... | Apr 29, 2... | 0 | 104,316 | 503 | 992 | 0 Mixed N... | | 0 | | May 1, 2... | 0 | | 334,979 |
| Packagi... | May 1, 2... | 337.04 | 235,136 | 462 | 904 | 7.12 Mixed N... | | 57.08 | | May 1, 2... | 1 | | 6,259 |

On the left side of the window, there are four configuration panels:

- Analysis Settings:** A dropdown menu set to 'Production Data'.
- Data Points:** A list of data points including 'Equipment Name', 'From Time Stamp', 'OEE', 'OEE Infeed Count', and 'OEE Outfeed Count'.
- Filter By:** A text area containing the expression 'Equipment Path = @eqPath AND Equipment Mode Name = 'Production''.
- Group By:** A list of fields including 'Product Code' and 'Work Order'.

- Open the **Production Data** window under **Analysis**.
- Enable **Include MES Line Objects** on the **MES Object Selector**.
- Add a custom property called **eqPath** (String datatype) to the **MES Analysis Controller** component and bind it to the *Equipment Item Path* property of the **MES Object Selector**.
- Right-click on the MES Analysis Controller, choose **Customizers, Custom Analysis Settings** and re-select the **Production Data** Stored Analysis that we created earlier.
- Select **Packaging Line 1** in the **MES Object Selector**.

You should see OEE results for Packaging Line 1 in the power table.




We now have a Stored Analysis called **Production Data** that takes in a parameter value for the equipment path and we are using a MES Analysis Controller that uses this Stored Analysis to create a dynamic production screen detailing production run information for a user selected line.

Create Production Report

We will now create a production report using the built-in Ignition report designer and access the same Stored Analysis Setting we just created called **Production Data**.

- Create a new report under Reports called **Production Report**.
- Click on the **Data** tab and add a new parameter called *eqPath*. Add the equipment path for Packaging line 1 for the default value.

You'll have to add extra slashes so that the report designer does not think they are escape characters i. e. *[global]\\Nuts Unlimited\\Folsom\\Packaging\\Packaging Line 1'*

- Under **Data Sources**, click the **+** button, select **MES Analysis** and select the **Production Data** stored analysis.
- Edit the **Production Data** stored analysis by clicking on the  icon.
 - Use the **Start Date** parameter dropdown to select the **StartDate** parameter and use the **End Date** parameter dropdown to select the **EndDate** parameter.
- **Save** your changes.

We now have a report that will pass the start and end date, as well as the equipment path for the production line we are interested in. Now we need to design the report to display data of interest.

Project Browser

- Global
 - Project
 - Properties
 - Scripts
 - Transaction Groups
 - Windows
 - Templates
 - Reports
 - Production Report

Key Browser

Report designer inactive.

Property Inspector

Report designer inactive.

Report Overview

Data

Design

Preview

Schedule

Parameters

StartDate

EndDate

eqPath

Data Sources

1 MES Analysis - Production Data

Analysis Selector

Analysis Settings

Production Data

Data Points

Equipment Name

From Time Stamp

OEE

OEE Infeed Count

OEE Outfeed Count

OEE Reject Count

Planned Downtime

Filter By

Equipment Path = @eqPath AND
Equipment Mode Name =
'Production'

Group By

Product Code

Work Order

Order By

General

<back

Save

Name

Production Data

Description

Scope

Public Private

Permissions

| Role | Execute | Modify |
|---------------|-------------------------------------|-------------------------------------|
| Administrator | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |

Settings

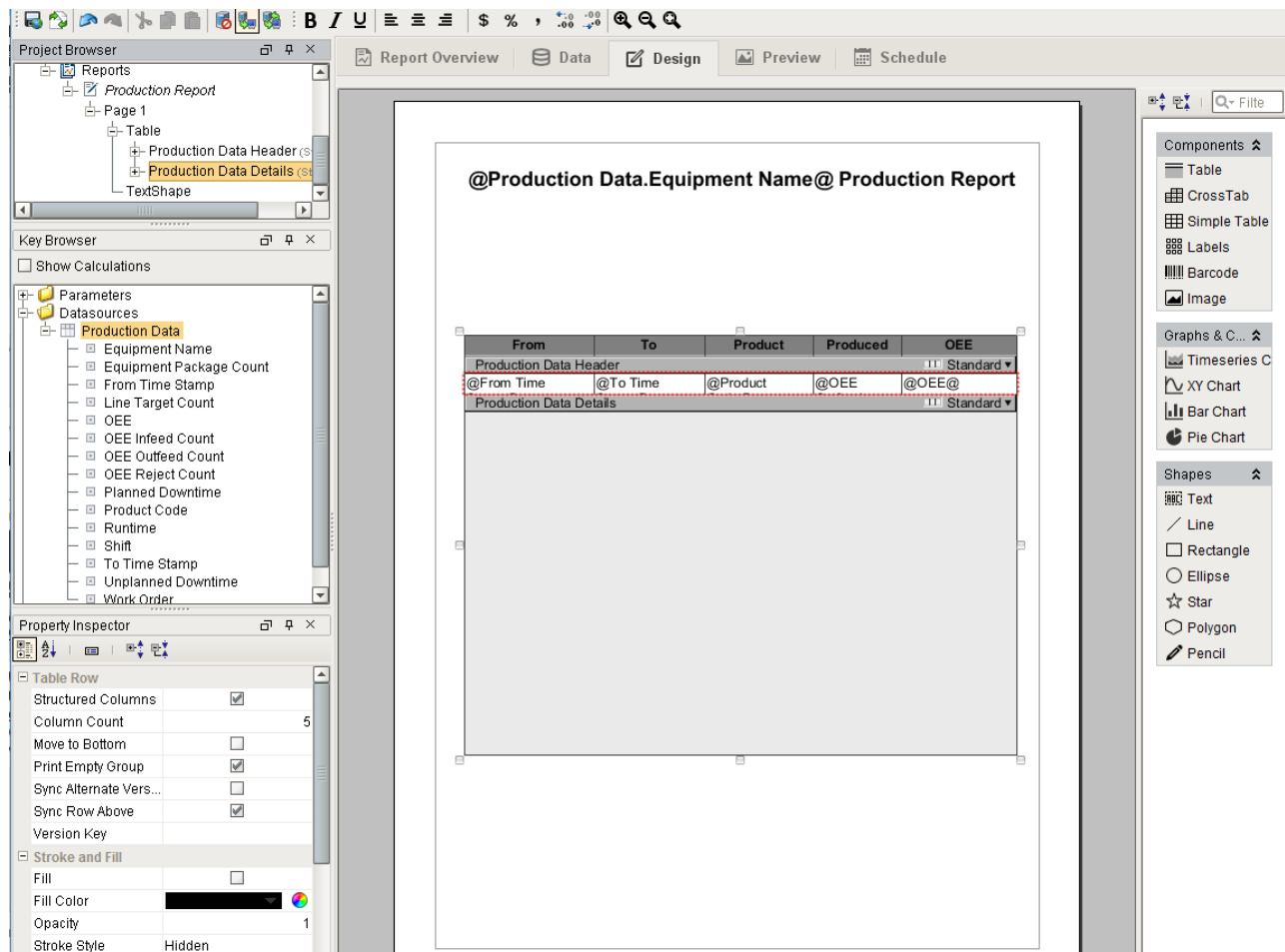
| Name | Value |
|----------------------|-----------|
| Start Date Parameter | StartDate |
| End Date Parameter | EndDate |

- Click on the **Design** tab.
- Under **Key Browser**, drag the Production Data data source onto the screen.
- Under **Property Inspector**, select the **Configure Table** tab and enable **Header** under **Rows**.
- Select the **Production Data Details** row on the page and set the **Sync Row Above** property to true and the **Column Count** value to 5 in the property inspector pane.

You may have to adjust the width of the table on the page to see the new columns.

- Add **From**, **To**, **Product**, **Produced** and **OEE** to the Production Data header columns.
- Under Datasources in the **Key Browser**, drag **From Time Stamp**, **To Time Stamp**, **Product Code**, **OEE Outfeed Count** and **OEE** to the respective table column cells.
- Drag a **Text** Shape to the top of the page and drag **Equipment Name** from the Key Browser to the text field. Add 'Production Report' to the text field.
- Select the **Preview** tab to view your work.

To finish this off, we will create a new window that will display the report we just created.



- Create a folder called **Reports** and add a new window here called **Production Report**.
- Drag a **Report Viewer** component from the **Reporting** component palette on the window.
- Select the **Production Report** from the **Data/Report Path** property of the Report viewer component.
- Add a **MES Object Selector** component to the window and enable **Include MES Line Objects**.
- Bind the *eqPath* property of the Report Viewer to the MES Object Selector *Equipment Path* property.
- Add two **Popup Calendar** components to the window and bind the *Start Date* and *End Date* properties of the **Report Viewer** to the *Date* properties of the popup calendars.

03/01/2017 12:43 PM
05/01/2017 12:44 PM
Packaging Line 1

Packaging Line 1 Production Report

| From | To | Product | Produced | OEE |
|--------------|--------------|-----------------|----------|------|
| Apr 20, 2017 | May 1, 2017 | Mixed Nuts 16oz | 2542 | 0.99 |
| Apr 21, 2017 | Apr 28, 2017 | Mixed Nuts 8oz | 582 | 1 |
| Apr 24, 2017 | Apr 24, 2017 | Test PC | 0 | 0.98 |
| Apr 26, 2017 | May 1, 2017 | Mixed Nuts 22oz | 1031 | 4.51 |
| Apr 27, 2017 | Apr 27, 2017 | Mixed Nuts 8oz | 0 | 1 |

Now we have a dynamic report that allows a user to select the production line and time frame to return production data for. Because we are using the Ignition Report designer, this report can be auto scheduled to be created and emailed out based on a schedule. This is not a particularly exciting or useful report, but that's ok. The purpose here is show how you can bring MES data into the report, not how to build fancy reports.

For more information, refer to the reporting section of the [Ignition Help Manual](#).

4.4 OEE 2.0 Review

This concludes the OEE 2.0 portion of the online tutorial. In the next section we shall learn about using the Web Services module to pull down Work Orders and Product codes from an ERP system.

4.4.1 What We Covered

In summary, we covered the following:

- Features and Framework of the OEE 2.0 module
- How to model a production line using line and cell objects in the Production model
- Using MES Counters
- The different Downtime Detection Modes
- Extending data collection using Additional Factors
- Managing Shift Configuration
- How to configure equipment with the OEE Equipment Manager component
- How to configure materials using the OEE Material Manager component
- How to start production runs using the OEE Run Director component
- Using MES Object Events to customize implementation
- Using the OEE Downtime Table to edit downtime events
- Adding lot tracking to a production line
- Using sub-lots
- Using Live Analysis to provide runtime values
- Creating Work Orders using the MES Work Order Table
- Scheduling production runs using the MES Schedule View component and starting scheduled runs using the MES Schedule Selector component
- Analyzing production data using the MES Analysis Selector and MES Analysis Controller components
- Creating a production report

4.4.2 What We Didn't Cover

What we haven't covered or touched in great detail are:

- Editing production values using the MES Value Editor component
- Creating analysis settings using the `system.mes.analysis.createMESAnalysisSettings` scripting function
- Using the OEE Time Chart component
- Running Changeovers with multiple products on the line
- Parameterized Tag Paths
- Adding Custom User Menu Items to MES Schedule View
- How to create Work Orders and scheduled operations through scripting
- How to start scheduled operations through scripting

4.4.3 Additional Resources




We can't cover everything in a tutorial, but we do have more resources available in the [MES Help manual](#) and [Sepasoft Knowledge Base](#) for the areas we didn't cover.


» WEB SERVICES 2.0

- › Create Data Sources
- › Create getWorkOrders() Endpoint
- › Create getProductCodes() Endpoint
- › Create getProductCodeAttributes() Endpoint
- › Create postProductionResults() Endpoint
- › Web Service Consumer
- › Web Service Review

5 Web Services 2.0

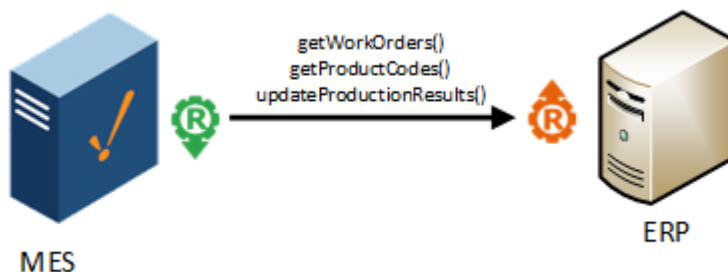
In this section, we'll learn how to use the Web Services module to create a Web Service Provider (WSP) and a Web Service Consumer (WSC). This allows us to turn our Ignition Gateway into an information hub and share any type of data (SCADA, operator input, MES information, etc.) with any other information system that can consume Web Services. In this tutorial, we will add a WSP that will allow us to act as our ERP system to provide work orders, product codes and allow other systems to update production counts and status against work orders. We will then create a WSC that we can use access these endpoints from our Web Service Provider.

- Open the Ignition Designer and make sure you're in **Read/Write Mode**.
- Open **Help -> About Ignition Designer** and ensure that you have Web Services Module version 2.9.x or greater. If you haven't yet installed the Web services module, refer to [Install the MES Modules](#).
-  Download and import the `MES_2.0_Training_WSP_Base.proj`. This contains a couple of pre-built screens that we will use during the tutorial.
-  Download the `WS Tags.xml` and import using the  icon in the Tag Browser pane..

This contains a tag dataset that will hold our ERP work orders, product codes and attributes. You may need to refresh  the tags for them to show up.

5.1 Web Service Provider

In this section, we will be creating the following RESTful Web Service Provider Endpoints...



.. and have our Ignition Gateway act like an ERP system serving up work order and product code data to whoever wants it. But before we do that, there are a couple of screens we should make so that we can create the work order and product code information that we will share.

There are a number of videos on the right hand side that go through the steps of creating RESTful endpoints in case you get stuck or need more detailed information.


5.1.1 Create Data Sources

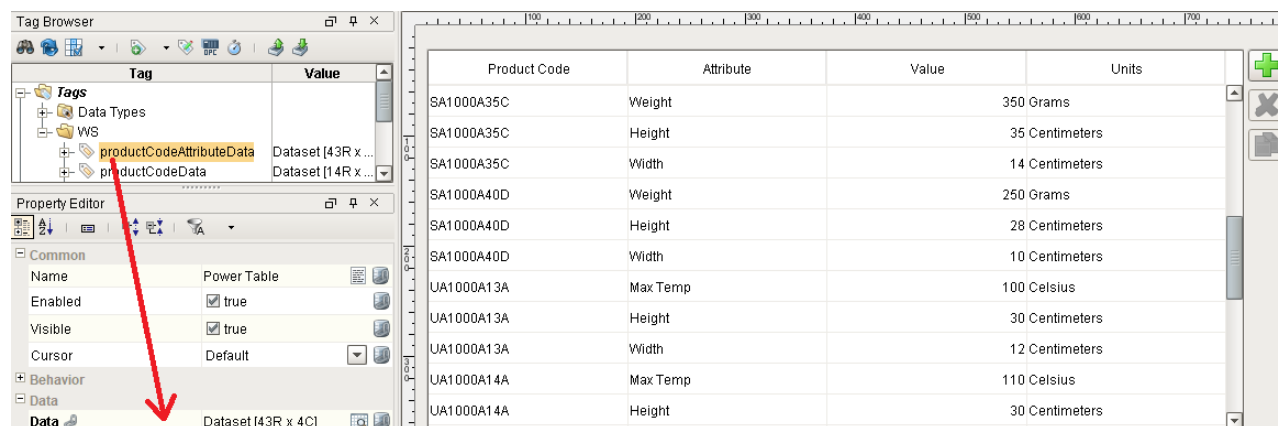
Create the ERP Product Code Attribute Screen

Later on in the Web Services Consumer tutorial, we will be calling the `getProductCodeAttributes()` endpoint to return a list of attributes by product code, so we need to create some data. We could hard code this in the endpoint script itself, but to make it a little more realistic we will use a dataset. We could do this by creating a custom database table, but for this example we will use an ignition memory tag of type dataset.

- Open up the project window called **createERPProductCodeAttributes** under the Web Services\WSP folder.
- Drag the **productCodeAttributeData** tag you just imported and drop it onto the **data** property of the **Power Table** on this window.

We have now created a bi-directional binding of the power table dataset and the tag dataset. As we make changes to the product codes in the power table, the tag dataset will be updated. The tag you imported already has some product codes in it.

- Go into **PreviewMode** by clicking  or pressing **F5** and start adding or modifying product code attributes.



The screenshot shows the Ignition software interface. On the left, the 'Tag Browser' pane displays a tree structure with 'Tags' expanded, showing 'Data Types', 'WS', and 'productCodeAttributeData'. The 'Property Editor' pane shows the 'Data' property of the 'Power Table' set to 'Dataset [43R x 4C]'. A red arrow points from the 'productCodeAttributeData' tag in the Tag Browser to the 'Data' property in the Property Editor. On the right, a data table is displayed with the following data:


| Product Code | Attribute | Value | Units |
|--------------|-----------|-------|-------------|
| SA1000A35C | Weight | 350 | Grams |
| SA1000A35C | Height | 35 | Centimeters |
| SA1000A35C | Width | 14 | Centimeters |
| SA1000A40D | Weight | 250 | Grams |
| SA1000A40D | Height | 28 | Centimeters |
| SA1000A40D | Width | 10 | Centimeters |
| UA1000A13A | Max Temp | 100 | Celsius |
| UA1000A13A | Height | 30 | Centimeters |
| UA1000A13A | Width | 12 | Centimeters |
| UA1000A14A | Max Temp | 110 | Celsius |
| UA1000A14A | Height | 30 | Centimeters |

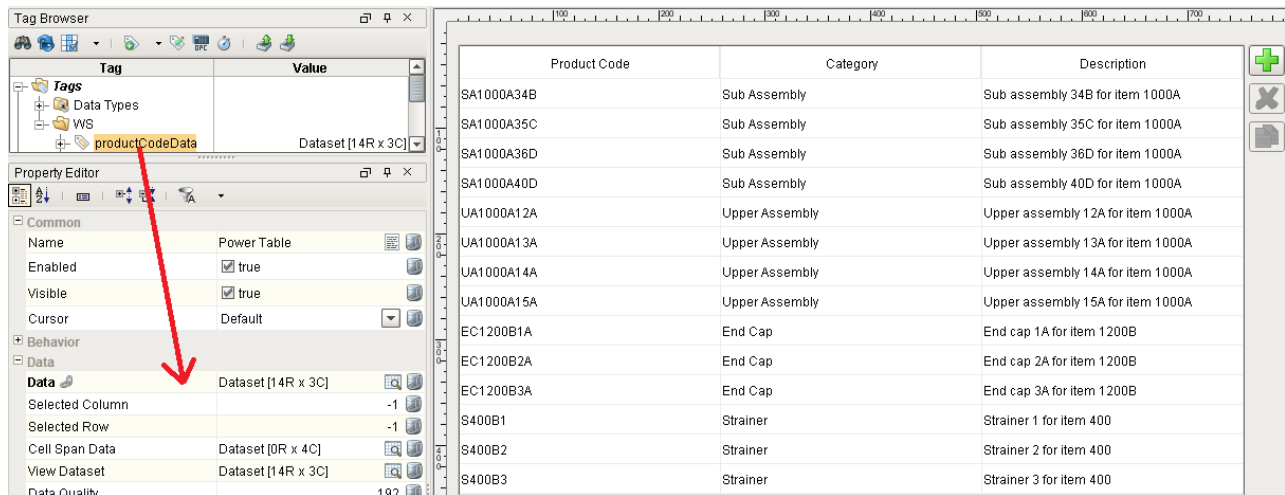
Create the ERP Product Code Screen

Later on in the Web Services Consumer tutorial, we will be calling the `getProductCodes()` endpoint to return a list of product codes, so we need to create some data. We could hard code this in the endpoint script itself, but to make it a little more realistic we will use a dataset. We could do this by creating a custom database table, but for this example we will use an ignition memory tag of type dataset.

- Open up the project window called **createERPProductCodes** under the Web Services\WSP folder.
- Drag the **productCodeData** tag you just imported and drop it onto the **data** property of the **Power Table** on this window.

We have now created a bi-directional binding of the power table dataset and the tag dataset. As we make changes to the product codes in the power table, the tag dataset will be updated. The tag you imported already has some product codes in it.

- Go into **PreviewMode** by clicking  or pressing **F5** and start adding or modifying product codes.



The screenshot shows the Ignition IDE interface. On the left, the **Tag Browser** displays a tree structure with **Tags** expanded, showing **productCodeData** as a **Dataset [14R x 3C]**. A red arrow points from this tag to the **Data** property in the **Property Editor** on the left. The **Property Editor** shows the **Power Table** with properties like **Name**, **Enabled**, **Visible**, **Cursor**, and **Data** (set to **Dataset [14R x 3C]**). The **Power Table** itself is displayed on the right, showing a table with columns **Product Code**, **Category**, and **Description**. The table contains 14 rows of data, including product codes like SA1000A34B, SA1000A35C, SA1000A36D, SA1000A40D, UA1000A12A, UA1000A13A, UA1000A14A, UA1000A15A, EC1200B1A, EC1200B2A, EC1200B3A, S400B1, S400B2, and S400B3.


| Product Code | Category | Description |
|--------------|----------------|-----------------------------------|
| SA1000A34B | Sub Assembly | Sub assembly 34B for item 1000A |
| SA1000A35C | Sub Assembly | Sub assembly 35C for item 1000A |
| SA1000A36D | Sub Assembly | Sub assembly 36D for item 1000A |
| SA1000A40D | Sub Assembly | Sub assembly 40D for item 1000A |
| UA1000A12A | Upper Assembly | Upper assembly 12A for item 1000A |
| UA1000A13A | Upper Assembly | Upper assembly 13A for item 1000A |
| UA1000A14A | Upper Assembly | Upper assembly 14A for item 1000A |
| UA1000A15A | Upper Assembly | Upper assembly 15A for item 1000A |
| EC1200B1A | End Cap | End cap 1A for item 1200B |
| EC1200B2A | End Cap | End cap 2A for item 1200B |
| EC1200B3A | End Cap | End cap 3A for item 1200B |
| S400B1 | Strainer | Strainer 1 for item 400 |
| S400B2 | Strainer | Strainer 2 for item 400 |
| S400B3 | Strainer | Strainer 3 for item 400 |

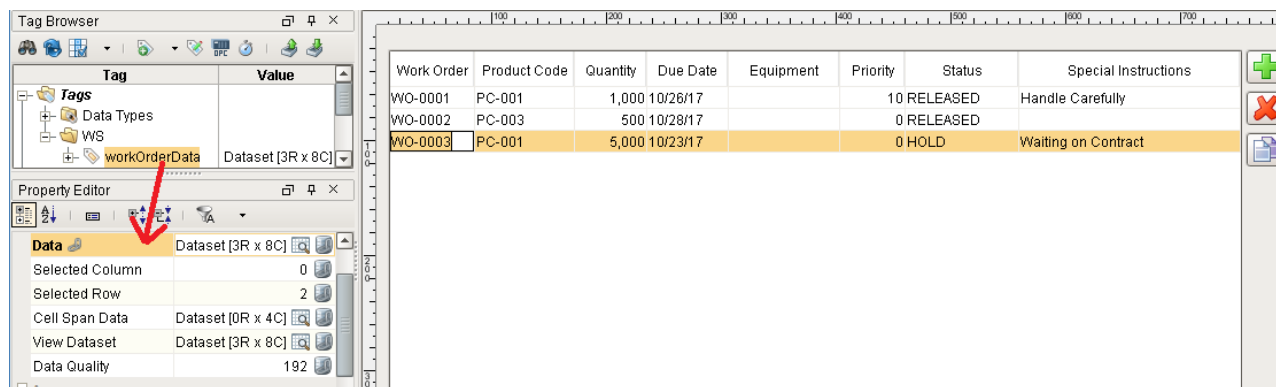
Create the ERP Work Order Screen

Later on in the Web Services Consumer tutorial, we will be calling the `getWorkOrders()` endpoint to return a list of work orders, so we need to create some data. We could hard code this in the endpoint script itself, but to make it a little more realistic we will use a dataset. We could do this by creating a custom database table, but for this example we will use an ignition memory tag of type dataset.

- Open up the project window called **createERPWorkOrders** under the Web Services\WSP folder.
- Drag the **workOrderData** tag you just imported and drop it onto the **data** property of the **Power Table** on this window.


We have now created a bi-directional binding of the power table dataset and the tag dataset. As we make changes to the work orders in the power table, the tag dataset will be updated. The tag you imported already has some work orders in it. You'll probably want to change the due date.

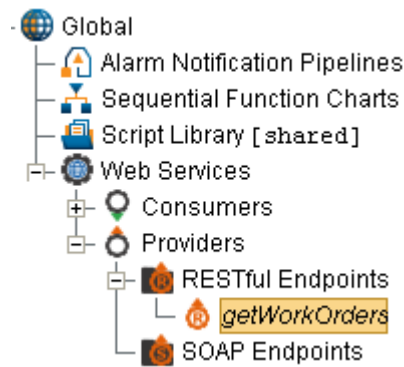
- Go into **PreviewMode** by clicking  or pressing **F5** and start adding and modifying work orders.




5.1.2 Create getWorkOrders() Endpoint

Now we will create the RESTful Web Service Provider for the getWorkOrders() endpoint.

- Navigate to **Global -> Web Services -> Providers > RESTful Endpoints** in the Project Browser panel of the Ignition designer.
- Right-click and select  **New RESTful Endpoint**.
- Rename this Endpoint **getWorkOrders**.



In the **Settings** panel, we can leave the default General and HTTP Authentication settings.

- Uncomment the code block in the `do_GET()` script function.
- **Save** the changes by pressing .

Settings

General

HTTP Method: GET

Data Format: JSON

Encoding: UTF-8

☐ Redirect to SSL

HTTP Authentication

Type: None

User Source: default

Required Role(s):

Script

```

def do_GET(request):
    """
    Responds to an incoming HTTP request with a response in dict format.

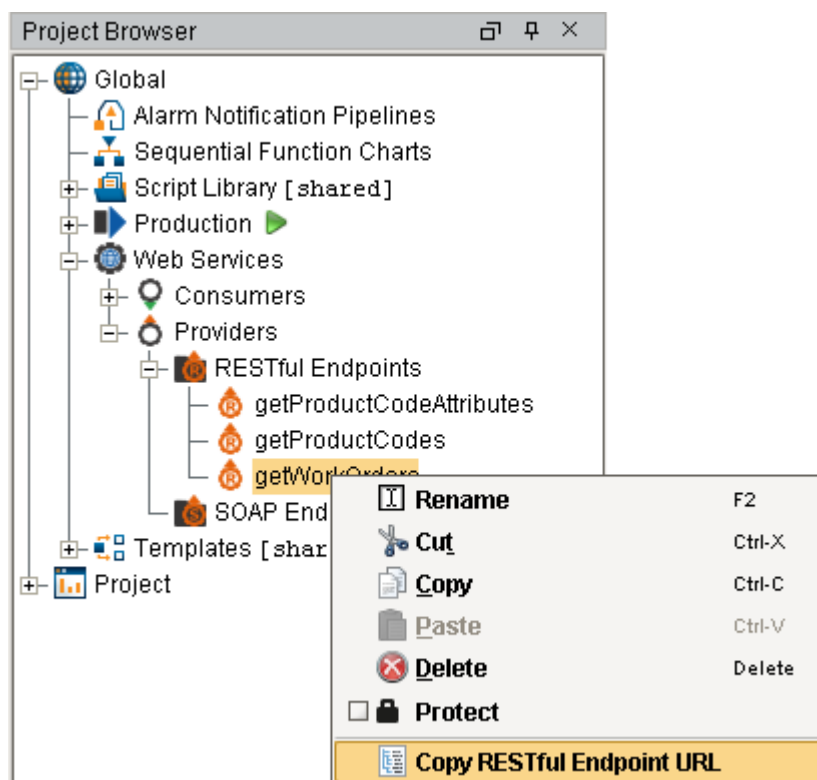
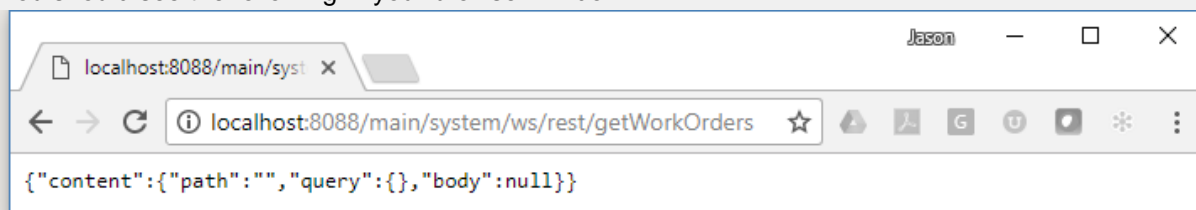
    Puts a HTTP status code in integer format at the key 'code' in the response
    dict.

    Puts a content in dict format at the key 'content' in the response dict.

    Arguments:
        request: A dict with an incoming HTTP request information. The path
        parameter is contained within a string at request['path']. The query
        parameters are contained within a dict at request['query']. The message
        body parameter is contained within a dict at request['body']. The HTTP
        headers are contained within a dict at request['headers'].
    """
    1
    2 path_string = request['path']
    3 query_dict = request['query']
    4 body_dict = request['body']
    5 content_dict = {'path': path_string, 'query': query_dict, 'body': body_dict}
    6 return {'code': 200, 'content': content_dict}
  
```

- Right-click on **getWorkOrders** in the Project Browser pane and copy the URL into your web browser.

You should see the following in your browser window...



When we created the *getWorkOrders* endpoint, the code stub for how input and output from the endpoint is handled was automatically created for us. All we have to do now is modify the script to handle the input and return whatever data we want to.

- Delete the stub code in the script window (all lines after the purple comment block) and replace it with the following...

```

query_dict = request['query']

if query_dict.has_key('woStatus') == False:
    return {'code': 200, 'content': {'path': request['path'],
'query': request['query'], 'body': request['body']}}
    #We'll only define one input parameter, a filter by WO Status
    woStatus = query_dict['woStatus']

    #Work Order data is stored in a tag dataset. We can right-
    click on a tag and say 'copy tag path'
    #but we need to add the provider i.e. '[default]' to make it
    a fully qualified tag path
    woData = system.tag.read("[default]WS/workOrderData").value
    woDict = {}
    woList = []

    #We'll now grab the WO data from the tag dataset and create
    the JSON data structure that will be passed back
    for row in range(woData.rowCount):
        if woData.getValueAt(row, 'Status') == woStatus or
woStatus == 'ALL':
            rowDict = {}
            for col in range(woData.columnCount):
                rowDict[str(woData.getColumnName(col))] = str(woDa
ta.getValueAt(row, col))

            woList.append(rowDict)
        woDict['workorders'] = woList

    return {'code': 200, 'content': woDict}

```

Script

```


1 def do_GET(request):
2     """
3     Responds to an incoming HTTP request with a response in dict format.
4
5     Puts a HTTP status code in integer format at the key 'code' in the response
6     dict.
7
8     Puts a content in dict format at the key 'content' in the response dict.
9
10    Arguments:
11        request: A dict with an incoming HTTP request information. The path
12                parameter is contained within a string at request['path']. The query
13                parameters are contained within a dict at request['query']. The message
14                body parameter is contained within a dict at request['body']. The HTTP
15                headers are contained within a dict at request['headers'].
16    """
17    query_dict = request['query']
18
19    if query_dict.has_key('woStatus') == False:
20        return {'code': 200, 'content': {'path': request['path'], 'query': request['query'], 'body': request['body']}}
21
22    woStatus = query_dict['woStatus']
23
24    woData = system.tag.read("[default]WS/workOrderData").value
25    woDict = {}
26    woList = []
27
28    for row in range(woData.rowCount):
29        if woData.getValueAt(row, 'Status') == woStatus or woStatus == 'ALL':
30            rowDict = {}
31            for col in range(woData.columnCount):
32                rowDict[str(woData.getColumnName(col))] = str(woData.getValueAt(row, col))
33
34            woList.append(rowDict)
35    woDict['workorders'] = woList
36
37    return {'code': 200, 'content': woDict}
38

```

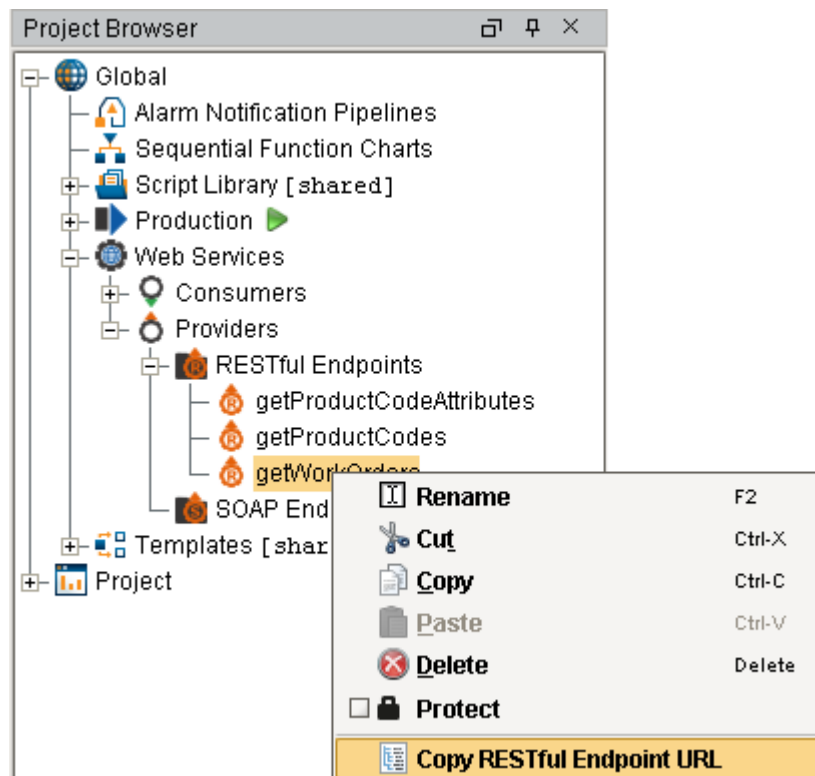
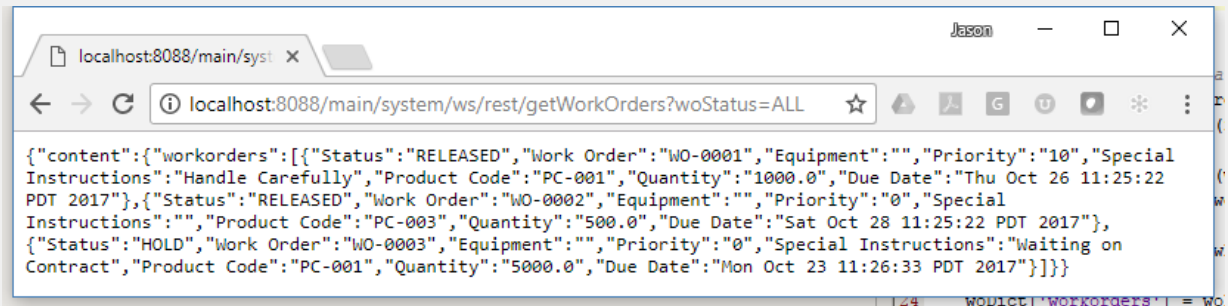
☐ Show whitespace characters

HTTP Status Codes

The value of 200 is not arbitrary, but instead uses the standard HTTP response code for success. To see valid HTTP response codes, please go here: https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

- **Save** the changes by pressing .
- Right-click on **getWorkOrders** in the Project Browser pane and copy the URL into your web browser.
- Add **?woStatus=ALL** to the end of the URL

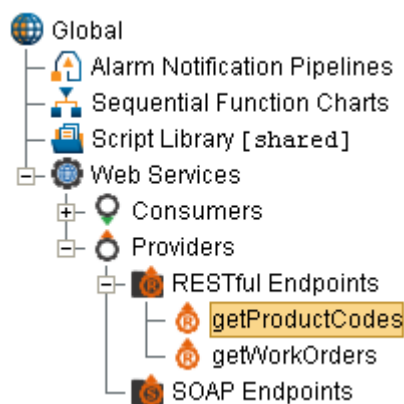
You should see the following in your browser window...



5.1.3 Create getProductCodes() Endpoint

Now we will create the RESTful Web Service Provider for the getProductCodes() endpoint.

- Navigate to **Global -> Web Services -> Providers > RESTful Endpoints** in the Project Browser panel of the Ignition designer.
- Right-click and select **New RESTful Endpoint**.
- Rename this Endpoint **getProductCodes**.
- In the **Settings** panel, we can leave the default General and HTTP Authentication settings.



The code stub for how input and output from the endpoint is handled was automatically created for us. All we have to do now is modify the script to handle the input and return whatever data we want to.

- Delete the stub code in the script window (all lines after the purple comment block) and replace it with the following...

```

query_dict = request['query']

if query_dict.has_key('category') == False:
    return {'code': 200, 'content': {'path': request['path'],
'query': request['query'], 'body': request['body']}}
    #We'll only define one input parameter, a filter by product
code category
    category = query_dict['category']

    #Product Code data is stored in a tag dataset. We can right-
click on a tag and say
    #'copy tag path' but we need to add the provider i.e.
'[default]' to make it a
    #fully qualified tag path
pcData = system.tag.read("[default]WS/productCodeData").value
pcDict = {}
pcList = []

    #We'll now grab the product code data from the tag dataset
and create the JSON data structure
    #that will be passed back
    for row in range(pcData.rowCount):
        if pcData.getValueAt(row, 'Category') == category or
category == 'ALL':
            rowDict = {}
            for col in range(pcData.columnCount):
                rowDict[str(pcData.getColumnName(col))] = str(pcDa
ta.getValueAt(row, col))

            pcList.append(rowDict)
pcDict['productcodes'] = pcList

    return {'code': 200, 'content': pcDict}

```

- Save the changes by pressing .

Script

```

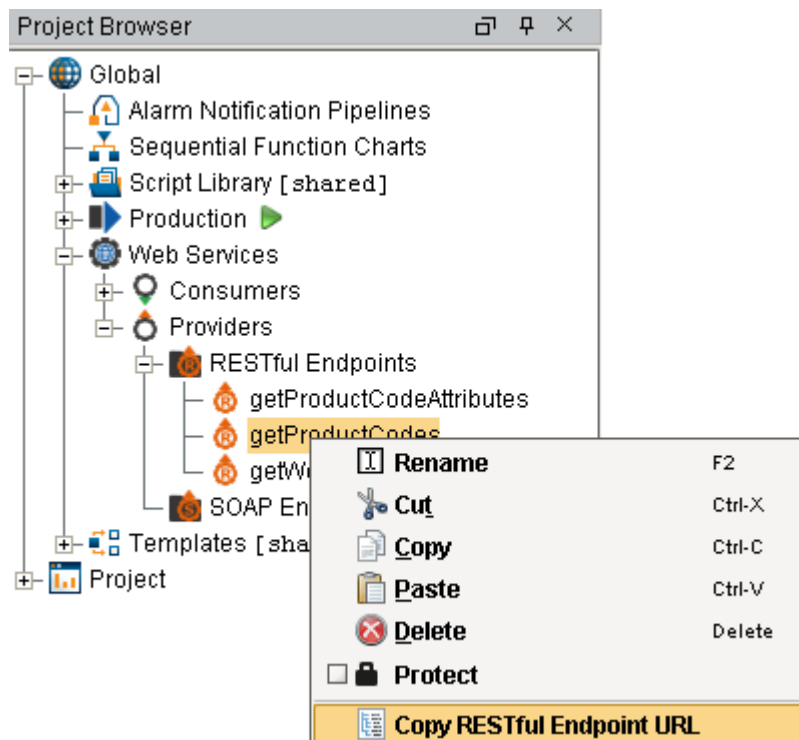
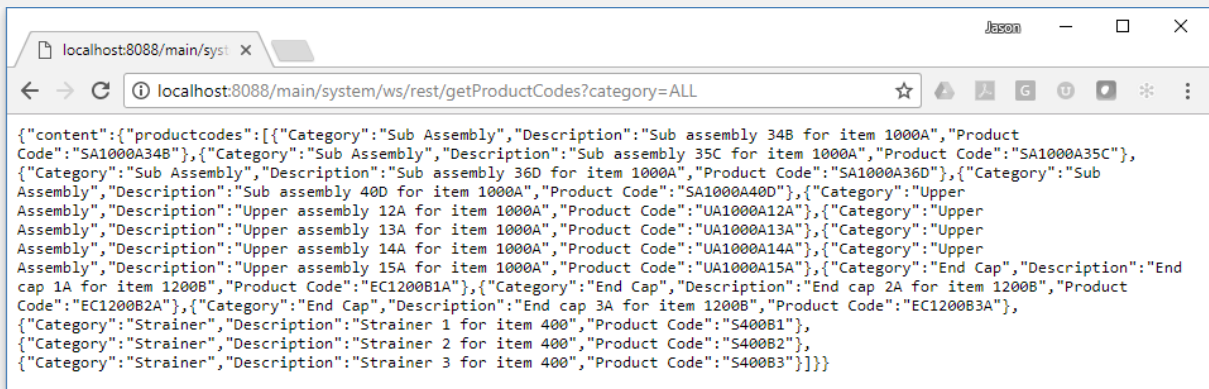
1 def do_GET(request):
2     """
3     Responds to an incoming HTTP request with a response in dict format.
4
5     Puts a HTTP status code in integer format at the key 'code' in the response
6     dict.
7
8     Puts a content in dict format at the key 'content' in the response dict.
9
10    Arguments:
11        request: A dict with an incoming HTTP request information. The path
12                parameter is contained within a string at request['path']. The query
13                parameters are contained within a dict at request['query']. The message
14                body parameter is contained within a dict at request['body']. The HTTP
15                headers are contained within a dict at request['headers'].
16    """
17    query_dict = request['query']
18
19    if query_dict.has_key('category') == False:
20        return {'code': 200, 'content': {'path': request['path'], 'query': request['query'], 'body': request['body']}}
21
22    ##We'll only define one input parameter, a filter by product code category
23    category = query_dict['category']
24
25    #Product Code data is stored in a tag dataset. We can right-click on a tag and say
26    #'copy tag path' but we need to add the provider i.e. '[default]' to make it a
27    #fully qualified tag path
28    pcData = system.tag.read("[default]WS/productCodeData").value
29    pcDict = {}
30    pcList = []
31
32    ##We'll now grab the product code data from the tag dataset and create the JSON data structure
33    #that will be passed back
34    for row in range(pcData.rowCount):
35        if pcData.getValueAt(row, 'Category') == category or category == 'ALL':
36            rowDict = {}
37            for col in range(pcData.columnCount):
38                rowDict[str(pcData.getColumnName(col))] = str(pcData.getValueAt(row, col))
39
40            pcList.append(rowDict)
41    pcDict['productcodes'] = pcList
42
43    return {'code': 200, 'content': pcDict}

```

☐ Show whitespace characters

- Right-click on **getProductCodes** in the Project Browser pane and copy the URL into your web browser.
- Add **?category=ALL** to the end of the URL

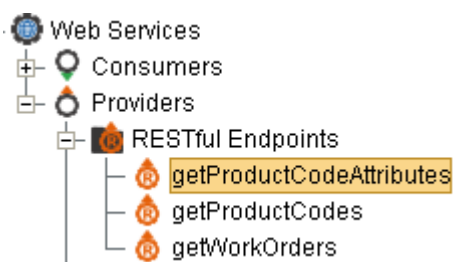
You should see the following in your browser window...



5.1.4 Create getProductCodeAttributes() Endpoint

Now we will create the RESTful Web Service Provider for the getProductCodeAttributes() endpoint.

- Navigate to **Global -> Web Services -> Providers > RESTful Endpoints** in the Project Browser panel of the Ignition designer.
- Right-click and select **New RESTful Endpoint**.
- Rename this Endpoint **getProductCodeAttributes**.
- In the **Settings** panel, we can leave the default General and HTTP Authentication settings.



The code stub for how input and output from the endpoint is handled was automatically created for us. All we have to do now is modify the script to handle the input and return whatever data we want to.

- Delete the stub code in the script window (all lines after the purple comment block) and replace it with the following...

```

query_dict = request['query']

if query_dict.has_key('productcode') == False:
    return {'code': 200, 'content': {'path': request['path'],
'query': request['query'], 'body': request['body']}}

#We'll only define one input parameter, a filter by product
code
productCode = query_dict['productcode']

#Product Code attribute data is stored in a tag dataset. We
can right-click on a tag and say
# 'copy tag path' but we need to add the provider i.e.
'[default]' to make it a
#fully qualified tag path
pcData = system.tag.read("[default]WS
/productCodeAttributeData").value
pcDict = {}
pcList = []

#We'll now grab the product code data from the tag dataset
and create the JSON data structure
#that will be passed back
for row in range(pcData.rowCount):
    if pcData.getValueAt(row, 'Product Code') == productCode o
r productCode == 'ALL':
        rowDict = {}
        for col in range(pcData.columnCount):
            rowDict[str(pcData.getColumnName(col))] = str(pcDa
ta.getValueAt(row, col))

        pcList.append(rowDict)
pcDict['productCodeAttributes'] = pcList

return {'code': 200, 'content': pcDict}

```

- Save the changes by pressing .

Script

```

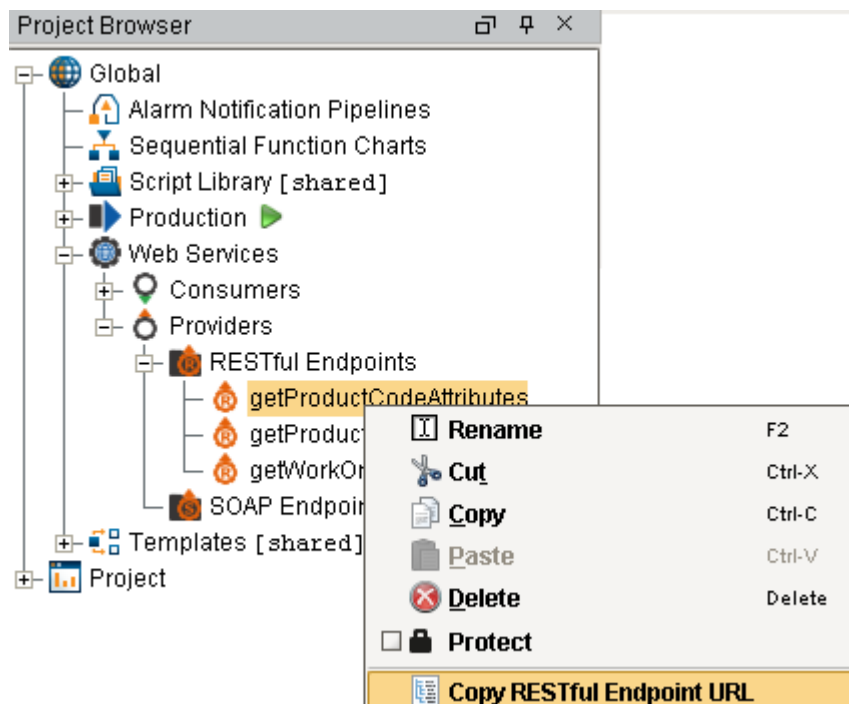
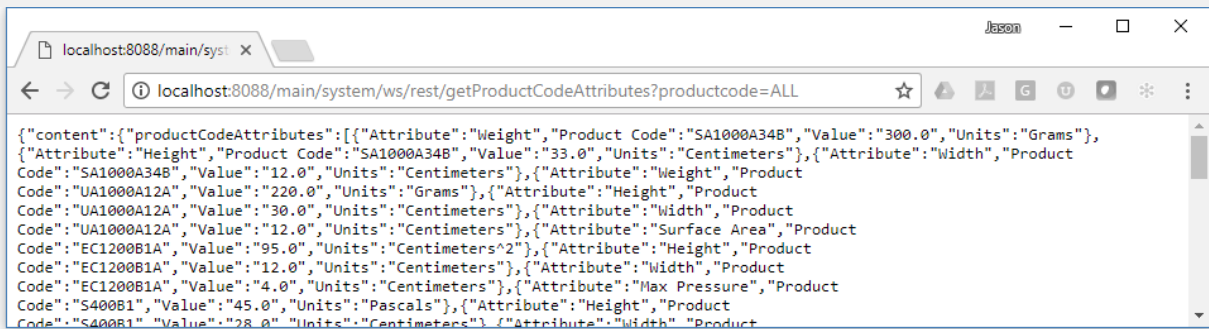
1 def do_GET(request):
2     """
3     Responds to an incoming HTTP request with a response in dict format.
4
5     Puts a HTTP status code in integer format at the key 'code' in the response
6     dict.
7
8     Puts a content in dict format at the key 'content' in the response dict.
9
10    Arguments:
11        request: A dict with an incoming HTTP request information. The path
12                parameter is contained within a string at request['path']. The query
13                parameters are contained within a dict at request['query']. The message
14                body parameter is contained within a dict at request['body']. The HTTP
15                headers are contained within a dict at request['headers'].
16    """
17    query_dict = request['query']
18
19    if query_dict.has_key('productcode') == False:
20        return {'code': 200, 'content': {'path': request['path'], 'query': request['query'], 'body': request['body']}}
21
22    #We'll only define one input parameter, a filter by product code
23    productCode = query_dict['productcode']
24
25    #Product Code attribute data is stored in a tag dataset. We can right-click on a tag and say
26    #'copy tag path' but we need to add the provider i.e. '[default]' to make it a
27    #fully qualified tag path
28    pcData = system.tag.read("[default]WS/productCodeAttributeData").value
29    pcDict = {}
30    pcList = []
31
32    #We'll now grab the product code data from the tag dataset and create the JSON data structure
33    #that will be passed back
34    for row in range(pcData.rowCount):
35        if pcData.getValueAt(row, 'Product Code') == productCode or productCode == 'ALL':
36            rowDict = {}
37            for col in range(pcData.columnCount):
38                rowDict[str(pcData.getColumnName(col))] = str(pcData.getValueAt(row, col))
39
40            pcList.append(rowDict)
41    pcDict['productCodeAttributes'] = pcList
42
43    return {'code': 200, 'content': pcDict}

```

☐ Show whitespace characters


- Right-click on **getProductCodeAttributes** in the Project Browser pane and copy the URL into your web browser.
- Add **?productcode=ALL** to the end of the URL

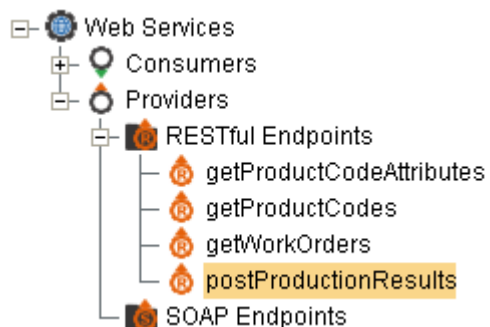
You should see the following in your browser window...



5.1.5 Create postProductionResults() Endpoint

Now we will create the endpoint for **postProductionResults()** which will allow us to update our ERP system as to work order fulfillment.

- Navigate to **Global -> Web Services -> Providers > RESTful Endpoints** in the Project Browser panel of the Ignition designer.
- Right-click and select  **New RESTful Endpoint**.
- Rename this Endpoint **postProductionResults**.



- In the **Settings** panel, set the HTTP Method to **Post**.

postProductionResults

RESTful Web Service Provider Endpoint

Settings

General

HTTP Method: POST

Data Format: JSON

Encoding: UTF-8

☐ Redirect to SSL

HTTP Authentication

Type: None

User Source: default

Required Role(s):

The code stub for how input and output from the endpoint is handled was automatically created for us. All we have to do now is modify the script to handle the input and return whatever data we want to.

- Delete the stub code in the script window (all lines after the purple comment block) and replace it with the following...

```

path_string = request['path']
query_dict = request['query']
body_dict = request['body']

beginDate = body_dict.get("actualBeginDate")
endDate = body_dict.get("actualEndDate")
actualEquipment = body_dict.get("actualEquipment")
qty = body_dict.get("actualQuantity")
status = body_dict.get("woStatus")
wo = body_dict.get("workOrder")

woData = system.tag.read("[default]WS/workOrderData").value

#We'll now grab the WO data from the tag dataset and create
the JSON data structure
#that will be passed back
for row in range(woData.rowCount):
    if woData.getValueAt(row, 'Work Order') == wo:
        # Consolidate changes into a dictionary
        updates = {"Actual Begin Date": beginDate, "Actual
End Date": endDate, "Equipment": actualEquipment, "Actual Quantity"
: qty, "Status": status}
        # Update the tag Dataset
        newData = system.dataset.updateRow(woData, row,
updates)

        system.tag.write("[default]WS/workOrderData", newData)
        break

content_dict = {'path': path_string, 'query': query_dict, 'bod
y': body_dict}
return {'code': 200, 'content': content_dict}

```

- Save the changes by pressing .


```

def do_POST(request):
    """
    Responds to an incoming HTTP request with a response in dict format.

    Puts a HTTP status code in integer format at the key 'code' in the response
    dict.

    Puts a content in dict format at the key 'content' in the response dict.

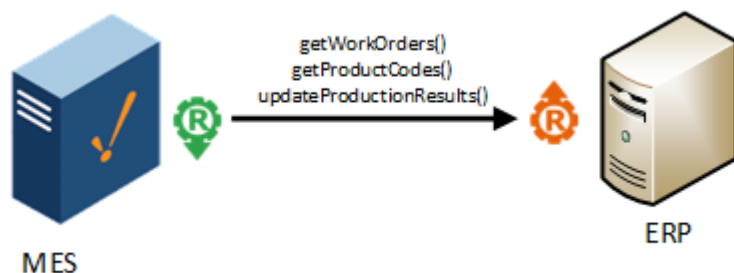
    Arguments:
        request: A dict with an incoming HTTP request information. The path
            parameter is contained within a string at request['path']. The query
            parameters are contained within a dict at request['query']. The message
            body parameter is contained within a dict at request['body']. The HTTP
            headers are contained within a dict at request['headers'].
    """
    1 path_string = request['path']
    2 query_dict = request['query']
    3 body_dict = request['body']
    4
    5 beginDate = body_dict.get("actualBeginDate")
    6 endDate = body_dict.get("actualEndDate")
    7 actualEquipment = body_dict.get("actualEquipment")
    8 qty = body_dict.get("actualQuantity")
    9 status = body_dict.get("woStatus")
    10 wo = body_dict.get("workOrder")
    11
    12
    13 # logger = system.util.getLogger("WS Logger")
    14 # logger.infof("wo is %s", wo)
    15
    16 woData = system.tag.read("[default]WS/workOrderData").value
    17
    18 #We'll now update the WO data back to the tag dataset
    19 for row in range(woData.rowCount):
    20     if woData.getValueAt(row, 'Work Order') == wo:
    21         # Consolidate changes into a dictionary
    22         updates = {"Actual Begin Date": beginDate, "Actual End Date": endDate, "Equipment"
    23         newData = system.dataset.updateRow(woData, row, updates) # Update the tag Dataset
    24         system.tag.write("[default]WS/workOrderData", newData)
    25         break
    26
    27 content_dict = {'path': path_string, 'query': query_dict, 'body': body_dict}
    28 return {'code': 200, 'content': content_dict}

```

We'll test that this endpoint is setup correctly when we create the postProductionResults() consumer.

5.2 Web Service Consumer

In this section, we will be creating the corresponding RESTful Web Service Consumer API's to use the endpoints we just created.

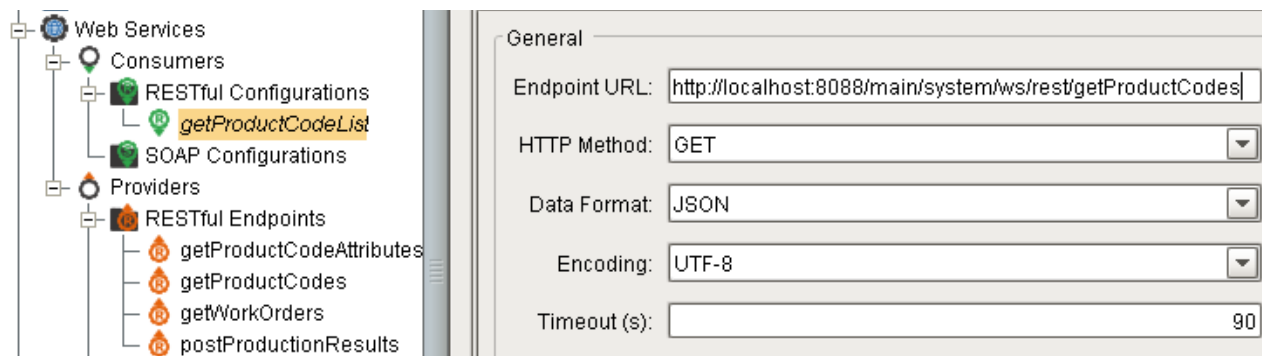



There are a number of videos on the right hand side that go through the steps of creating a RESTful consumer in case you get stuck or need more detailed information.

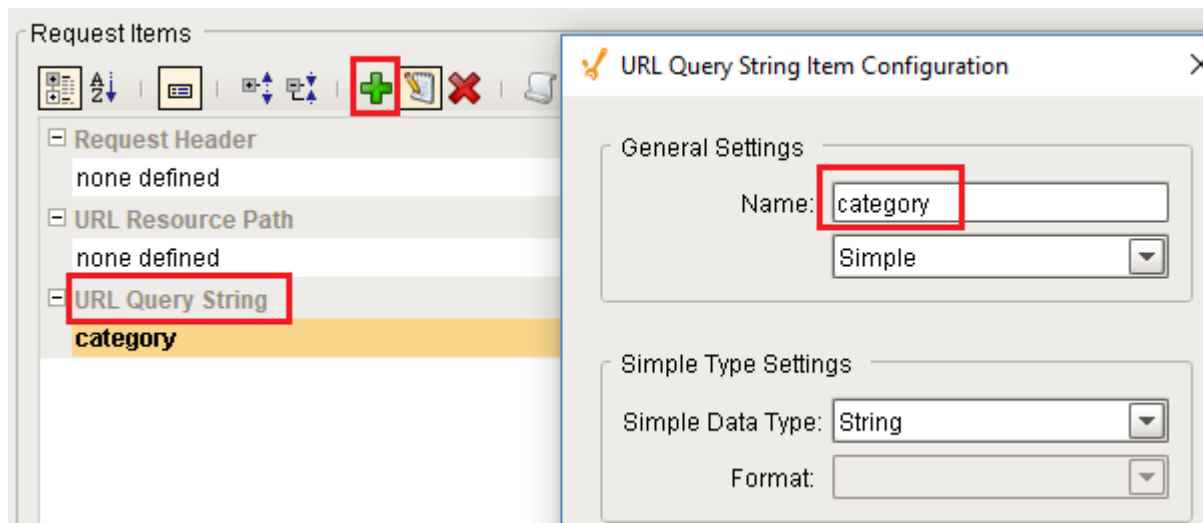
5.2.1 Get Product Code List

We will now create a **RESTful Web Service** configuration to access Product Codes from our ERP system.

- Navigate to **Global -> Web Services -> Consumers > RESTful Configurations** in the Project Browser panel of the Ignition designer.
- Right-click and select **New RESTful Configuration**.
- Rename this Web Service to **getProductCodeList**.
- Right-click on **getProductCodes** RESTful Endpoint under the Providers folder, select 'Copy RESTful Endpoint URL' and paste it into **EndPoint URL** for the **getProductCodeList** consumer.

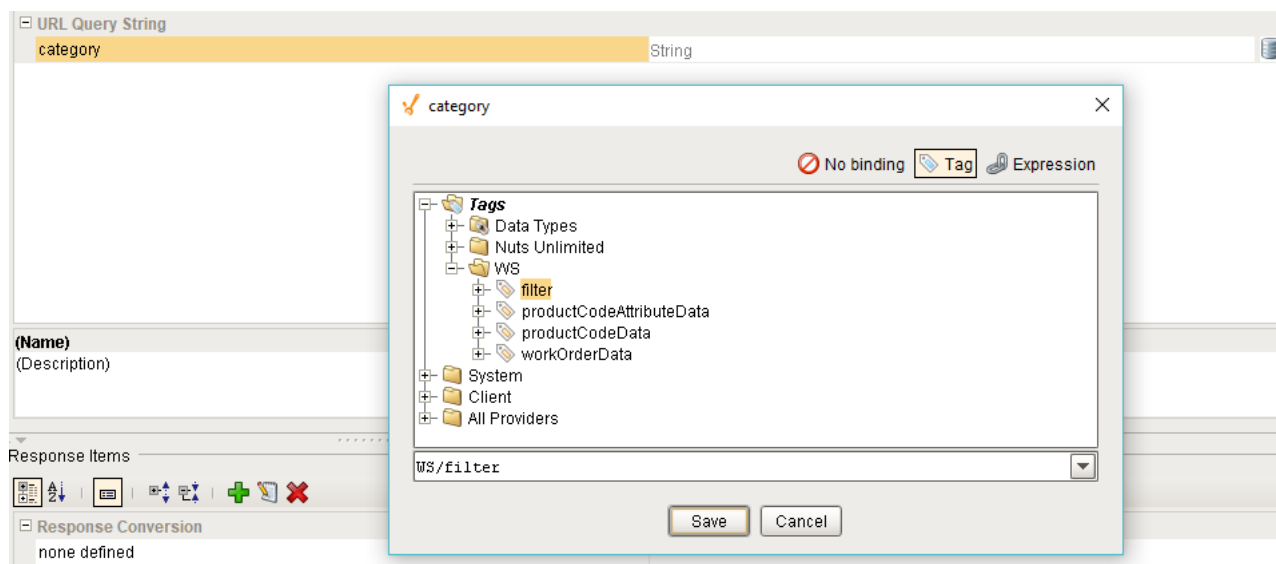


- In the **Request Items** panel, select **URL Query String** and click the  icon to add a new string.
- Specify **category** as the name, leaving all other options as default.



- Bind the **category** URL Query String to your **WS/filter** tag.
- **Save** the changes by pressing



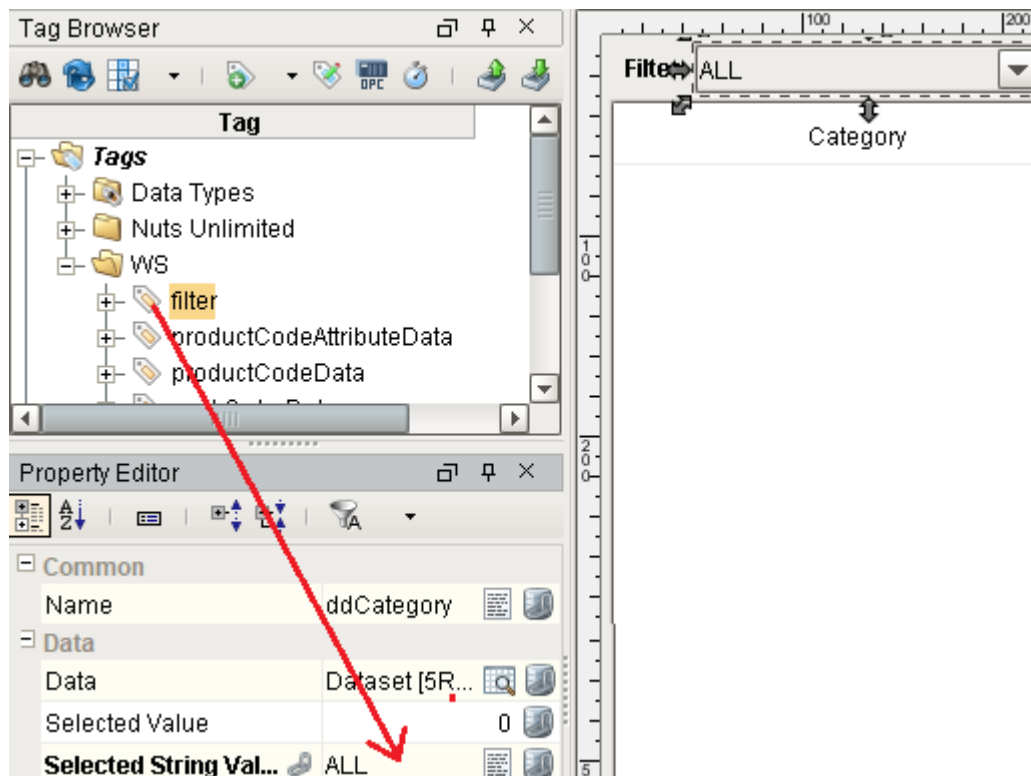


- Open the **Web Services\Product Codes** window that we imported at the beginning of the Web Services tutorial section.
- Drag the **WS/filter** tag onto the **ddCategory** dropdown **Selected String Value** property.
- Add the following script to the **Get Product Codes** Button **actionPerformed** event:

```
wsResponse = system.ws.runWebService("getProductCodeList")
data = wsResponse.getChild('content').getChild('productcodes').
toDataset()
event.source.parent.getComponent('pTableProductCodes').data =
data
```

- Go into Preview Mode (**F5**), select **ALL** in the Filter dropdown and click the **Get Product codes** button. The Power Table should fill with data.

The filter dropdown allows you to filter by Category.



So What Exactly Just Happened....

Let's examine the script to see what just happened.

| Line | Script | Description |
|------|---|--|
| 1 | <pre>wsResponse = system.ws. runWebService ('getProductCodeList')</pre> | <p>We used the system.ws.runWebService scripting function to call the getProductCodeList web service that we configured. system.ws.runWebService is an overloaded scripting function that can be called with or without parameters. In this case, we didn't need to pass any parameters as the getProductCodeList web service gets the category filter parameter from the tag binding that we defined in the RESTful consumer configuration.</p> <p>The system.ws.runWebService returned a WS Variable object containing data. This object also has built-in functions that allow us to access the data within the object. Understanding how the data is returned by the web service is important when determining how to access it.</p> |

| Line | Script | Description |
|------|---|---|
| 2 | data = wsResponse.
getChild('content').
getChild
('productcodes').
toDataset() | We then used the .getChild() function to access the productcode name /value data, which is a child itself of the content value/pair. The .getChild() also returns a WS Variable object. We then used the .toDataset() function of the WS Variable object to convert the product data into a dataset that we pass to our table. |

5.2.2 Get Product Attributes

We'll now add another consumer to obtain attributes associated with product codes. These attributes could be used to enable product codes for certain production lines and include standard and schedule rates that we could use in our OEE implementation. In this tutorial, the web service will return dimensional data for our products.

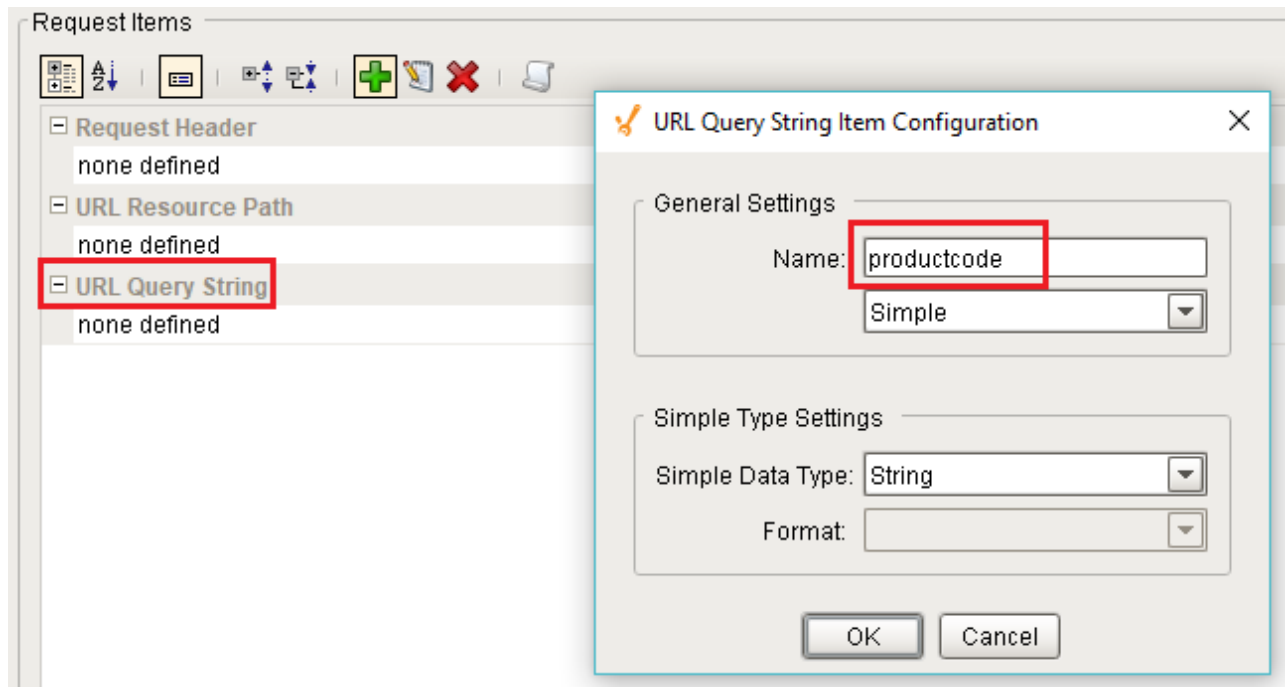
- Create a new RESTful configuration and call it **getProductCodeAttributes**.
- Right-click on the **getProductCodeAttributes** RESTful Endpoint under the Providers folder, select 'Copy RESTful Endpoint' and paste it into **EndPoint URL** for the **getProductCodeAttributes** consumer.

This time, rather than using the URL Query String/Tag binding, we are going to pass the parameters to the web service call in the script.

The screenshot shows the 'Web Services' tree on the left with 'Consumers' expanded. Under 'RESTful Configurations', 'getProductCodeAttributes' is highlighted. The right pane shows the 'RESTful Web Service Consumer Configuration' dialog for this consumer. The 'Settings' tab is active, showing the 'General' section with the following values:

- Endpoint URL: `http://localhost:8088/main/system/ws/rest/getProductCodeAttributes`
- HTTP Method: `GET`
- Data Format: `JSON`
- Encoding: `UTF-8`


- Create a URL Query String called **productcode**, but do not bind it to a tag.
- Save your Web Service Configuration.



- In the **Product Codes** window, add the following script to the Product Code Table **onMousePress** extension function:

```
productCode = self.data.getValueAt(rowIndex, 'Product
Code')
pc = {'productcode':productCode}
wsResponse = system.ws.runWebService("getProductCodeAttri
butes",pc,None,None)
data = wsResponse.getChild("content").getChild("productCo
deAttributes").toDataset()
table = self.parent.getComponent("pTablePCAttributes").
data = data
```

As you click on entries in Product Code table, you should see attributes filling your new table.

| Filter: <input type="text" value="ALL"/> | |  Get Product Codes |
|--|-----------------------------------|--|
| Category | Description | Product Code |
| Sub Assembly | Sub assembly 34B for item 1000A | SA1000A34B |
| Sub Assembly | Sub assembly 35C for item 1000A | SA1000A35C |
| Sub Assembly | Sub assembly 36D for item 1000A | SA1000A36D |
| Sub Assembly | Sub assembly 40D for item 1000A | SA1000A40D |
| Upper Assembly | Upper assembly 12A for item 1000A | UA1000A12A |
| Upper Assembly | Upper assembly 13A for item 1000A | UA1000A13A |
| Upper Assembly | Upper assembly 14A for item 1000A | UA1000A14A |
| Upper Assembly | Upper assembly 15A for item 1000A | UA1000A15A |
| End Cap | End cap 1A for item 1200B | EC1200B1A |
| End Cap | End cap 2A for item 1200B | EC1200B2A |
| End Cap | End cap 3A for item 1200B | EC1200B3A |
| Strainer | Strainer 1 for item 400 | S400B1 |
| Strainer | Strainer 2 for item 400 | S400B2 |
| Strainer | Strainer 3 for item 400 | S400B3 |

| Attribute | Product Code | Value | Units |
|--------------|--------------|-------|---------------|
| Surface Area | EC1200B1A | 95.0 | Centimeters^2 |
| Height | EC1200B1A | 12.0 | Centimeters |
| Width | EC1200B1A | 4.0 | Centimeters |

So What Exactly Just Happened....

Let's examine the script to see what just happened.

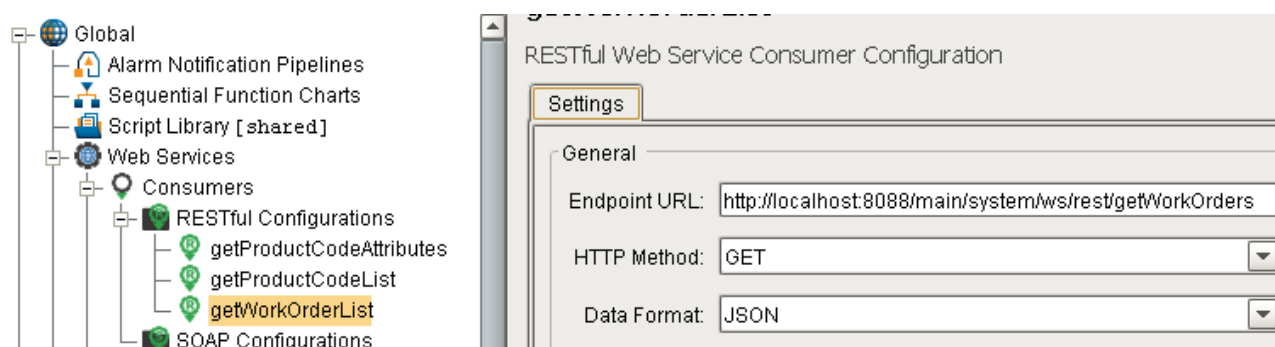
| Line | Script | Description |
|------|--|--|
| 2 | pc = {'productcode':
productCode} | We built a python dictionary that we will pass to the request header of the web service call. |
| 3 | wsResponse = system.ws.runWebService
("getProductCodeAttributes",
pc,None,None) | We used the system.ws.runWebService scripting function to call the getProductCodeAttributes web service that we configured. system.ws.runWebService is an overloaded scripting function that can be called with or without parameters. In this case, we passed the pyDictionary pc = {'productCode':productCode} as the urlParam argument, leaving the headersObject and bodyObject parameters empty. |

| Line | Script | Description |
|------|--|--|
| | | <p>The <code>system.ws.runWebService</code> returned a WS Variable object containing data. This object also has built-in functions that allow us to access the data within the object. Understanding how the data is returned by the web service is important when determining how to access it. If we added print wsResponse to the onMousePressed event, we could see the format of the data returned in the client console which looks as shown.</p> <p>If we added a print wsResponse to our mouse pressed event, we could see the structure of the returned data to determine how to access the stuff we are interested in.</p> <pre> { "Root": { "attributes": [{ "AttributeName": "Weight", "AttributeValue": "250", "ProductCode": "SA1000A40D", "Units": "Grams" }, { "AttributeName": "Height", "AttributeValue": "28", "ProductCode": "SA1000A40D", "Units": "Centimeters" }, { "AttributeName": "Width", "AttributeValue": "10", "ProductCode": "SA1000A40D", "Units": "Centimeters" }] } } </pre> |
| 4 | data = wsResponse.getChild("content").getChild("productCodeAttributes").toDataset() | <p>We then used the .getChild() function to drill down and access the attributes name/value data which itself returns a WS Variable object. We used the .toDataset() function of the WS Variable object to convert the data into a dataset that we could then pass to our table component.</p> |

5.2.3 Get Work Order List

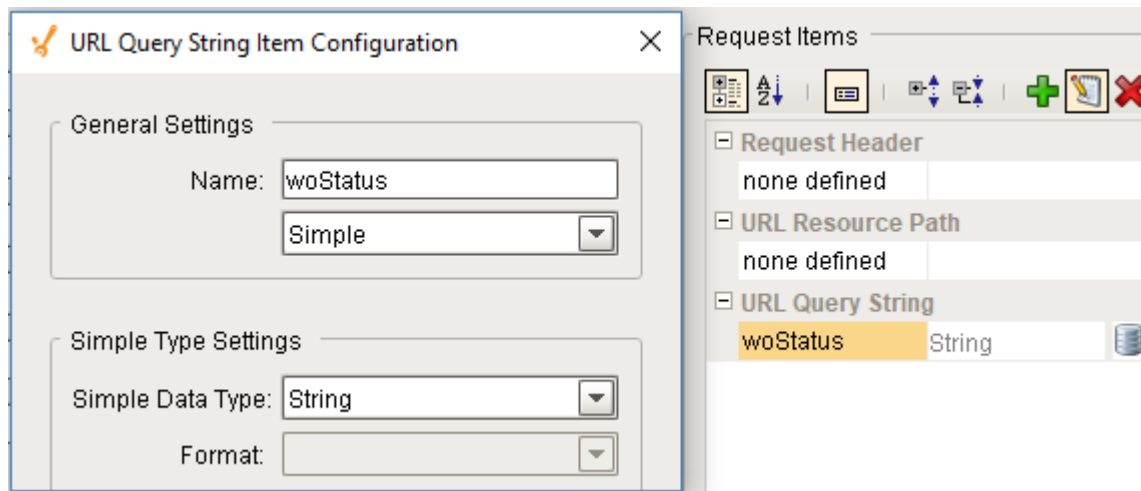
Now that we have product codes taken care of, we will create an interface to pull work orders from our ERP.

- Create a new RESTful Configuration called **getWorkOrderList**
- Right-click on the **getWorkOrders** RESTful Endpoint under the Providers folder, select 'Copy RESTful Endpoint URL' and paste it into **EndPoint URL** for the **getWorkOrderList** consumer.



- Add the Query string **woStatus**.
- **Save** the changes by pressing





- Open the **Web Services\Work Orders** that we imported at the beginning of the Web Services tutorial.
- Copy the code below to your **Get Work Orders** button's **actionPerformed** event handler:

```
woStatus = event.source.parent.getComponent('ddWoStatus').
selectedLabel
table = event.source.parent.getComponent('pTableWO')
input = {'woStatus':woStatus}
wsResponse = system.ws.runWebService('getWorkOrderList',input,None
,None)
table.data = wsResponse.getChild('content').getChild('workorders')
.toDataset()
event.source.parent.parent.getComponent('Text Area').text =
wsResponse.toString()
```

- Go into Preview mode by pressing **F5** and select 'ALL' from the Status dropdown.
- Click the button to pull works orders down from ERP.

If you did this right, your power table should populate with work order information. The text area will populate with the JSON data, so that you can see how the data is formatted.

| Get Work Orders | | | | | | | | | | |
|-----------------|-------------|-------------|-----------|----------|--------------|---------------|-------------|--------------|----------|---------------|
| Status | Actual Q... | Work Ord... | Equipment | Priority | Actual En... | Special I... | Product ... | Actual Be... | Quantity | Due Date |
| None | 100.0 | WO-0001 | Press 3 | 10 | Sun Oct 0... | Handle C... | UA1000A1... | Sun Oct 0... | 1000.0 | Thu Oct 2... |
| RELEASED | None | WO-0002 | | 0 | None | | EC1200B1A | None | 500.0 | Sat Oct 28... |
| HOLD | None | WO-0003 | | 0 | None | Waiting on... | S400B1 | None | 5000.0 | Mon Oct 2... |

Status:

5.2.4 Post Production Results

The next interface that we'll create will be to post production data back to the ERP system. As product is made and work orders are completed, we can use this to update ERP of order fulfillment and work order completion.

- Create a new RESTful configuration and call it **postProductionResults**.
- Right-click on the **postProductionResults** RESTful Endpoint under the Providers folder, select 'Copy RESTful Endpoint' and paste it into **EndPoint URL** for the **postProductionResults** consumer.
- Set the HTTP Method to **POST**.
- Add a new Request Header Item and name it **Content-type** with a value of **application /json**.
- **Save your changes**.



- Open the **Web Services\Work Orders** window and add the following script to the **actionPerformed** event of the **Update Production** button.

```
actualBeginDate = event.source.parent.getComponent('calBegin').
date
actualBeginDate = system.date.format(actualBeginDate,"yyyy-MM-dd
H:mm:ss")
actualEndDate = event.source.parent.getComponent('calEnd').date
actualEndDate = system.date.format(actualEndDate,"yyyy-MM-dd H:mm:
ss")
actualEquipment = event.source.parent.getComponent('txtEquipment')
.text
workOrder = event.source.parent.getComponent('txtWorkOrder').text
actualQuantity = event.source.parent.getComponent('numQty').
intValue
woStatus = event.source.parent.getComponent('ddWoStatus').
selectedLabel
jsonObj = {}
jsonObj['actualBeginDate'] = actualBeginDate
jsonObj['actualEndDate'] = actualEndDate
jsonObj['actualEquipment'] = actualEquipment
jsonObj['workOrder'] = workOrder
jsonObj['actualQuantity'] = actualQuantity
jsonObj['woStatus'] = woStatus
result = system.ws.runWebService('postProductionResults',
None,
{
  'Content-type': "application/json"
},
jsonObj)
event.source.parent.parent.getComponent('Text Area').text =
result.toString()
```

- Go into Preview mode by pressing **F5** and click on any work order in the table to auto-populate the work order field in the 'Update Production Data' panel.
- Enter a produced quantity, actual begin and end date, equipment used and new status for the selected Work Order.
- Press the **Update Production** button.

Get Work Orders

| Status | Actual Q... | Work Ord... | Equipment | Priority | Actual En... | Special I... | Proc |
|------------|-------------|-------------|-----------|----------|----------------|---------------|------|
| IN PROCESS | 200.0 | WO-0001 | Line 2 | 10 | Fri Oct 27 ... | Handle C... | UA10 |
| COMPLETED | 450.0 | WO-0002 | Line 1 | 0 | Fri Oct 27 ... | | EC12 |
| HOLD | None | WO-0003 | | 0 | None | Waiting on... | S400 |

Status:

Update Production Data

Work Order:

Begin Date:

End Date:

Equipment:

Quantity:

Status:

```

{"Root": {"content": {
  "body": {
    "actualBeginDate": "2017-10-26 17:24:53",
    "actualEndDate": "2017-10-27 17:24:53",
    "actualEquipment": "Line 2",
    "actualQuantity": 300,
    "woStatus": "HOLD",
    "workOrder": "WO-0001"
  },
  "path": "",
  "query": null
}}}

```

If everything went according to plan, we just updated our ERP system via our web service interface and our text area shows the response from the web service provider, which in this case is simply returning that data that we sent.

- Press the **Get Work Orders** button to retrieve the posted work order results and update our Work Order table.

So What Did We Do Here....

Let's examine the script to see what just happened.

| Line | Script | Description |
|-------|---|--|
| 9-15 | <pre> jsonObj = {} jsonObj['actualBeginDate'] = actualBeginDate jsonObj['actualEndDate'] = actualEndDate jsonObj['actualEquipment'] = actualEquipment jsonObj['workOrder'] = workOrder jsonObj['actualQuantity'] = actualQuantity jsonObj['woStatus'] = woStatus </pre> | We created a python dictionary to hold our name/value pairs |
| 16-21 | <pre> result = system.ws. runWebService ('postProductionResults', None, { 'Content-type': "application /json" }, jsonObj) </pre> | We then passed our python dictionary called jsonObj as the bodyObject parameter to the webService call |

5.3 Web Service Review

This concludes the Web services portion of the tutorial. In the next section we shall learn about the Recipe module and continue the Nuts Unlimited project by adding Recipe Management to the packaging line.

5.3.1 What We Covered

In summary, we covered the following:

- [Web Services Overview](#)
- [Creating RESTful Configurations for Providers and Consumers](#)
- [Getting and Posting data via web services](#)
- [Using the `system.ws.runWebService` scripting function](#)
- [Using the WS Variable object](#)

5.3.2 What We Didn't Cover

What we haven't covered or touched in great detail are:

- [SOAP Configuration for Providers and Consumers](#)
- [WS Security](#)
- [Using data from Web Services to create Work Order Objects, Material Objects, Material Production settings and Scheduling Production through scripting](#)

5.3.3 Additional Resources

We can't cover everything in a tutorial, but we do have more resources available in the [MES Help manual](#), [Sepasoft Knowledge Base](#) and [Videos](#) for the areas we didn't cover.

» RECIPE MANAGEMENT

- › Configure Recipe Tags
- › Add Recipe Tags to Production Model
- › Create Process Recipes
- › Viewing Recipe Changes
- › Test Recipes with the Recipe Selector
- › Viewing Recipe Variances
- › Add Real Time Recipe Monitoring
- › Add Recipe Selection to the Operations Segment
- › Using Analysis to Obtain Recipe Information
- › Recipe Review

6 Recipe Management

In this section, we will add the ability to create recipes that hold machine settings for the packaging line equipment. We will then download these settings to the line and monitor machine setpoint variance from actual.

The Recipe Management module provides support for managing machine settings for lines, cell groups, cells and location production items. Recipe values can be added to these production items and represent settings that may be written to a PLC or other controller via Ignition tags. The recipe values that are added at the line level are propagated down to production cells, cell groups and locations within the line. This provides a quick method for adding recipe values that are common to all machines (cells) within a line. It also allows for the ability to propagate a recipe value down to all production items within an area.

We have created a couple of templates and popup windows that we'll use during the recipe training that will speed up the development effort.

-  Download [MES_2.0_Training_Recipe_Base.proj](#) and import into your project.
- **Save** your project.

✓ Generally location production items would only be used by recipe when other production items are not available due to not having OEE or T&T modules installed or licensed.

6.1 Configure Recipe Tags

To add recipe functionality to the packaging line, we'll need some tags that we'll write machine setting values to. To keep it simple, memory tags will be used so we can manipulate them to test variance tracking. To make it quicker, we have provided a script to create the tags. You could create the tags yourself, but we have some templates later on that are expecting this structure.

- Copy the script below into the script console to create the necessary tag structure.

```
#You may need to change the pPath below if your tag structure is different
```

```

pPath = 'Nuts Unlimited/Folsom/Packaging/Packaging Line 1
/plc_PackagingLine1'
if not system.tag.exists(pPath):
    print pPath + " folder not found. \nPlease copy your folder
tag path to the line above."
else:
    tags = []
    tags.append(['/CasePacker', 'Case Size', 'MEMORY', 'Int2', 25]
)
    tags.append(['/Checkweigher', 'Max Weight', 'MEMORY', 'Float4'
,12])
    tags.append(['/Checkweigher', 'Min Weight', 'MEMORY', 'Float4'
,10])
    tags.append(['/Checkweigher', 'Target Weight', 'MEMORY', 'Floa
t4',11])
    tags.append(['/Filler', 'Container Size', 'MEMORY', 'Float4',2
2.4])
    tags.append(['/Filler', 'Speed', 'MEMORY', 'Float4',100.0])
    tags.append(['/Palletizer', 'Pattern Code', 'MEMORY', 'Float4'
, 45])
    tags.append(['', 'Dummy', 'MEMORY', 'Boolean',False])

    for item in tags:
        if not system.tag.exists(pPath + item[0] + "/" + item[1]):
            print "adding tag " + item[1]
            system.tag.addTag(parentPath=pPath + item[0], name=item
m[1], tagType=item[2], dataType=item[3], value=item[4])
        else:
            print item[1] + " already exists"

```

- From the tag browser, copy the tag path to your *plcPackagingLine1* folder into the first line of the script.
- **Execute** the script.

Your tag browser should look as shown.

| Tag Path | Tag Name | Tag Type | Data Type |
|---|-----------|----------|-----------|
| Nuts Unlimited/Folsom/Packaging/Packaging Line 1
/plc_PackagingLine/ CasePacker | Case Size | Memory | Int2 |
| | | Memory | Float4 |

| Tag Path | Tag Name | Tag Type | Data Type |
|---|----------------|----------|-----------|
| Nuts Unlimited/Folsom/Packaging/Packaging Line 1
/plc_PackagingLine/ Checkweigher | Max Weight | | |
| Nuts Unlimited/Folsom/Packaging/Packaging Line 1
/plc_PackagingLine/ Checkweigher | Min Weight | Memory | Float4 |
| Nuts Unlimited/Folsom/Packaging/Packaging Line 1
/plc_PackagingLine/ Checkweigher | Target Weight | Memory | Float4 |
| Nuts Unlimited/Folsom/Packaging/Packaging Line 1
/plc_PackagingLine/ Filler | Container Size | Memory | Float4 |
| Nuts Unlimited/Folsom/Packaging/Packaging Line 1
/plc_PackagingLine/ Filler | Speed | Memory | Float4 |
| Nuts Unlimited/Folsom/Packaging/Packaging Line 1
/plc_PackagingLine/ Palletizer | Pattern Code | Memory | Float4 |
| Nuts Unlimited/Folsom/Packaging/Packaging Line 1
/plc_PackagingLine | Dummy | Memory | Boolean |

| Tag | Data Type |
|--------------------|-----------|
| Tags | |
| Data Types | |
| Nuts Unlimited | |
| Folsom | |
| Mixing | |
| Packaging | |
| Packaging Line 1 | |
| plc_PackagingLine1 | |
| CasePacker | |
| Case Size | Int2 |
| Infeed | Int4 |
| Outfeed | Int4 |
| State | Int4 |
| Waste | Int4 |
| Checkweigher | |
| Infeed | Int4 |
| Max Weight | Float4 |
| Min Weight | Float4 |
| Outfeed | Int4 |
| State | Int4 |
| Target Weight | Float4 |
| Waste | Int4 |
| Filler | |
| Container Size | Float4 |
| Infeed | Int4 |
| Outfeed | Int4 |
| Speed | Float4 |
| State | Int4 |
| Waste | Int4 |
| Palletizer | |
| Infeed | Int4 |
| Outfeed | Int4 |
| Pattern Code | Float4 |
| State | Int4 |
| Waste | Int4 |
| Cardboard Vendor | String |
| DOW | Int4 |
| Dummy | Boolean |

6.2 Add Recipe Tags to Production Model

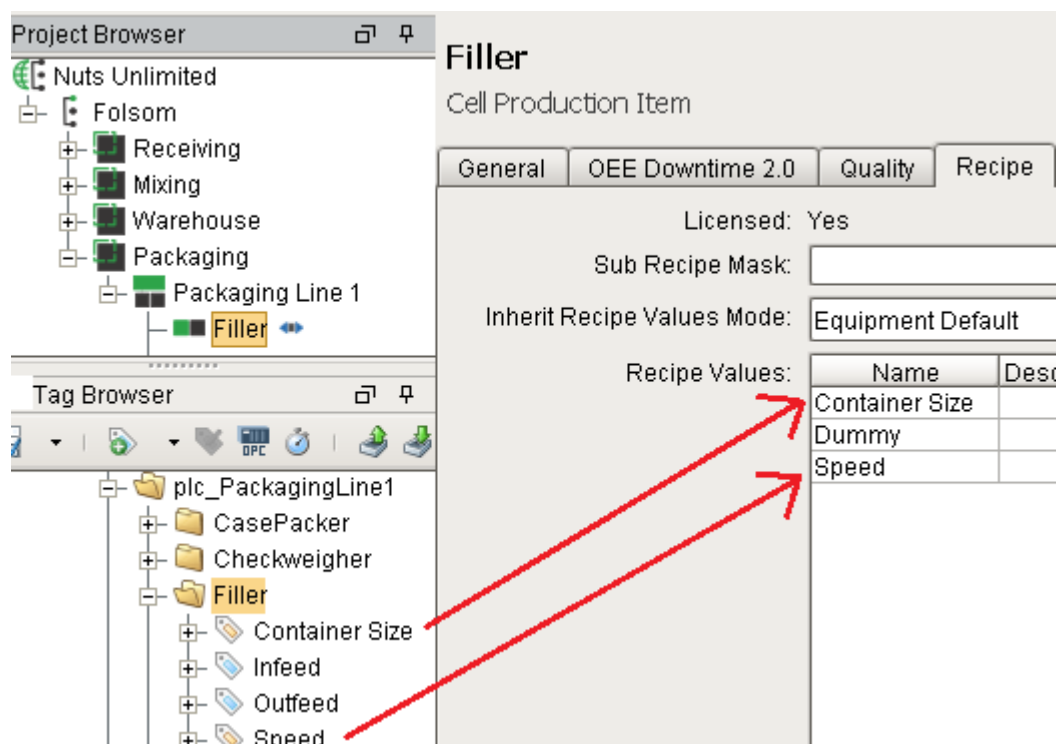
Recipe value parameters are created and bound to Ignition tags in the Recipe tab of the Production model. In this section we will create the recipe value parameters for machine setpoints that we want to control on Packaging Line 1. If you do not see the recipe tab, you will need to install the recipe module.

To add recipe values for the tags we just added in the previous section, using the following steps:

- In the Production Model, click on **Packaging Line 1**, select the **Recipe** tab and drag the **Dummy** tag over from *Nuts Unlimited/Folsom/Packaging/Packaging Line 1/plc_PackagingLine*.

Why did we do this? Well if we want to use the recipe selector component to select a recipe for the Packaging line, we have to add the packaging line as a production item to the recipe, and this is only possible if the packaging line has a recipe value.

- Click on the **Filler** cell and drag the Filler **Container Size** and **Speed** tags to the **Recipe Values** pane.
- Click on the **Checkweigher** cell and drag the Checkweigher **Max Weight**, **Min Weight** and **Target Weight** tags to the **Recipe Values** pane.
- Click on the **Casepacker** cell and drag the Casepacker **Case Size** tag to the **Recipe Values** pane.
- Click on the **Palletizer** cell and drag the Palletizer **Pattern Code** tag to the Recipe Values pane.
- **Save** the changes made to the production model.



We have now made an association between the Packaging Line 1 machine setting tags and our production model. This will allow us to create recipes that can store, download and monitor these setpoints during a production run.

6.3 Create Process Recipes

We will now use the [Recipe Editor](#) component to create some recipes for the packaging line. We have already downloaded a window that has this component on it and it has been extended with user menu items to view recipe changes and copy recipes.

- Open the **Recipe Editor** window in the *Recipe* folder.
- Go into preview mode by pressing **F5**.
- Add a new recipe by right clicking on the Recipes node and selecting **Add Recipe**.
- Enter **Mixed Nuts 8oz** for the recipe name and click **OK**.
- Right-click on the **Mixed Nuts 8oz** node and choose **Select Production Items** to associate the following equipment with this recipe: **Packaging Line 1, Casepacker, Checkweigher, Filler, Palletizer**.
- Add a new recipe called **Mixed Nuts 16oz** and click **OK**.

- Right-click on the **Mixed Nuts 16oz** node and choose **Select Production Items** to associate the following equipment with this recipe: **Packaging Line 1, Casepacker, Checkweigher, Filler, Palletizer**.

- Browse down through the **Mixed Nuts 8oz** recipe to the cells under Packaging Line 1 and enter values for each of the machine settings (tags).
- Do the same for the **Mixed Nuts 16oz** recipe and enter different values.

6.4 Viewing Recipes Changes

At some stage, we'll want to see what changes were made to a recipe, by whom and when. The Recipe Editor window has already been customized to add a user menu item called **Change Log**.

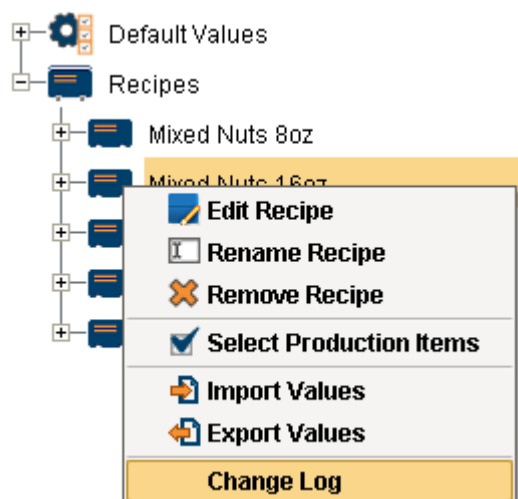
- Go into preview mode by pressing **F5**.
- Right-click on a recipe and select 'Change Log'.

The table below the recipe editor should populate with all changes made to the recipe.


- Click on a table row to see more information about each specific change.

| ItemPath | Info | Date and Time | Details |
|---|------------------------------|-------------------|--|
| Nuts Unlimited\Folsom\Packaging\Packaging ... | ValueName=Pattern Code | 2/4/2018 11:14 AM | Recipe: Mixed Nuts 16oz
Item: Nuts Unlimited\Folsom\Packaging\Packaging Line 1\Filler
Value Name: Container Size
From Value:
To Value: 12.5
Changed By: admin
Note: |
| Nuts Unlimited\Folsom\Packaging\Packaging ... | ValueName=Container Size | 2/4/2018 11:14 AM | |
| Nuts Unlimited\Folsom\Packaging\Packaging ... | ValueName=Speed | 2/4/2018 11:13 AM | |
| Nuts Unlimited\Folsom\Packaging\Packaging ... | ValueName=Target Weight | 2/4/2018 11:13 AM | |
| Nuts Unlimited\Folsom\Packaging\Packaging ... | ValueName=Min Weight | 2/4/2018 11:13 AM | |
| Nuts Unlimited\Folsom\Packaging\Packaging ... | ValueName=Max Weight | 2/4/2018 11:13 AM | |
| Nuts Unlimited\Folsom\Packaging\Packaging ... | ValueName=Case Size | 2/4/2018 11:13 AM | |
| Nuts Unlimited\Folsom\Packaging\Packaging ... | Added item path: Nuts Unl... | 2/4/2018 11:13 AM | |
| Nuts Unlimited\Folsom\Packaging\Packaging ... | Added item path: Nuts Unl... | 2/4/2018 11:13 AM | |
| Nuts Unlimited\Folsom\Packaging\Packaging ... | Added item path: Nuts Unl... | 2/4/2018 11:13 AM | |

☐ Show Default Value Changes



How did we do this?

We added a custom menu item to the **User Menu Items** property of the Recipe Editor. Click on the Dataset Viewer  icon to see or add to the options.

*This menu item will only show up when we click on a **Recipe** node of the Recipe Editor.*

Dataset Viewer

| MenuType | MenuName | MenuIcon |
|----------|------------|----------|
| Recipe | Change Log | |

We then added a script to the Recipe Editor **menu/userMenuItemClicked** event:

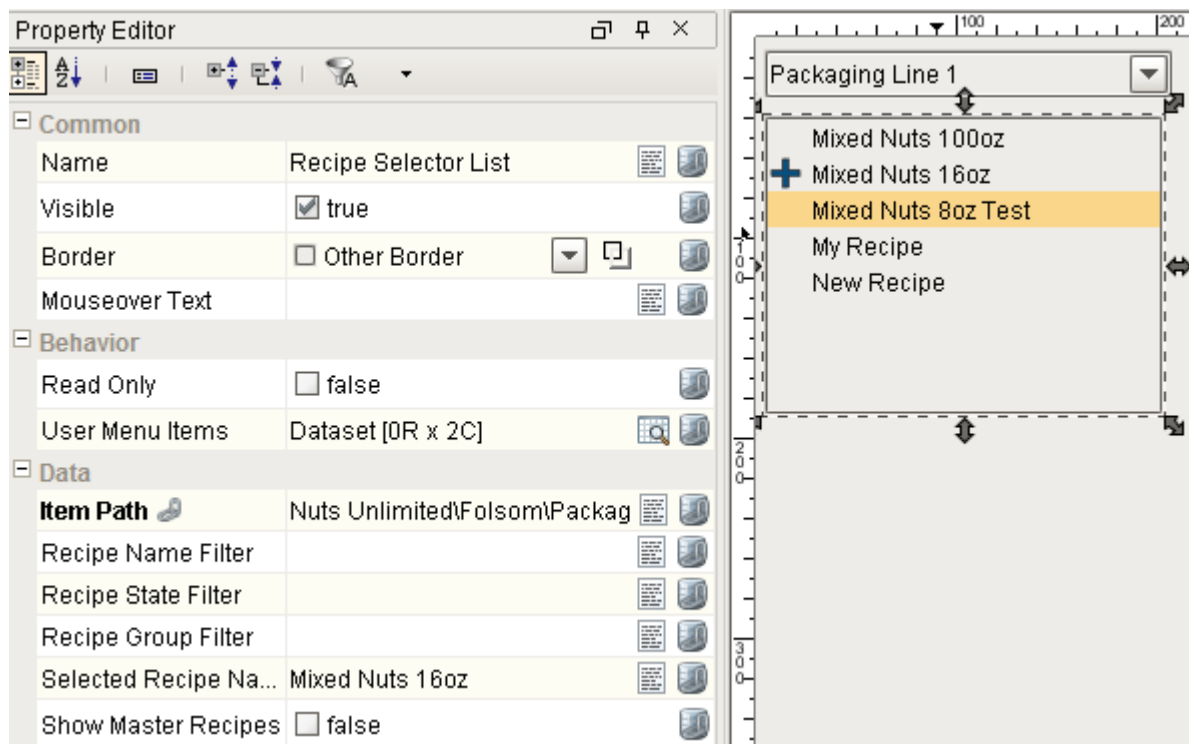
```
menuItem = event.getMenuItemName()
if menuItem == 'Change Log':
    recipeName = event.getSelectedRecipe()
    valueName = event.getSelectedValueName()
    itemPath = event.getSelectedItemPath()
    event.source.parent.getComponent('Recipe Changelog Viewer').
recipeNameFilter = recipeName
    event.source.parent.getComponent('Recipe Changelog Viewer').
recipeValueNameFilter = valueName
    event.source.parent.getComponent('Recipe Changelog Viewer').
itemPathFilter = itemPath
```

6.5 Test Recipes with the Recipe Selector

We've just configured the recipe tags, values, added a recipe editor window and created some recipes. Now we can move on to testing the implementation.

- Add a new window called **Recipe Control** under the **Recipe** folder.
- Drag an **MES Object Selector** from the Production component palette onto the window and enable *Include MES Line Objects*.
- Drag a **Recipe Selector List** from the Recipe component palette onto the window and bind its **Item Path** property to the MES Object Selector **equipmentItemPath**.
- Go into preview mode by pressing F5 and select **Packaging Line 1**.

*The recipes you have created and associated with the Packaging line 1 should now appear. If no recipes show up, check that you have enabled those recipes for the Packaging Line 1 production item in the Recipe Editor and also make sure that you have a recipe value at the Packaging Line 1 level in the Recipe tab in the Production Model Designer. Even if you don't have recipe tag values for the line, you still need to create at least one recipe value at the line level for recipes to show up at the line level. In this case, if no recipe values will be inherited for cells under the line, create a recipe value called **Dummy** with no ignition tag associated.*

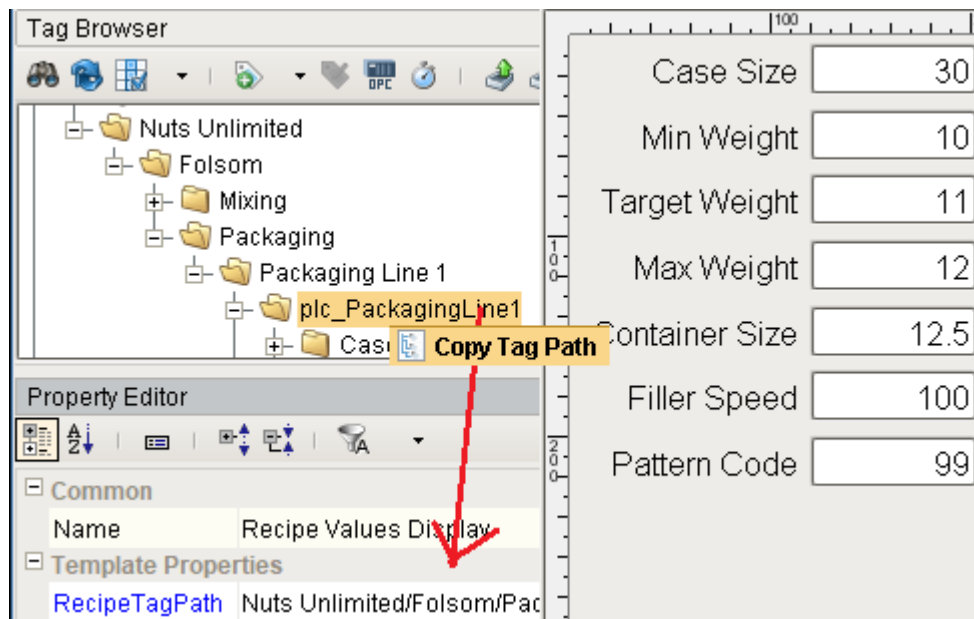


We could select the recipe we want now by right-clicking on it and choosing **Select Recipe** and monitor the ignition tags that we associated with the recipe values, but let's add a display that will also allow us to change those values so we can see recipe variance monitoring in action.

- Drag the **Recipe Values Display** template from the **Recipe** folder under **Templates** onto the screen.
- Copy the Tag Path for the **plc_PackagingLine1** folder in the Tag Browser and paste it into **RecipeTagPath** property of the **Recipe Values Display** template.

The Recipe Values template simply provides an already built display of labels and numTextfields indirectly bound to the recipe tags via the **RecipeTagPath** template parameters. This is a bi-directional binding so that we can view and adjust the recipe values. We could have simply dragged to the ignition tags onto the screen, but this template will just make this exercise quicker.

- Go ahead now and select different recipes and verify that the recipe values change.



6.6 Viewing Recipe Variances

When a recipe is selected, and the machine settings are downloaded to the ignition tags, the recipe module automatically tracks any changes to the recipe tag values. In this section, we will add the [Recipe Variance Viewer](#) component so that we can display any variances.

- Drag the **Recipe Variance Viewer** from the Recipe component palette onto your Recipe Control window.
- Bind the **Item Path Filter** property to the MES Object Selector.**equipmentItemPath**.
- Bind the **Recipe Name Filter** property to the Recipe Selector List **selectedRecipeName** property.
- Set **Show Full Details** to false.
- In the Recipe Variance Viewer **Customizers/Table Customizer**, Hide RecipeName, TrackingUUID, Units and Description columns.
- Select a recipe and start changing some of the values using the Recipe Values Display template we added to the window earlier.

You should start to see the Variance Viewer populate with information on the variances recorded.

Packaging Line 1

Mixed Nuts 100oz
Mixed Nuts 16oz
+ Mixed Nuts 8oz Test
My Recipe
New Recipe


Line Speed
Case Size
Min Weight
Target Weight
Max Weight
Container Size
Filler Speed
Pattern Code

| ItemPath | RecipeValue... | Recip... | FromVa... | ToValue | TimeStamp ▾ |
|---|----------------|----------|-----------|---------|----------------------|
| Nuts Unlimited\Folsom\Packaging\Packaging Line 1\Checkweigher | Target Weight | 0.0 | 10.0 | 20.0 | May 15, 2017 3:00 PM |
| Nuts Unlimited\Folsom\Packaging\Packaging Line 1\Checkweigher | Target Weight | 0.0 | 0.0 | 10.0 | May 15, 2017 3:00 PM |
| Nuts Unlimited\Folsom\Packaging\Packaging Line 1\Checkweigher | Target Weight | 0.0 | 15.0 | 20.0 | May 15, 2017 2:59 PM |
| Nuts Unlimited\Folsom\Packaging\Packaging Line 1\Checkweigher | Target Weight | 0.0 | 10.0 | 15.0 | May 15, 2017 2:59 PM |
| Nuts Unlimited\Folsom\Packaging\Packaging Line 1\Checkweigher | Target Weight | 0.0 | 5.0 | 10.0 | May 15, 2017 2:59 PM |
| Nuts Unlimited\Folsom\Packaging\Packaging Line 1\Checkweigher | Target Weight | 0.0 | 0.0 | 5.0 | May 15, 2017 2:59 PM |
| Nuts Unlimited\Folsom\Packaging\Packaging Line 1\Checkweigher | Target Weight | 0.0 | 0.0 | 40.0 | May 15, 2017 2:59 PM |

There are many ways that variance monitoring can be customized. When the recipe values were created in the Designer, we could have specified whether to Enable Variance Logging and set threshold values to prevent recording minor fluctuations to the recipe value. We could also have added an Evaluate variance script which would have allowed us to set the conditions that would cause a 'Variance Exists' exception.

6.7 Add Real Time Recipe Monitoring

The Recipe module automatically provides a number of OPC Production tags that you can use in your project to display real-time recipe information as well as alert and notify you to any recipe variances as they occur. In this section, we will add these tags and use them to set an alarm when a recipe value is adjusted outside of its allowable range.

- Create a folder called **Recipe\Filler** under **Nuts Unlimited\Folsom\Packaging\Packaging Line 1** in the Tag Browser.
- Click on **Tags** in the Tag Browser and select the  icon to browse OPC Servers.
- In the **Production** OPC-UA Server, browse down to **[global]\Nuts Unlimited\Folsom\Packaging\Packaging Line 1\Filler** and select the following tags by holding the **CTRL** key down and selecting:
 - ActiveRecipeName, EnableRecipe, ProductCodeMask, RecipeActive, RecipeLoading, RecipeScale, RecipeTrackingUUID, RecipeVarianceExists, RecipeWriteError, ValueMonitorEnabled.
- Drag these tags over to the folder we created before called **Recipe\Filler**.

For this example, we are only pulling OPC production recipe tags for the Packaging Line 1\Filler production item. These tags also exist for the packaging line and each cell under the packaging line.



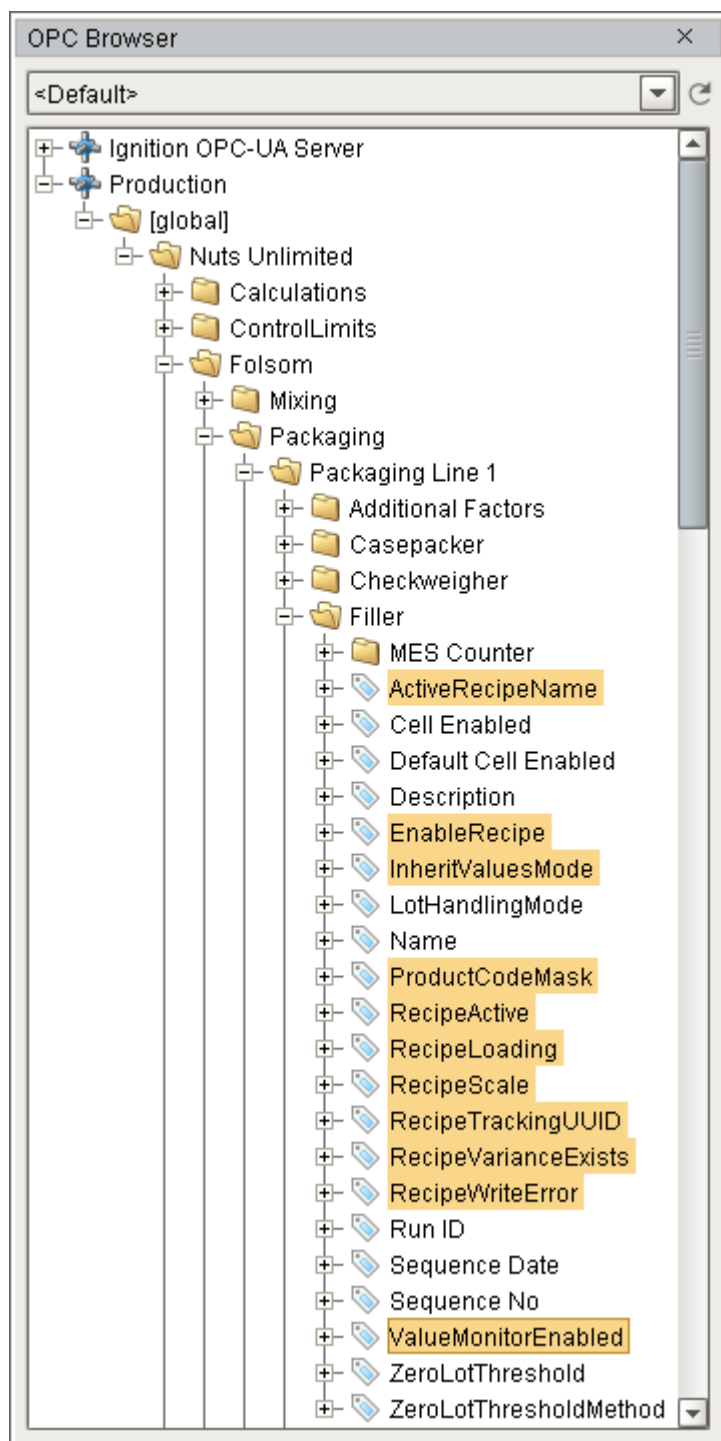
If we were smart here, we would create a tagUDT that would have indirect binding to these tags and would be passed an equipment path, or we could add these tags to the eqOEE UDT we created in the OEE section. This would allow us to quickly access these tags for all production items.

- Drag the **ActiveRecipeName** tag onto the **Recipe Control** window and display it as a **label**.
- Drag the **RecipeVarianceExists** tag onto the **Recipe Control** window and display it as a **Multi State Indicator**.
- Right-click on the Multi State Indicator and select **Customizer/Style Customizer**:
 - Delete Value 2 and 3.
 - Change background color of value 0 to green and change the Text to **OK**.
 - Change background color of value 1 to orange and change the Text to **Variance Exists**.
 - Enable animate and add a second step with a grey background. Make both steps duration equal to 1000ms.
- Select a recipe and change the Filler Speed value.

You should see the recipe name you have selected and a flashing **Variance Exists** message.

Mixed Nuts 16oz

Variance Exists



6.7.1 Add Alarming

We can use the built-in functionality of the Ignition alarm management to generate an alarm whenever we see a recipe variance occur.

- We have now setup this tag to generate an alarm whenever **RecipeVarianceExists** goes true.*

- As you change the value of the Line Speed recipe value from the recipe setpoint to a different value and back, you will see a corresponding alarm generated and then automatically cleared.*

| <input type="checkbox"/> | Active Time | Display Path | Current State | Priority |
|--------------------------|-----------------|---|-------------------------|----------|
| <input type="checkbox"/> | 5/15/17 3:58 PM | Nuts Unlimited\Folsom\Packaging\Recipe\RecipeVarianceExists\Alarm | Active, Unacknowledged | Low |
| <input type="checkbox"/> | 5/15/17 3:58 PM | Nuts Unlimited\Folsom\Packaging\Recipe\RecipeVarianceExists\Alarm | Cleared, Unacknowledged | Low |

- General
- Numeric
- Metadata
- Permissions
- History
- Alarming**
- Tag Events

Alarm Configuration

Alarm - Equal, Low

|
 |
 |
 |
 |
 |

Main

| | | |
|--------------------|--------|--|
| Name | Alarm | |
| Enabled | true | |
| Priority | Low | |
| Timestamp Source | System | |
| Display Path | | |
| Ack Mode | Manual | |
| Notes | | |
| Ack Notes Required | false | |
| Shelving Allowed | true | |

Alarm Mode Settings

| | | |
|----------|-------|---|
| Mode | Equal | |
| Setpoint | | 1 |

Deadbands and Time Delays

| | | |
|------------------------|----------|---|
| Deadband | | 0 |
| Deadband Mode | Absolute | |
| Active delay (seconds) | | 0 |

6.8 Add Recipe Selection to the Operations Segment

The Operation Segments created for us by the OEE Material Manager can be used to trigger the downloading of recipe values, and we can choose whether we want this done during the changeover segment or the production segment. If the recipe was part of the Track & Trace implementation, we could associate the recipe to any of the operation segments created. In fact we could create a process segment which was simply 'Setup Recipe' and use that to download recipe settings as well as control who could do the operation.

In this section, we will associate the Mixed Nuts 16oz recipe (non-modified recipe) with the Production Operations Segment so that the recipe values are downloaded whenever Packaging Line 1 is ready to run a Mixed Nuts 8oz packaging operation.

- Open the **MES Management** window under **Administration** and go into preview mode (F5).
- Select **Segments & Operations** and click on the **Mixed Nuts 16oz:Nuts Unlimited:Folsom:Packaging:Packaging Line 1** operation.
- Right-click on the Production operation segment (second segment under the operations definition) and select **Edit Settings**.
- Use the dropdown to select **Mixed Nuts 16oz** for the **Segment Recipe Name** property.

Operations Segment, Mixed Nuts 8oz-Nuts Unlimited:Folsom:Packaging:Packaging Line 1_CO

| Core Properties | |
|-----------------------------|---|
| Name | Mixed Nuts 8oz-Nuts Unlimited:Folsom:Packagir |
| Description | |
| End Operation When Complete | <input type="checkbox"/> |
| Segment Recipe Name | Mixed Nuts 8oz |

| Custom Properties | |
|-------------------|------------------|
| none defined | Mixed Nuts 8oz |
| | Mixed Nuts 100oz |
| | Mixed Nuts 16oz |

We have now associated the **Mixed Nuts 16oz** recipe with the Mixed Nuts 16oz packaging operation on Packaging Line 1. Whenever the line goes into Production, these recipe setpoints will be downloaded.

It's entirely up to you how you set this up.



- Perhaps you want a line technician to control when the setpoints are downloaded, in which case you can simply use the [Recipe Selector List](#) component.
- Perhaps you want the recipe downloaded during changeover in which case set the **Segment Recipe Name** in the Changeover Operations Segment.
- Perhaps you want different setpoints for the Changeover and Production Operations Segments.
- Perhaps you want to go into a setup operation after changeover and before production, in which case you can create a process segment called **Setup** and add it the Mixed Nuts Operation Definition.

The system is extremely flexible to tailor it to how you need it to operate, however be advised that some customizations such as adding a setup process segment will mean that you cannot use the OEE Run Director component.


Let's test that the recipe values will change whenever we start a packaging operation

- Open the **Packing** window and set the OEE Run Director *Selection Mode* property to **Material**.
- Drag a new **Recipe Values Display** template instance onto the screen.
- Copy the **plc_PackagingLine1** tag path and paste it into the **RecipeTagPath** template property.
- Copy and Paste the **ActiveRecipeName** label, **RecipeActive** and **VarianceExists** Multi-State Indicators from the Recipe control window onto the Packing window.
- **Save** your changes.

We'll now start the operation and monitor the value changes.


- Go into preview mode by pressing **F5**.
- Select **Packaging Line 1** in the MES Object Selector.
- Select **Mixed Nuts 16oz** in the OEE Run Director and click  to start the Changeover.
- Click  to end Changeover and go into Production.

You should see the **Mixed Nuts 16oz** recipe become active and the recipe tag values change (as long as a recipe with different values had been selected before).

| | | | | | |
|---|--|-------------------------------|----------------------------|------------------|---------------------|
| Line Mode
Production | Line State
Filler - Case Jam | Infeed
37 | Outf...
0 | Waste
0 | Elapsed
0.4 mins |
| Work Order | Product Code
Mixed Nuts 16oz | Std. Rate
150.0 per Min | Sch. Rate
100.0 per Min | Std. Count
55 | |
| Packaging Line 1 | Mixed Nuts 16oz | Mixed Nuts 16oz Active | | | |
|  | | Variance Exists | | | |
| | | Case Size | 80 | | |
| | | Min Weight | 20 | | |
| | | Target Weight | 22 | | |
| | | Max Weight | 24 | | |
| | | Container Size | 44.8 | | |
| | | Filler Speed | 100 | | |
| | | Pattern Code | 99 | | |
| Downtime | | Run Chart | Tracer | | |

6.9 Using Analysis to Obtain Recipe Information

Whereas OEE 2.0 now uses the new MES Analysis Controller component, the Recipe and SPC modules use the original Analysis controller component. This component now resides in the **Production Legacy** component palette.

-  Download [Recipe Analysis.proj](#) window and import into your project.
- **Save** your changes.
- Go into preview mode (**F5**) and create some stored analysis.

Recipe and SPC analysis can also be generated through scripting. Refer to [system.production.utils.createAnalysisSettings\(\)](#) for more details.

Titled Panel
menu

Variance Report

Filter By
Add

Item Path
☒ Nuts Unlimited\Folsom\Packaging\Packaging Line 1\Filler

Compare By
Add

Data Points
Add

☒ Recipe Name
☒ Item Path
☒ Value Name
☒ From Value
☒ To Value

| Recipe Name | Item Path | Value Name | From Value | To Value |
|-----------------|------------------------------------|------------|------------|----------|
| Mixed Nuts 16oz | Nuts Unlimited\Folsom\Packaging... | Speed | 400.0 | 405.0 |
| Mixed Nuts 16oz | Nuts Unlimited\Folsom\Packaging... | Speed | 405.0 | 400.0 |
| Mixed Nuts 16oz | Nuts Unlimited\Folsom\Packaging... | Speed | 400.0 | 410.0 |

6.10 Recipe Review

This concludes the Recipe portion of the tutorial. In the next section we shall learn about the SPC Module and continue the Nuts Unlimited project by adding SPC to the packaging line.

6.10.1 What We Covered

In summary, we covered the following:

- Features and Framework of the Recipe module
- Creating recipe tags and adding them to the Production Model
- Using the Recipe Editor to create recipes
- Change Management
- Variance Monitoring

6.10.2 What We Didn't Cover

What we haven't covered or touched in great detail are:

- [Default Recipe Values and Sub Recipes](#)
- [Master Recipes and Descendants](#)
- [Recipe Scaling](#)
- [Recipe Security](#)
- [Recipe States and Recipe Groups](#)
- [Importing and Exporting Recipes](#)
- [Analysis and Reports](#)
- [Recipe Scripting](#)

6.10.3 Additional Resources

We can't cover everything in a tutorial, but we do have more resources available in the [MES Help Manual](#) and [Sepasoft Knowledge Base](#) for the areas we didn't cover.


» STATISTICAL PROCESS CONTROL

- › SPC Module Features
- › Application of SPC
- › Control Charts
- › Control Limits
- › Gathering the SPC Requirements
- › Add SPC to Packaging Line
- › Add SPC to Mixing Line
- › Notification of Out-of-Control Processes
- › SPC Review

7 SPC

In this section we will add Statistical Process Control to our Nuts Packaging Line.

We've already installed the SPC module at the beginning of the tutorial, but we have some pre-built screens and icons that we can add to our project.

-  Download [MES_2.0_Training - SPC Base.proj](#) and import all the screens into your project.
- **Save** your project.

Before we move on, let's review what SPC is and then take a look at the features provided by the SPC module.

7.1 SPC Module Features

Now that we understand what SPC is, let's have a look at the features provided by the SPC module.

7.2 Application Of SPC

Before we add SPC to our Packaging Line, let's understand what information we need to be able to implement an SPC solution.

7.3 Control Charts

There are a number of different types of control charts that we can use for SPC. Let's take a look at them.

7.4 Control Limits

7.5 Gathering the SPC Requirements

For our tutorial, we are going to add SPC to the Checkweigher process cell to ensure that the container weight of the packaged goods is within our spec limits for the product. The checkweigher cell provides us with a tag value that we can add real-time data collection to and then monitor during our production runs.

We will also add an inspected count of nut defects to the Mixing Line. This will be a visual inspection performed by an operator or QA tech and so we will create a manual sample entry screen to obtain the data. We'll use the table below to define what we are going to collect, who, how and where we are going to collect it and what type of control charting is required.

| Sample Definition | Sample Attribute | Data Type | Location | Sample Data Source | Sample Interval | Access Roles | # of Measurements per Sample | Spec Limits | Control Limits | Signals |
|-------------------|------------------|---------------------|---------------------------------|--|--|---|------------------------------|---------------------------------------|---------------------------------------|--|
| CheckWeigher | Weight | Float | Packaging Line 1 - Checkweigher | Tag - Nuts
Unlimited\Folsom\Packaging\PLC\Packaging Line 1\Checkweigher | Every 10 Value Changes during production run | Auto-collection

Process Engineer role required to adjust Control limits | 1 | Min Weight, Target Weight, Max Weight | LSL, USL, Cp, LCL, Cp, UCL, Cp Target | Individual Outside, Individual Nelson Rule 3 |
| Nut Defects | | Inspected Count | Mixing Line - Inspection | Manual Entry | Timed Interval (every 3 minutes during production run) | 'Operator' role required to enter samples

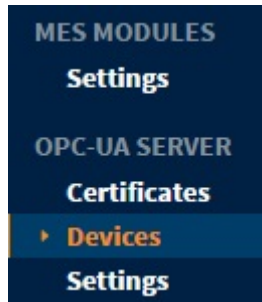
Sample Approval by 'Process Engineer' role | 1 | | np LCL, np UCL, p LCL and p UCL | |
| | Broken | Nonconforming Count | | | | | | | | |
| | Bad Color | Nonconforming Count | | | | | | | | |
| | Split | Nonconforming Count | | | | | | | | |
| | Too Small | Nonconforming Count | | | | | | | | |
| | Too Large | Nonconforming Count | | | | | | | | |

7.6 Add SPC to Packaging Line

In this section we will add SPC to the packaging line that was previously created. At the beginning of the tutorial, we installed the SPC module. Now, we will setup a simulator for the checkweigher so that the container weight value can be collected and displayed in SPC charts.

- In the Ignition Gateway configuration page, select **Configuration->OPC-UA->Devices**.
- Select the **Create New Device** link.

- Select the **Simulators Generic Simulator** driver.
- Enter the name **Checkweigher** and save changes.



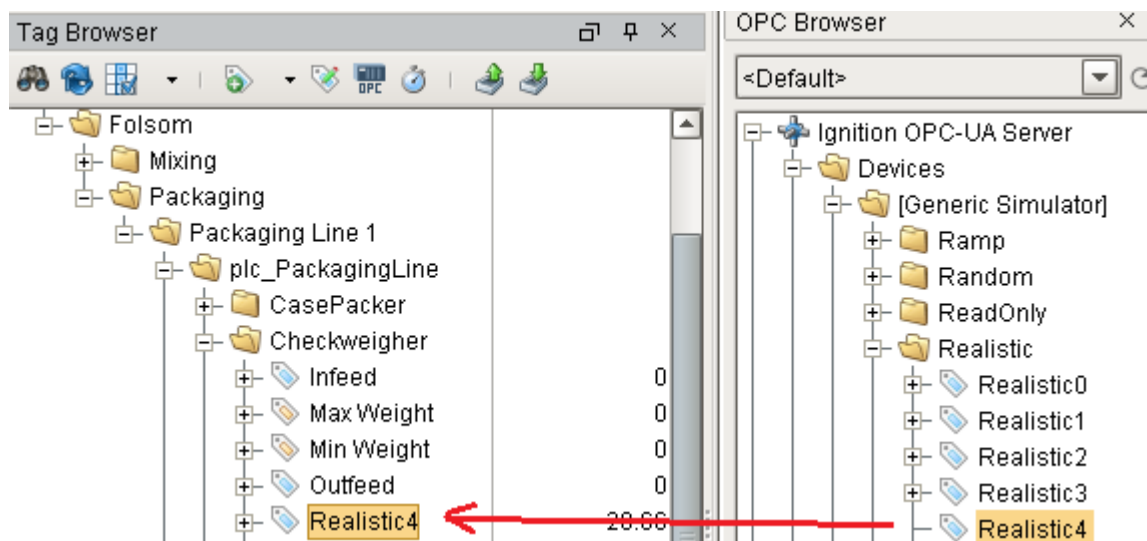
You should now see the new device with a status of **Connected**.

Devices

✓ Successfully created new Device "Generic Simulator"

| Name | Type | Description | Enabled | Status | | |
|-------------------|------------------------------|-------------|---------|-----------|--------|------|
| Generic Simulator | Simulators Generic Simulator | | true | Connected | delete | edit |
| Simulator | Production Simulator | | true | Connected | delete | edit |

- Go back into the Designer and open the **OPC Browser**.
- Navigate to **Devices->Checkweigher->Realistic->Realistic4** and drag it to the folder:
**Nuts Unlimited\Folsom\Packaging\Packaging Line
1\plc_PackagingLine\Checkweigher**



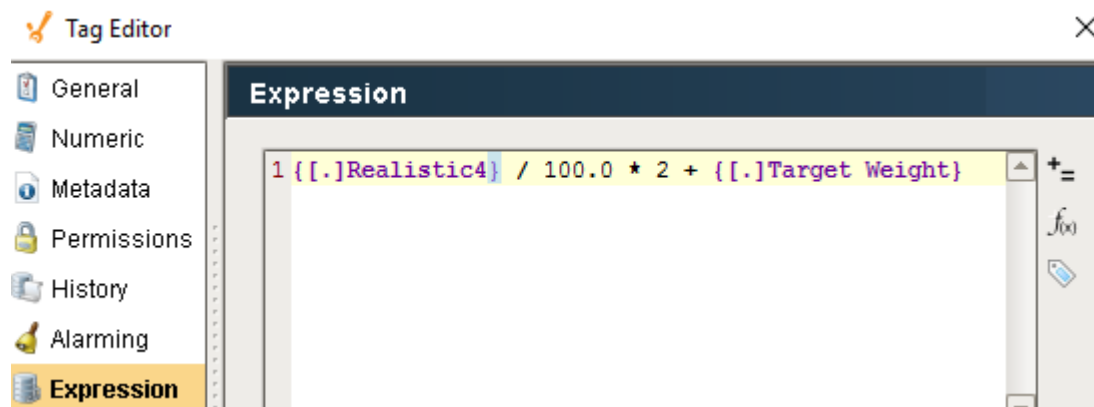
- You should have three Memory tags called **Min Weight**, **Target Weight** and **Max Weight** in this folder from the recipe section, but if not, go ahead and add them.

We will use these tags to set the spec limits for the checkweigher.

- Add a new **Expression Tag** in the same folder called **Weight** and set the data type to **Float**.
- Add the following expression to this tag:


```
{[.]Realistic4} / 100.0 * 2.0 + {[.]Target Weight}
```


The **Weight** tag uses the **Realistic4** tag as a seed value that will give us a somewhat realistic value that might come from a checkweigher.



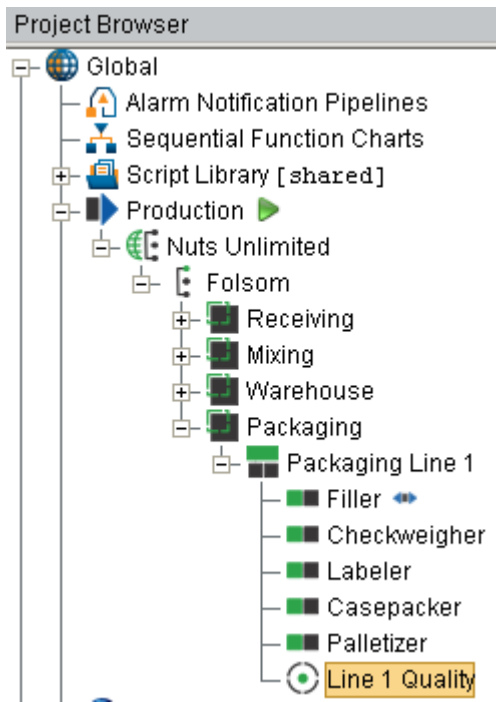
7.6.1 Add Checkweigher Tag Sample Collector

The sample data for the Checkweigher container weight will come from the simulation tag we just created. When we use SPC with data that we can collect in real-time, we use a Tag Sample Collector. These collectors can be added, edited or deleted for the SPC Location in the Production Model Designer under the **Quality** tab. Refer to the [Tag Sample Collectors](#) in the Help Manual for more information.


SPC samples are always collected at **Location**  production items in the Production Model, so we'll add a Location called **Line 1 Quality** for where this sample will come from.


- In the Production Model Designer, add a Location  production item to the Packaging line called **Line 1 Quality**.

For our tutorial, we are only collecting samples for the Checkweigher at this location, but really, any samples that we want to associate with the Packaging line 1 could use this location.



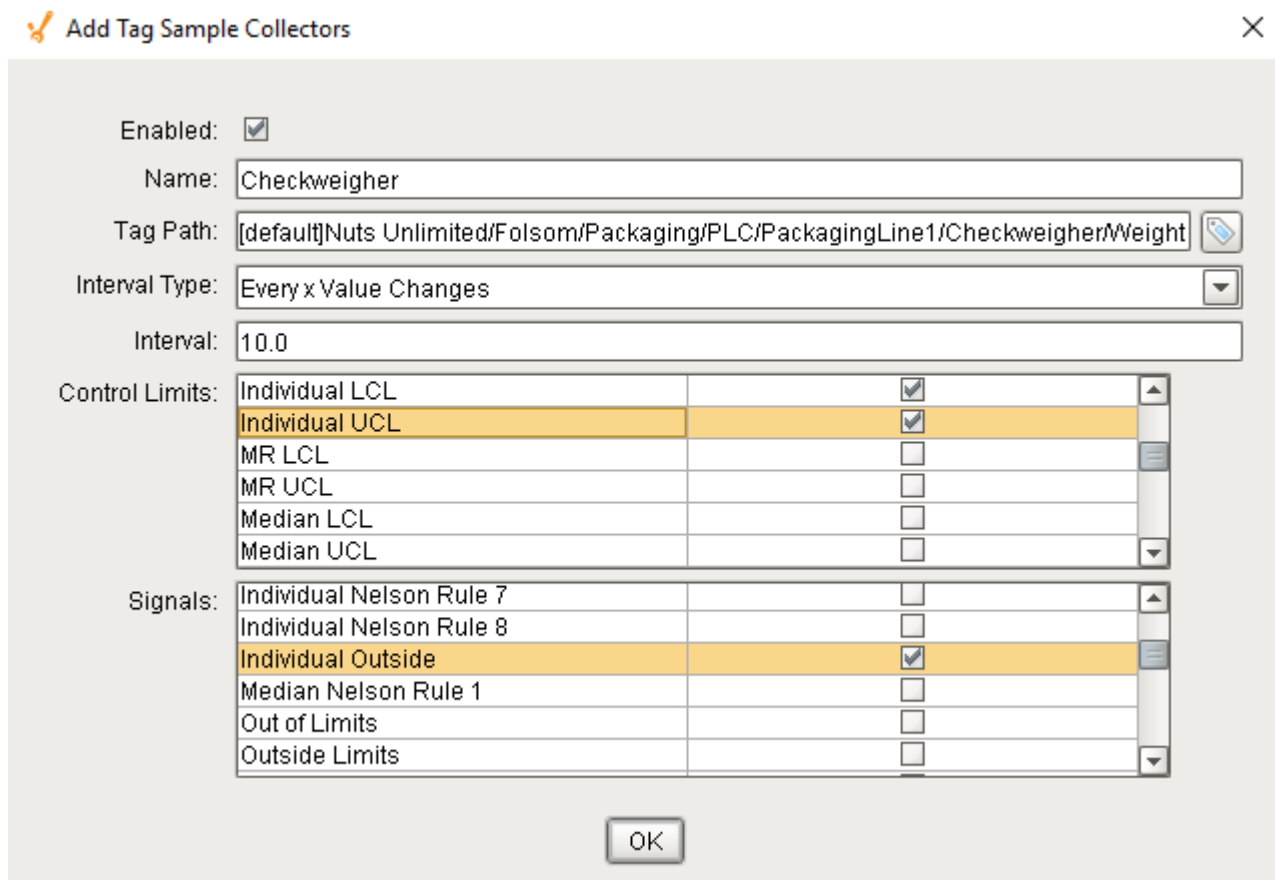
Now let's add a tag sample collector called **Checkweigher** to our **Line 1 Quality** location.

- In the Production Model Designer, select the **Line 1 Quality** location and select the **Quality** tab on the right.
- Right-click in the **Tag Sample Collectors** pane and select  **New** menu item.
- Configure the collector with the following settings:

| Property | Value | Description |
|-----------------|--|---|
| Enabled | True | Tag collectors can be enabled and disabled with this property |
| Name | Checkweigher | Name of the Tag Collector we can select in our Control Charts. Tag Collectors will have 'SQL-' pre pended to the name. |
| Tag Path | Nuts Unlimited/Folsom
/Packaging/Packaging
Line 1
/plc_PackagingLine1
/Checkweigher/Weight | Path to the Checkweigher tag. Your path might be different. Use the  icon to navigate to your tag. |

| Property | Value | Description |
|----------------|---|--|
| Interval Type | Every x Value changes | We will take a sample value every ten times the value changes. Refer to the Sample Intervals section in the help manual for more information |
| Interval | 10.0 | |
| Control Limits | <ul style="list-style-type: none"> • Individual LCL • Individual UCL • Cp LSL • Cp USL • Cp Target | SPC will notify us if the value falls outside of the LCL and UCL limits. Refer to the Control Limits section in the help manual for more information |
| Signals | <ul style="list-style-type: none"> • Individual Outside • Individual Nelson Rule 3 | SPC will notify us if there are 6 consecutive points in increasing or decreasing order or if we fall outside of the Individual Control limits. Refer to the Out of Control Signals section in the help manual for more information |

- Press **OK** to save and make sure to save your changes in the designer as well.

 Add Tag Sample Collectors

Enabled: ☒

Name: Checkweigher

Tag Path: [default]Nuts Unlimited/Folsom/Packaging/PLC/PackagingLine1/Checkweigher/Weight

Interval Type: Every x Value Changes

Interval: 10.0

Control Limits:

| | |
|----------------|-------------------------------------|
| Individual LCL | <input checked="" type="checkbox"/> |
| Individual UCL | <input checked="" type="checkbox"/> |
| MR LCL | <input type="checkbox"/> |
| MR UCL | <input type="checkbox"/> |
| Median LCL | <input type="checkbox"/> |
| Median UCL | <input type="checkbox"/> |

Signals:

| | |
|--------------------------|-------------------------------------|
| Individual Nelson Rule 7 | <input type="checkbox"/> |
| Individual Nelson Rule 8 | <input type="checkbox"/> |
| Individual Outside | <input checked="" type="checkbox"/> |
| Median Nelson Rule 1 | <input type="checkbox"/> |
| Out of Limits | <input type="checkbox"/> |
| Outside Limits | <input type="checkbox"/> |

OK

As soon as you press **Save** in the designer, the weight tag value of the Checkweigher will start to be automatically sampled based on the interval type. In this case, we will collect one sample after every ten value changes. That's it!

7.6.2 View Checkweigher Sample Data

We configured the collection of checkweigher values in the previous section and some data will have been collected by now. Let's take a look at the data in the control charts.

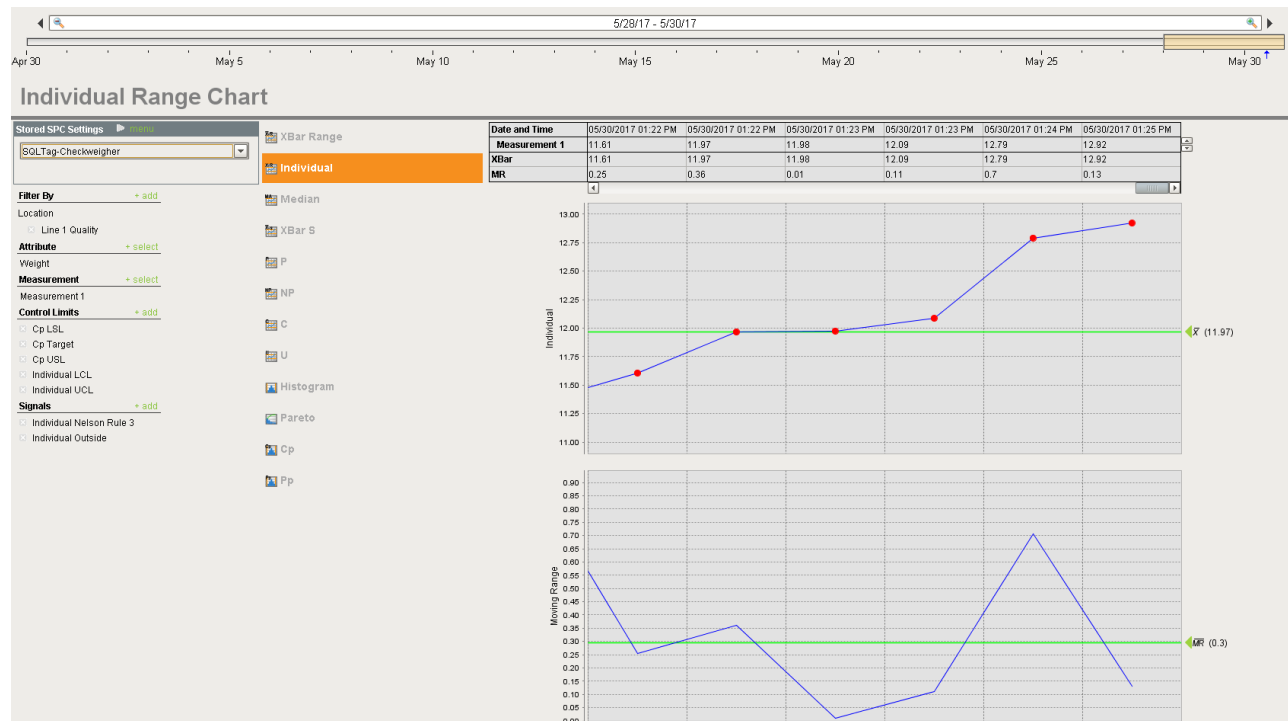
- Open the **Control Charts** window located in the **Quality** folder.
- Select the **SQLTag-Checkweigher** stored SPC settings and view the results.

The Control Charts window has been built out of the SPC Stored Selector, SPC Selector and Control Chart components. This window allows us to select a SPC sample and view it on the different control charts provided. Not all charts are applicable for samples, that will be dependent on the sample configuration i.e.

number of measurements (sub-group) and whether the attributes are value based, boolean, Inspected Count, Nonconfirming or Nonconformity Count types.

In the example, we are viewing the checkweigher values on an Individual Control chart. The red dots indicate that those sample points are in violation of the Nelson Rule 3.

The charts provide many properties that will allow us to customize how they look and the functions available to a user. We can configure based on user roles, whether a user can hide or delete sample points, add notes, show and even set control limits through these chart components. We'll play with these properties in a later section.




Viewing this in the Control Chart window is interesting for some but may not be appropriate for the operator. Next, we'll add a control chart to the Packing window.

7.6.3 Modify Packing Screen

We could create a new screen with all the Control Charts on, but that may not be appropriate or convenient for an operator, so we'll add a new window with a control chart for the Packaging Line using the **SPC Controller** component to obtain the SPC data. The **SPC Controller** is like the **SPC Selector** which we used on the **Control Chart** but without the user interface. All selections are set using properties of the **SPC Controller**. This component is not visible in the client, only in the designer.

- Open the **Packing - SPC** window in the **Packaging Folder** and add a SPC Controller component to it from the SPC component palette.
- Change the **Stored SPC Name** property to **SQLTag-Checkweigher**.
- Add the following expression binding to the **End Date** property:
 - **now(10000)**
- Add the following expression binding to the **Start Date** property:
 - **addHours({Root Container.SPC Controller.endDate}, -4)**
- Change the **Automatic Update** property to True.
- Change the **Auto Refresh** property to True.

As we have been collecting checkweigher samples for some time now, we should have some SPC Data in the dataset. If you do not see any data, check the **Error Message** and **Warning Message** property of the SPC Controller.



Property Editor

Common

Name: SPC Controller

Data

| | |
|-----------------------------|--|
| Automatic Update | <input checked="" type="checkbox"/> true |
| Auto Refresh | <input checked="" type="checkbox"/> true |
| Row Limit | 100 |
| SPC Data Format | Individual |
| Stored SPC Name | SQLTag-Checkweigher |
| Definition Name | SQLTag-Checkweigher |
| Attribute Name | Weight |
| AttributeUnits | |
| Use Default Chart Type | <input type="checkbox"/> false |
| Filter | Location=Nuts Unlimited\Folsom\Packaging\Packaging Line 1\Line 1 Quality |
| Control Limits | Cp LSL,Cp Target,Cp USL,Individual LCL,Individual UCL |
| Signals | Individual Nelson Rule 3,Individual Outside |
| Measurement Filter | Measurement 1 |
| Nonconforming Filter | |
| Nonconformity Filter | |
| Pareto Filter | |
| Additional Factors | |
| Include Disabled Attributes | <input type="checkbox"/> false |
| Start Date | 05/30/2017 13:02:03 -0700 |
| End Date | 05/30/2017 17:02:04 -0700 |
| SPC Data | Dataset [100R x 13C] |
| Error Message | |
| Warning Message | |

Now let's visualize the SPC data from the Checkweigher.

- Drag a **Process Capability Chart** component from the SPC component palette to the **Packing - SPC** window.

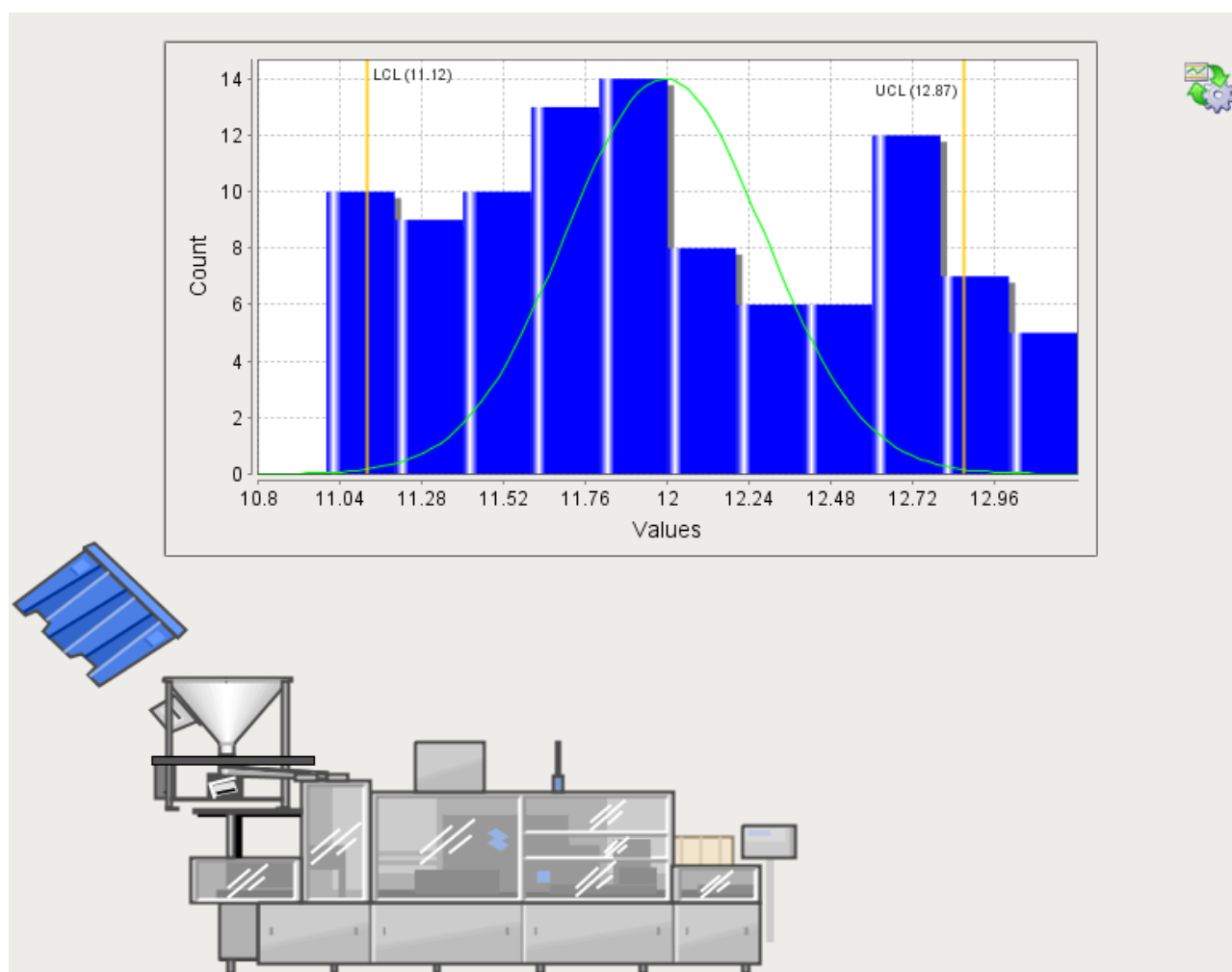
- Bind the **SPC Results** property of the Chart to the **SPC Results** property of the SPC Controller.

Nothing right? Component keeps 'retrieving data, but displays nothing. We need to tell the SPC controller what type of SPC results we want returned.

- Set the **SPC Data Format** property of the **SPC Controller** to **Process Capability**.

*The **SPC Data Format** property will default back to **Individual**, so bind it to an expression with the value **11**, to force it to the **Process Capability** format.*

Now you see a bar chart that displays a count of grouped sample values along with a distribution curve and control limits.



The Process Capability Chart is generally used when a process is stable to provide analysis information on just how capable a process is to meet a specification. It's a good chart for us to use because our application will provide information to operators and process engineers on the stability of the process.

Setting Control Limits

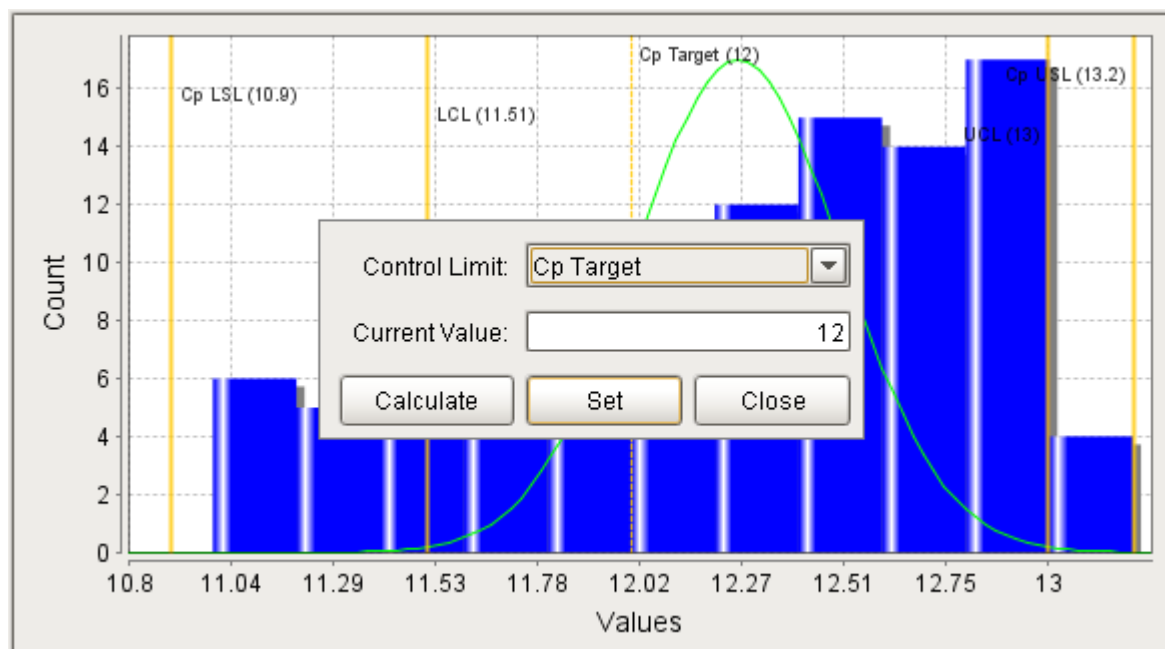
When we configured the Checkweigher SPC Tag collector in the Quality tab of the Production Designer for the Line 1 Quality location, we enabled the following control limits: **Individual LCL**, **Individual UCL**, **Cp LSL**, **Cp Target** and **Cp USL**. On our chart we can see LCL and UCL being displayed and that is because the values are being calculated for us in the script defined for these control limits. The Cp Target, Cp LSL and Cp USL however are simply specification limits and we have no script defined for how these values should be set.

There are a number of ways that we can set what these limit values should be.

Setting Control limits using the Chart Components

- Go into Preview mode in the Designer (F5) and right-click on the chart.
- Select **Cp Target**, set a value somewhere in the mid-range of your dataset and press **Set**.
- Do the same for Cp LSL and Cp USL and set them somewhere outside of the current dataset.

If you press the **Calculate** button, it will generate an error for the Cp limits as no script is defined. When you press **Calculate**, it simply executes the defined script in the Quality Tab for the Enterprise production item. Try it on the Individual LCL. It will pass the sample data points displayed in the control chart to the defined script and return the lower control limit as 3 sigma from mean.



Setting Control Limits through Scripting




It may well be that product specification limits are stored in a Quality Management System (QMS), an Industrial Engineering database, a Product Lifecycle Management system (PLM) or simply stored in spreadsheets. If this is the case, then we probably want to access the specifications when we take samples so that we can apply the correct control limits. We can do this by adding a script to the Cp USL, Cp LSL and Cp Target control limits in the designer that queries the other system and sets the control limit. We can also do it through scripting when ever we see a production run start on a line or the product code change. When we get to adding SPC to the Mixing Line, we will go through control limit scripting .

7.7 Add SPC to Mixing Line

In this section we'll add SPC to the Mixing Line operation that we created during the Track & Trace section.

7.7.1 Create Sample Definition

Now we'll create a sample definition for **Nut Defects**. Because this will be a manually entered sample, we will use the components in the Quality component palette to create the sample definition as opposed to dragging tags into the production model. At the beginning of this SPC section, we imported a screen that uses these components allows us to add new definitions and assign attributes, locations, control limits, and signals to the definition.



- Open the **Sample Manager** window under **Configuration** in the navigation window.
- In the **Definitions** section
 - Enter **Nut Defects** for the sample name
 - Set the **Measurement Count** to **1** since we will only taking a single measurement whenever we take a sample.
 - Make sure the definition is enabled.
 - Set the **Interval Type** to **Manual**.
- Click the  **Add** button. *If the button says  **Update**, press the  **Clear** button to clear the currently selected sample definition*

Sample Name: Interval Type:

Description:

Interval: Duration:

Enabled: ☒ Auto Approve: ☒ Coming Due: Overdue:

 **Clear**  **Add** Measurement Count:

7.7.2 Add Sample Attributes

- In the **Attribute** pane, add the following attributes using the settings as shown in the table below.

| Name | Inspected Count | Broken | Bad Color | Split | Too Small | Too Large |
|----------------------|-----------------|---------------------|---------------------|---------------------|---------------------|---------------------|
| Required | True | True | True | True | True | True |
| Datatype | Inspected Count | Nonconforming Count | Nonconforming Count | Nonconforming Count | Nonconforming Count | Nonconforming Count |
| Format | #0 | #0 | #0 | #0 | #0 | #0 |
| Default Value | 10 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 |

| Name | Inspected Count | Broken | Bad Color | Split | Too Small | Too Large |
|-----------|-----------------|--------|-----------|-------|-----------|-----------|
| Min Value | | | | | | |
| Max Value | 10 | 10 | 10 | 10 | 10 | 10 |

Name: Enabled: ☒ Required: ☒
 Description: Default Chart Type:
 Attribute Weight: Order: Measurement Count:

| | Default | Min | Max | Units | Format | Datatype |
|--------|---------------------------------|--------------------------------|---------------------------------|----------------------|---------------------------------|--|
| Value: | <input type="text" value="10"/> | <input type="text" value="0"/> | <input type="text" value="10"/> | <input type="text"/> | <input type="text" value="#0"/> | <input type="text" value="Inspected Count"/> |

The data type should match the type of value to be entered, whether it is an integer or a floating point value or other type. If the default value setting is not blank, that value will show up in the form when a user enters in a sample. Otherwise, it will be blank. The min and max values, if specified, will be used to perform range validation. Lastly, set the required setting if a user must enter in a value for the attribute. Otherwise, they could skip it.

7.7.3 Define Sample Locations

When creating a sample definition, we can also define where the sample comes from by associating it to a SPC Location production item. This gives us the ability to return Sample Definitions by Location in the [Definition Selector](#) component, and to filter sample data by location.

- In the **Locations** pane, set the location settings to the following values and leave all other values at their default values:

| Settings | Inspection Location | Description |
|---------------|--------------------------|--|
| Location | Line 1 Quality | SPC location that can be selected for this sample |
| Enabled | True | Location enabled for this sample |
| Auto Approve | False | We will require a sample entry to be approved before it will be considered in Control Limit calculations |
| Interval Type | Timed Interval (Minutes) | A sample will be scheduled every 3 minutes |
| Interval | 3 | |
| Tag | Test Stations | Provides a filter mechanism that we can use in the Location Sample List component |

Location: Interval Type:
 Tag: Interval: Duration:
 Enabled: ☒ Auto Approve: ☐ Coming Due: Overdue:

7.7.4 Add Control Limits and Signals

We can now specify the control limits and signals for this sample definition.

- Select the **np LCL**, **np UCL**, **p LCL** and **p UCL** Control Limits.
- Click the **Save** button to save this sample definition.

| Control Limits | | |
|----------------|--|-------------------------------------|
| Name | | Enable |
| Individual LCL | | <input type="checkbox"/> |
| Individual UCL | | <input type="checkbox"/> |
| Median LCL | | <input type="checkbox"/> |
| Median UCL | | <input type="checkbox"/> |
| MR LCL | | <input type="checkbox"/> |
| MR UCL | | <input type="checkbox"/> |
| np LCL | | <input checked="" type="checkbox"/> |
| np UCL | | <input checked="" type="checkbox"/> |
| p LCL | | <input checked="" type="checkbox"/> |
| p UCL | | <input checked="" type="checkbox"/> |
| Pp LSL | | <input type="checkbox"/> |
| Pp Target | | <input type="checkbox"/> |
| Pp USL | | <input type="checkbox"/> |

7.7.5 Add Mixing Samples

We can now add some samples for the mixing line to test that we have setup the sample definition correctly. We could build a screen using the components provided in the Quality component palette, but we imported a screen that has already been built.

- From the runtime client, open up the **Samples Due** screen under **Quality**.

This window uses the [Location Sample List](#), [Location Selector](#) and [Sample Entry](#) components. It can be used to display scheduled samples that are coming due, to create unscheduled samples and to enter sample data into.

You'll probably notice that it is displaying every Checkweigher sample that has been created. We can't use the [Location Selector](#) component to filter these out as both samples are associated with this location, but we can use the **Tag** filter property of the [Location Sample List](#) to only show samples associated with the **Test Stations** value that we defined in the **Location** for the Nut Defects sample definition.

- In the Designer, open up the **Samples Due** screen under **Quality** and set the **Tag** property of the [Location Sample List](#) to **Test Stations**.
- **Save** your changes in the designer and update the client window.

Now we should only see scheduled **Nut Defect** samples.

- In the runtime client, go ahead and **Edit Samples** to add some values that we can see later in Control Charts.
- You can enter **Product Code** and **Work Order** information along with the samples if you'd like.

*The SPC Module provides a product code field for samples. For the Work Order, we used the **refNo** field of the sample object to store this information, but in your application, you could use the **refNo** field for any type of metadata.*

We set **Auto-Approve** to false, so press the **Approve** button to approve them. Only approved samples will show up in Control Charts.

Location: <Select One>

Show Samples Legend
☐ Overdue ☐ Due ☐ Coming Due ☐ Waiting Approval ☒ Approved

| Sample | ProductCo... | Line | Work Order | Scheduled | Taken On | Taken By | Note |
|----------------------|--------------|-----------------|------------|--------------------|------------------|----------|------|
| Nut Defects | | Packaging Li... | | 3/2/2018 1:54 PM | | | |
| Nut Defects | | Packaging Li... | | 3/2/2018 1:57 PM | | | |
| Nut Defects For Pete | | Packaging Li... | | 3/1/2018 1:54 PM | 3/1/18 1:45 PM | admin | |
| Nut Defects For Pete | | Packaging Li... | | | 3/1/18 1:45 PM | admin | |
| Nut Defects For Pete | | Packaging Li... | | 3/1/2018 1:55 PM | 3/1/18 1:45 PM | admin | |
| Nut Defects For Pete | | Packaging Li... | | 3/1/2018 1:52 PM | 3/1/18 9:24 AM | admin | |
| Nut Defects For Pete | | Packaging Li... | | 3/1/2018 1:53 PM | 3/1/18 9:24 AM | admin | |
| Nut Defects TEST | | Packaging Li... | | 3/1/2018 10:58 ... | 3/22/17 9:24 ... | admin | |
| Nut Defects TEST | | Packaging Li... | | 3/1/2018 11:01 ... | 3/22/17 9:24 ... | admin | |
| Nut Defects TEST | | Packaging Li... | | 3/1/2018 11:04 ... | 3/22/17 9:24 ... | admin | |
| Nut Defects TEST | | Packaging Li... | | 3/1/2018 11:07 ... | 3/22/17 9:24 ... | admin | |
| Nut Defects TEST | | Packaging Li... | | 3/1/2018 11:10 ... | 3/22/17 9:24 ... | admin | |
| Nut Defects TEST | | Packaging Li... | | 3/1/2018 11:13 ... | 3/22/17 9:24 ... | admin | |

Sample Information

| Attribute | Value |
|-----------------|-------|
| Inspected Count | 10 |
| Broken | 0 |
| Bad Color | 0 |
| Split | 0 |
| Too Small | 0 |
| Too Large | 0 |

Sample Definition: Nut Defects Work Order: Product Code:

Sample Taken By: admin Taken On: 03/02/2018 01:48 PM

☐ Show Limit Warning ☒ Show Units

Cancel Save

Add Note

Add Sample

Edit Sample

Approve

Un-Approve

Delete Sample

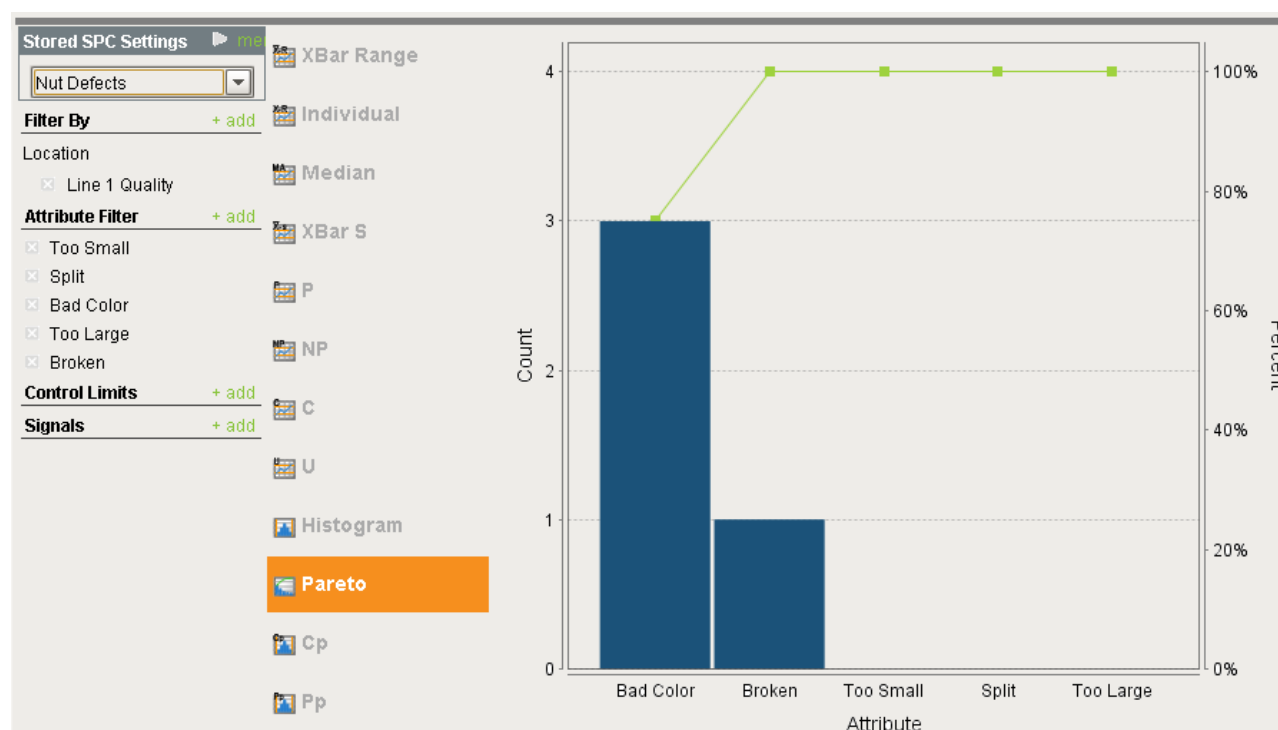
Delete All

7.7.6 View Mixing Sample Data

Now we can take a look at the type of control charts we can use for inspected and nonconforming counts type samples.

- In the runtime client, open the **Control Charts** window located in the **Quality** folder.
- Select the **Nut Defects** stored SPC settings and choose **Line 1 Quality** for **Location**.
- You can select the **Pareto**, **NP** or **P** charts to view the results.

If you don't see any results, check the **Sample Taken Date** of the sample in the Samples due screen and ensure your data range in the control Charts screen covers this time period.




7.8 Notification Of Out-Of-Control Processes

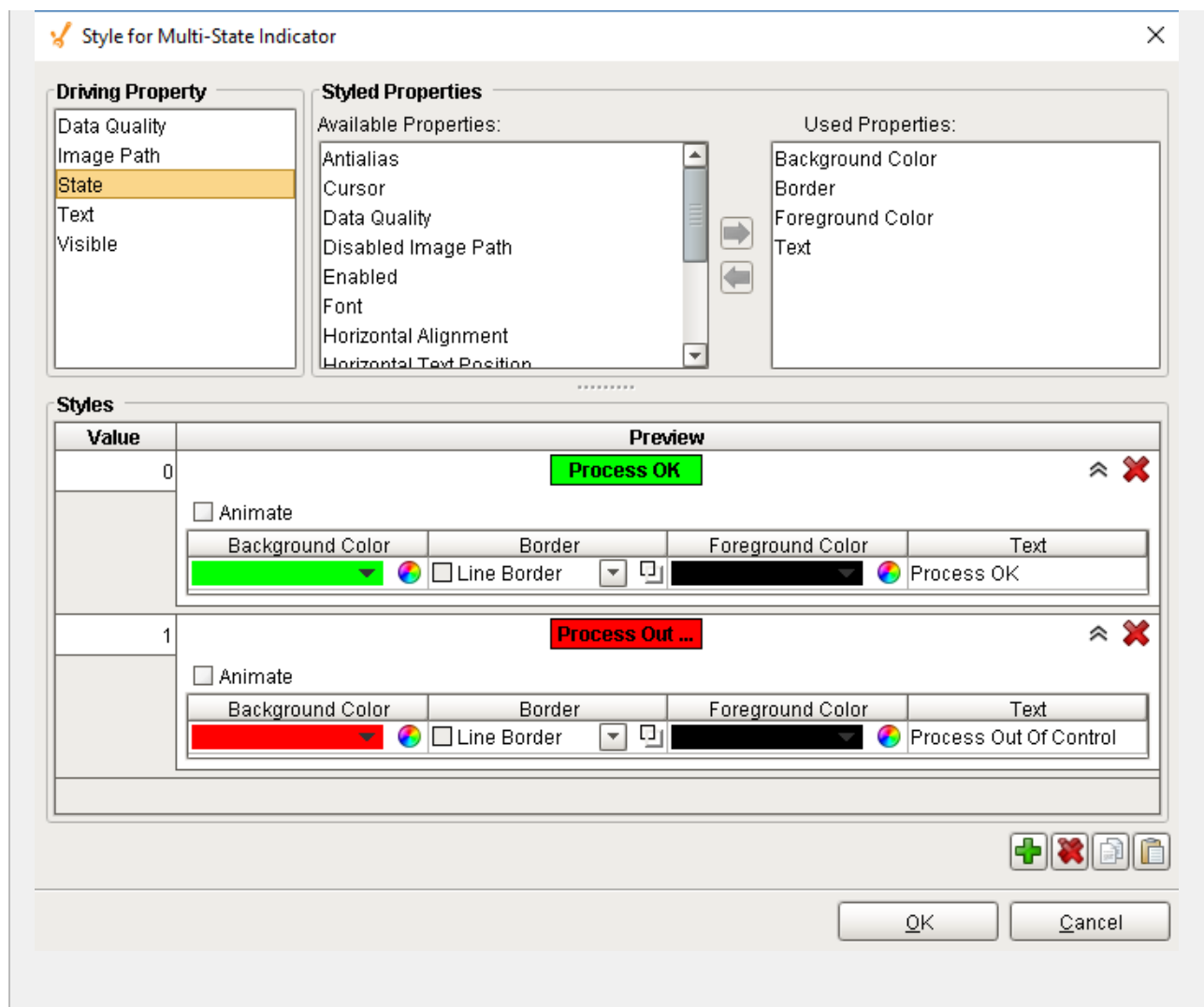
The SPC Module provides Production OPC Server tags that we can use in our application to alert us to out-of-control process conditions as well as missed samples. We'll use some of these to add alarming to our Nuts Unlimited project.

7.8.1 OPC Production Server Tags - SPC

For every SPC location that is defined in the Production Model, the following SPC tags are created:

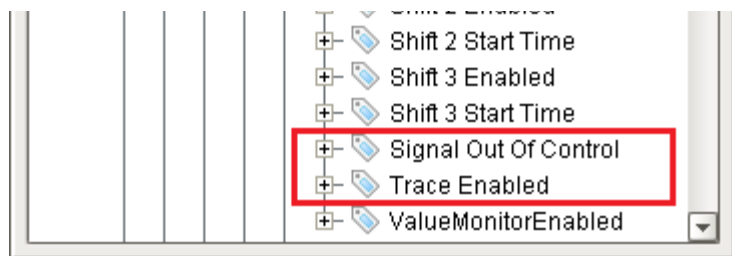
| Tag | DataType | Description |
|--------------------------------|----------|---|
| Sample Coming Due | Boolean | Scheduled sample is coming due. This flag is set based on the Coming Due value entered for the Location in the Sample Definition. |
| Sample Due | Boolean | Scheduled sample is now due. |
| Sample Overdue | Boolean | Scheduled sample is overdue. This flag is set based on the Over Due value entered for the Location in the Sample Definition. |
| Sample Waiting Approval | Boolean | Sample entry is waiting approval. |
| Signal Out of Control | Boolean | Becomes true when the sample violates the signals we setup for the sample. |
| TraceEnabled | Boolean | This flag is set whenever the system.production.setLocationProductCode() scripting function is used to associate a productCode to a location. It is cleared by the system.production.cancelLocationProductCode scripting function. The Intervals Once at Production Start and Once at Production End will check if Trace is enabled and create a sample. |

- Open the OPC Tag Browser by clicking on the  button in the Tag browser panel.
- Navigate to the **Signal Out Of Control** tag in the **Production\[global]\Nuts Unlimited\Folsom\Packaging\Packaging Line 1\Line 1 Quality** folder and drag it to your **Nuts Unlimited\Folsom\Packaging** folder.
- Drag your new **Signal Out Of Control** tag onto the **Packing - SPC** window in the **Packaging** folder and display it as a Multi-State Indicator.
- Right-click the Multi-State indicator, select **Customizer > Style Customizer** and set it up as follows....



Whenever an out-of-control condition is evaluated, it will be shown on our screen.

Process Out Of Control






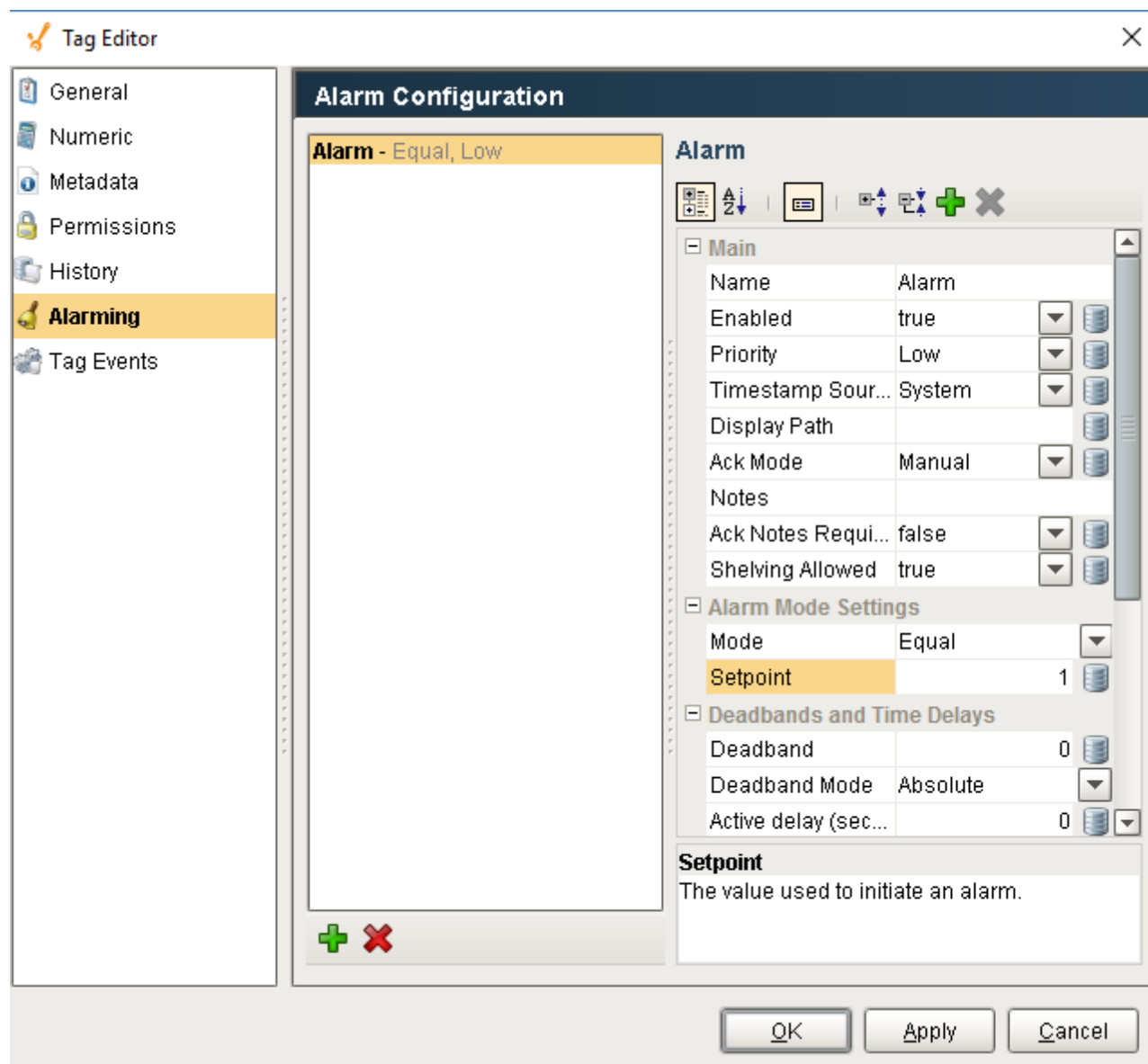
We could also add Ignition Alarming to the tag itself by editing the tag properties and then use the Ignition Alarm Table to display any alarms.

| <input type="checkbox"/> Active ... | Display Path | Current State | Priority |
|-------------------------------------|---|------------------------|----------|
| <input type="checkbox"/> | 6/1/17 3... Nuts Unlimited/Folsom/Signal Out Of Control/Alarm | Active, Unacknowledged | Low |

Acknowledge

Shelve





7.9 SPC Review

This concludes the SPC portion of the tutorial, however we are not entirely finished with SPC. In the final section of the tutorial, we will show how sample data can be pulled from a file using the Instrument Interface module and the scripting functions used to then create sample data.

7.9.1 What We Covered

In summary, we covered the following:

- [SPC Module Overview](#)
- [Types of Control Charts](#)
- [Using Tag Sample Collectors for real-time sample capture](#)
- [How to build a Process Capability screen using the SPC Controller component](#)
- [Setting Control Limits](#)
- [Creating Manual Sample Entries](#)
- [Entering Samples](#)

7.9.2 What We Didn't Cover

What we haven't covered or touched in great detail are:

- [Creating New Control Limits](#)
- [Saving Control Limits by Product Code](#)
- [Only capturing samples when production is running](#)
- [Creating sample definitions and sampling through scripting](#)

7.9.3 Additional Resources

We can't cover everything in a tutorial, but we do have more resources available in the [MES Help manual](#) and [Sepasoft Knowledge Base](#) for the areas we didn't cover.

» INSTRUMENT INTERFACE

- › File Monitoring
- › Parse Template
- › File Monitoring Component
- › Using the Instrument Interface with the SPC Module
- › Instrument Interface Review

8 Instrument Interface


In this section of the tutorial, we will finish off by going through the features of the Instrument Interface module and parsing data from a lab inspection equipment to our SPC application. Let's start off an overview.

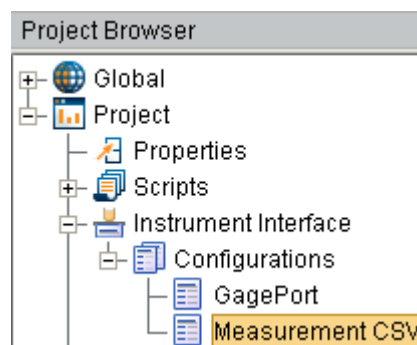
8.1 File Monitoring

Since we don't have a serial device connected to our application, we'll focus on configuring a simple file parser using the Instrument Interface module.

- Expand  **Instrument Interface** under  **Project** in the Production Model and click on the **Configurations** menu item.

We have no configurations yet. We could create new ones by right-clicking on **Configurations**, but we will import some that we've already made.

-  Download the Instrument configuration file and import it by choosing **File > Import**.
- Click on the **Measurement CSV** configuration.
- Click on the **File Monitor Settings** tab (since we are configuring file monitoring).



-  The **GagePort** configuration we imported provides an example of serial parsing.

The **File Monitor Settings** pane is where you can enable file monitoring, as well as specifying the settings such as how often to check for a file change or creation and what to do after parsing the file. If multiple files exist, we can define in what order they are processed here whether it is by timestamp or filename, and whether we delete them, move them or rename them once we are done parsing.

- Set the **Post Processing Handling** to **Append Extension**.
- **Save** your changes.


The screenshot shows the 'Measurement CSV' window with the 'Instrument Configuration' section. The 'File Monitor Settings' tab is selected. Under the 'General' section, 'Enable File Monitoring' is checked. Under the 'File Monitor Settings' section, 'Auto Monitor Files' is checked, 'Monitor Rate' is set to 1,000, 'File Processing Order' is set to 'File Timestamp', 'File Name Date Format' is empty, 'Post Process Handling' is set to 'Append Extension' (highlighted with a red box), 'Character Encoding' is set to 'UTF-8', 'Completion File Extension' is set to 'done', and 'Error File Extension' is set to 'error'.


8.2 Parse Template

The next step is to configure a parse template so that the Instrument module knows how to parse the files. In the Parse template pane, you'll see that we already have data. When setting up a parse template, you will simply copy example data from the file that you are creating a parse template for and use that to define where the values of interest are.

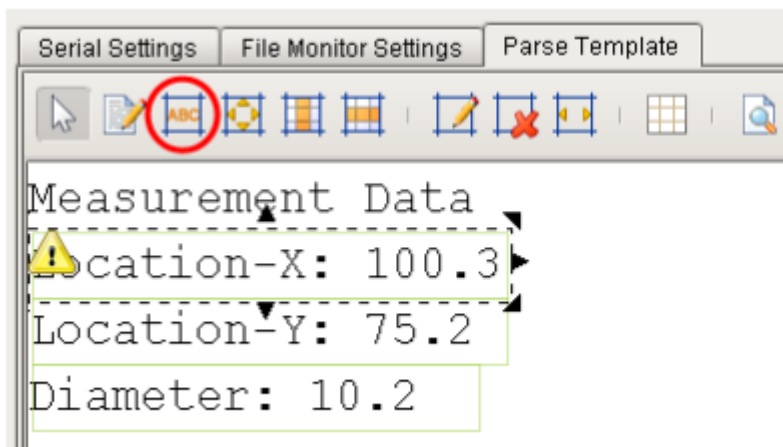
- Click on the **Parse Template** tab for the **Measurement CSV** configuration.

There are three green boxes already around the values that we are interested in. We'll remove those first.

- Right-click on the green boxes one at a time and select **Delete** or choose the  icon to remove the parsing box.

- Select the  tool and drag your cursor over **Location-X: 100.3** text.

A  warning sign will be displayed because the label parsing box is not yet configured.




- Right-click on the green box and select **Properties**.
- Give the box a name. we'll call it **LocationX**.

*We want to find the value to the right of the **Location-X**: label, so enter in **Location-X**: for the **Label Name**. The value position is to the right of the label.*

- Set the datatype to Float4.

You should see 100.3 as the result at the bottom.

- Now press the  button to see the parse results. If everything is successful you will see some values.


- Select the  tool and drag your cursor over **Location-Y: 75.2** text.
- Right-click on the green box and select **Properties**.
- Give the box a name. we'll call it **LocationY**.

*We want to find the value to the right of the **Location-Y:** label, so enter in **Location-Y:** for the **Label Name**. The value position is to the right of the label.*

- Set the datatype to Float4.

You should see 75.2 as the result at the bottom.

- Now press the  button to see the parse results. If everything is successful you will see some values.

- Select the  tool and drag your cursor over **Diameter: 10.2** text.
- Right-click on the green box and select **Properties**.
- Give the box a name. we'll call it **Diameter**.

*We want to find the value to the right of the **Diameter:** label, so enter in **Diameter:** for the **Label Name**. The value position is to the right of the label.*

- Set the datatype to Float4.

You should see 10.2 as the result at the bottom.

- Now press the  button to see the parse results. If everything is successful you will see some values.

Label Box Properties

General

Item Name:

Required: ☒ Required

Expand Horizontal: ☐

Include Units: ☐

Label Name:

Value Position:

Format

Datatype:

Format:

Information

Area Size: (2, 1) to (2, 17)

Character Count: 17

Result:

OK Cancel

8.3 File Monitoring Component

Now that we have an Instrument Interface configuration completed with settings and a parse template, we can now parse any files that are accessible from a client machine using the [File Monitor Controller](#) component. This component handles detecting, reading and parsing of data in files and is considered invisible as it appears on the screen in the designer but not during runtime.

- Add a new window called **File Monitoring** and drag the **File Monitor Controller** component from the **Instrument** component palette onto the window.

- Set the **Instrument Interface Name** property to **Measurement CSV**. Use the  icon to select it.

*When we selected **Measurement CSV**, all the configuration information we setup earlier is passed to the file monitor component.*

- Set the **File Path** to: **C:\Temp*.txt** to monitor for any **.txt** file in the **C** directory.

The next step is to tell the controller what to do with the data it parses. For that we need to configure a simple script on the component.

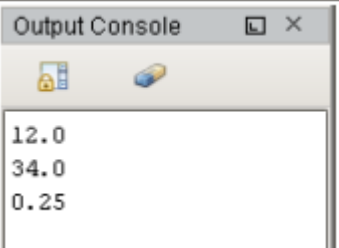
- Right click on the component and select scripting.
- Select the **onAfterParse** event in the parse folder. Enter in the following script on the Script Editor:

```
results =
event.
getParseResults()
if results.
isValid():
    print
results.
getValue("LocationX")
    print
results.
getValue("LocationY")
    print
results.
getValue("Diameter")
```

The script will print out the results to the console. You can do whatever you want with the values: write them to tags, write them to a database, use them with SPC, and more.

- Add a file called **parseMe.txt** in **C:\Temp** directory with the following contents:

```
Measurement Data
Location-X: 12
```

| | |
|---|---|
| | Location-Y: 34
Diameter: .25 |
| <ul style="list-style-type: none"> • Open the Output Console from Tools > Console menu in the designer and clear the panel using the eraser icon. • Go into preview mode by pressing F5. <p>The File Monitor component should parse the file automatically and then rename the extension. You can view the results in the console.</p> |  |

That's it! In the runtime client, when the window is opened, it will automatically parse files in a directory.

8.4 Using the Instrument Interface with the SPC Module

You can use the Instrument Interface module to create sample data from a flat file. In this section we will grab the data out a file and create a sample object that we can then display in a control chart.

We'll first add a new Sample definition that we'll use to store the measurement data we get through the Instrument interface.

- Create a new Sample Definition called **Measurement** using the **Definition Management** window in the runtime client.
- Set Measurement count to **1** and Interval Type to **Manual**.
- Add the following attributes:

| Attribute Name | Data Type | Format |
|------------------|-----------|--------|
| LocationX | Real | #0.0 |
| LocationY | Real | #0.0 |
| Diameter | Real | #0.0 |

- Add the following Location:

| Property | Interval Type |
|----------------|---------------|
| Line 1 Quality | Manual |

- Add the following Control limits:
 - Individual LCL
 - Individual UCL
- Add the following Signal:
 - Individual Outside
- **Save** your sample definition.

Definitions

| Name | Version |
|-------------|---------|
| Measurement | 1 |
| Nut Defects | 1 |

Attributes

| Name | Description | Data Type | For... | Enab... | Req... | Units | M... | De... | Weight |
|-----------|-------------|-----------|--------|-------------------------------------|-------------------------------------|-------|------|-------|--------|
| LocationX | | Real | ##0.00 | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | | 1 | None | 1 |
| LocationY | | Real | ##0.00 | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | | 1 | None | 1 |
| Diameter | | Real | ##0.00 | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | | 1 | None | 1 |

Locations

| Location ID | Location Na... | Interval Type | Interval | Duration | Tag | Auto Approve | Enabled |
|-------------|----------------|---------------|----------|----------|---------------|-------------------------------------|-------------------------------------|
| 4 | Line 1 Qual... | Manual | 0 | 5 | Test Stations | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |

Control Limits

| Name | Enable |
|---------------------|-------------------------------------|
| Box and Whisker LCL | <input type="checkbox"/> |
| Box and Whisker UCL | <input type="checkbox"/> |
| c LCL | <input type="checkbox"/> |
| c UCL | <input type="checkbox"/> |
| Cp LSL | <input type="checkbox"/> |
| Cp Target | <input type="checkbox"/> |
| Cp USL | <input type="checkbox"/> |
| CpPp LSL | <input type="checkbox"/> |
| CpPp Target | <input type="checkbox"/> |
| CpPp USL | <input type="checkbox"/> |
| Histogram LCL | <input type="checkbox"/> |
| Histogram UCL | <input type="checkbox"/> |
| Individual LCL | <input checked="" type="checkbox"/> |
| Individual UCL | <input checked="" type="checkbox"/> |
| Median LCL | <input type="checkbox"/> |

Signals

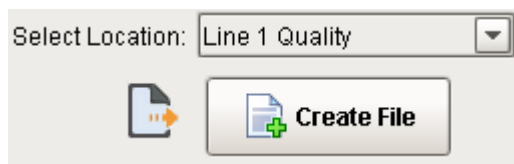
| Name | Enable |
|--------------------------|-------------------------------------|
| c Nelson Rule 1 | <input type="checkbox"/> |
| Individual Nelson Rule 1 | <input type="checkbox"/> |
| Individual Nelson Rule 2 | <input type="checkbox"/> |
| Individual Nelson Rule 3 | <input type="checkbox"/> |
| Individual Nelson Rule 4 | <input type="checkbox"/> |
| Individual Nelson Rule 5 | <input type="checkbox"/> |
| Individual Nelson Rule 6 | <input type="checkbox"/> |
| Individual Nelson Rule 7 | <input type="checkbox"/> |
| Individual Nelson Rule 8 | <input type="checkbox"/> |
| Individual Outside | <input checked="" type="checkbox"/> |
| Median Nelson Rule 1 | <input type="checkbox"/> |
| np Nelson Rule 1 | <input type="checkbox"/> |
| Out of Limits | <input type="checkbox"/> |
| Outside Limits | <input type="checkbox"/> |
| p Nelson Rule 1 | <input type="checkbox"/> |

- Drag a **Location Selector** from the Production Component palette onto your **File Monitoring** window.
- Add a button and add the following script to the **ActionedPerformed** event:

```
import system
fileName = "c:\\temp\\parseMe.txt"
import random
random.seed()
dia = random.randint(0, 9) / 10.0 + 25
x = random.randint(0, 9) / 10.0 + 100
y = random.randint(0, 9) / 10.0 + 50
data = "Measurement Data\nLocation-X: %f\nLocation-Y: %f\nDiameter: %f" % (x, y, dia)
system.file.writeFile(fileName, data)
```

- Add the expression **{Root Container.Location Selector.selectedIndex} != -1** to the enabled property of the button.

Whenever we press the button, a new file will be created that will randomly generate values for the Measurement attribute.



We'll now add script to create an SPC sample whenever the file is parsed.

- Add the following script to the **onAfterParse** event on the **File Monitor Controller**:

```
if event.source.parent.getComponent('Location Selector').
selectedIndex != -1:
    results = event.getParseResults()
    if results.isValid():
        print "New sample file found. Creating sample from data."
        locationX = results.getValue("LocationX")
        locationY = results.getValue("LocationY")
        diameter = results.getValue("Diameter")
```

```

        location = event.source.parent.getComponent('Location
Selector').selectedLocationPath #Location where the sample is
created
        sampleDef = "Measurement"
        sample = system.quality.sample.data.createSampleByName
(' ', sampleDef, location)
        sample.setSampleData(1, "LocationX", str(locationX))
        sample.setSampleData(1, "LocationY", str(locationY))
        sample.setSampleData(1, "Diameter", str(diameter))
        sample.setApproved(1)
        system.quality.sample.data.updateSample(location, sample,
1)
    else:
        print "No location selection to store data"

```

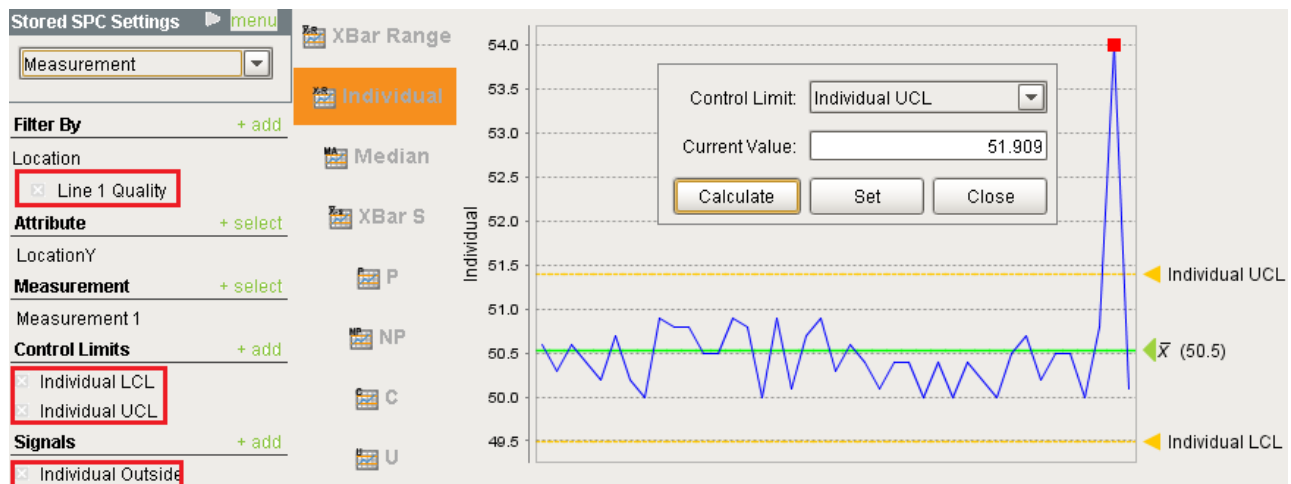
- Press your **Create File** button a few times to create some samples.

We can now view the samples being created from the file in the Control Charts window.

- Open the **Control Charts** window under **Quality**.
- Select the **Measurement** sample definition.
- Select the **Individual Chart Type**
- You may need to add **Location:Line 1 Quality** to the *Filter By* setting

You should see your sample points on the screen. If you right-click on the chart to bring up the **Control Limits** settings box, no control limits show up. Why Not? We need to add the control limits that we want to return data for in the Stored SPC Setting.

- Add **Individual LCL** and **Individual UCL** to the *Control Limits* setting.
- **Save** your changes by selecting menu>save on the *Stored SPC Settings* box.



Now you can right-click on the chart to bring up the **Control Limits** settings box. Go ahead and select **Individual UCL** and **Individual LCL** and press the **Calculate** button. This button causes the **Individual LCL** and **UCL** python script as defined in the Production Designer to run for the sample datapoints shown in chart.

[Click here to see Individual UCL python script...](#)

```
#Individual UCL Calculation
#Get the SPC data that the Individual UCL will be calculated for
ds = event.getData()
#Get the columnn index within the SPC data
xBarColNdx = ds.getColumnIndex("XBar")
mrColNdx = ds.getColumnIndex("MR")
#Initialize XBar and moving range sums that are need to calculate
average Xbar and moving range.
xBarSum = 0.0
mrSum = 0.0
#Cycle through each row and add to the sum
for row in range(ds.rowCount):
    xBarSum = xBarSum + ds.getValueAt(row, xBarColNdx)
    mrVal = ds.getValueAt(row, mrColNdx)
    if mrVal <> None:
        mrSum = mrSum + mrVal
#Calculate the average XBar and moving range
xDBar = xBarSum / ds.rowCount
mrBar = mrSum / (ds.rowCount - 1)
#Calculate the Individual UCL
ucl = xDBar + 2.66 * mrBar
#Return the new Individual UCL back to the SPC module
event.setControlLimitValue(ucl)
```

8.5 Instrument Interface Review

This concludes the Instrument Interface portion of the tutorial.

8.5.1 What We Covered

In summary, we covered the following:

- [Instrument Interface Overview](#)
- [Setting up File Monitoring](#)
- [Using Instrument Interface with the SPC module](#)

8.5.2 What We Didn't Cover

What we haven't covered or touched in great detail are:

- [Serial Port Monitoring](#)

8.5.3 Additional Resources

We can't cover everything in a tutorial, but we do have more resources available in the [MES Help manual](#) and [Sepasoft Knowledge Base](#) for the areas we didn't cover.

9 Training Complete

Congratulations! You have completed the online MES training and built a project that encompasses most of the base functionality of our MES modules. This should give you a good starting point in creating your own applications using our modules. We did not cover everything here. The modules provide a significant amount of functionality. We will add more to the training over time. For more information on advanced capabilities, and for implementing custom solutions that are a little different, you can refer to our [Knowledge Base Articles](#), our [Help Manual](#), our [online videos](#), and of course contact us at support@sepassoft.com.

Good Luck!

The Sepasoft Team



Built On **Ignition!**
by inductive automation

For Sales & Demos, contact Inductive Automation:
800.266.7798 | info@inductiveautomation.com