TrakSYS™ Real Time Operations & Performance Management

TrakSYS™ 7.0 Advanced Training Course Lab Manual

Revised 7/1/2011

Parsec Automation Corp.
180 North Riverview, No. 300
Anaheim, CA 92808 USA
Phone +1 714 996 5302
Fax +1 714 996 1845
www.parsec-corp.com

Contents

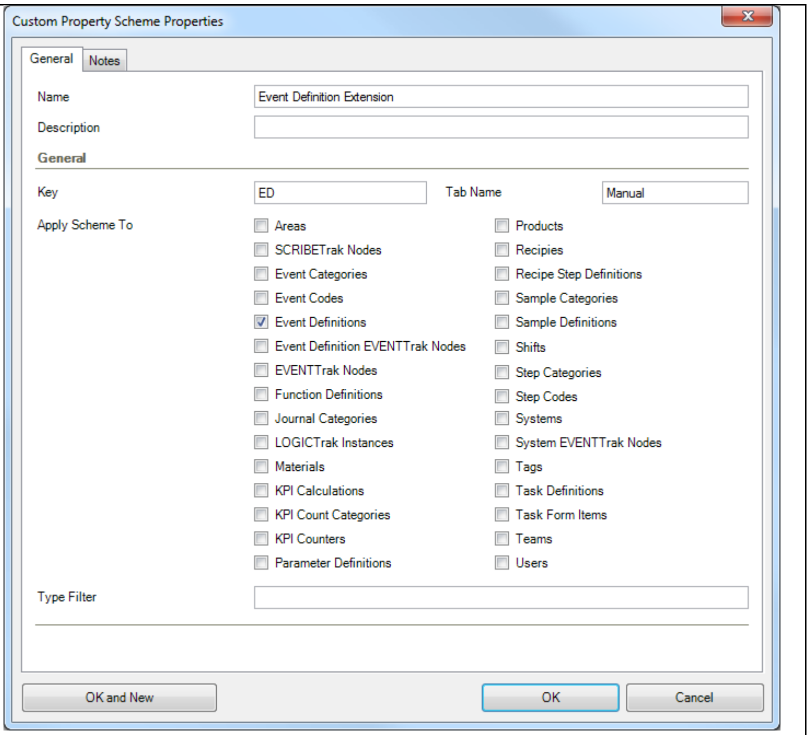
This document is for training purposes only. Parsec Automation makes no warranties, express or implied, in this summary. Parsec and TrakSYS are either registered trademarks or trademarks of Parsec Automation Corporation in the United States and/or other countries. The name of actual companies and products mentioned in here may be the trademarks of their respective owners.

Lab 01: MODELTrak – Custom Properties

Objectives	• Create Custom Property Schemes • Add Custom Settings to Event Definitions • Query Custom Values from the Database
Estimated Time to Complete This Lab	20 minutes

In this lab, you will extend the configuration of MODELTrak Event Definitions by adding new settings through Custom Properties. You will then learn how to retrieve Custom Property data by querying data views in the TrakSYS database.

Tasks	Detailed Steps
Open TrakSYS MODELTrak	1. Go to **Start | All Programs | Parsec | TrakSYS | TrakSYS MODELTrak**.
Create a Custom Property Scheme and Group	1. Go to the **Custom Properties** panel in the MODELTrak tree view. 2. Right-click on the **Custom Property Schemes** Group and select **New Custom Property Scheme** from the popup menu. 3. In the **Custom Property Scheme Properties** page, type **Event Definition Extension** for the **Name**, **ED** for the **Key**, and **Manual** for the **Tab Name**. 4. Check the box for **Event Definitions** in the **Apply Scheme To** list. 5. Click **OK** to save this Custom Property Scheme. 6. Right-click on the **Event Definition Extension** Scheme in the tree view and select **New Custom Property Group** from the popup menu. 7. In the **Custom Property Group Properties** page, type **General** for the **Name**. 8. Click **OK** to save this Custom Property Group. 9. Right-click on the **General** Custom Property Group in the tree view and select **New Custom Property**. 10. In the **Custom Property Properties** page, type **Enable Manual Event Button** for the **Name**, **ET.BUTTON** for the **Key**, **Discrete** for the **Data Type**, and **0** for the **Default Value**. 11. Click **OK** to save this Custom Property.



Custom Property Scheme Properties

Name: Event Definition Extension

Description:

General

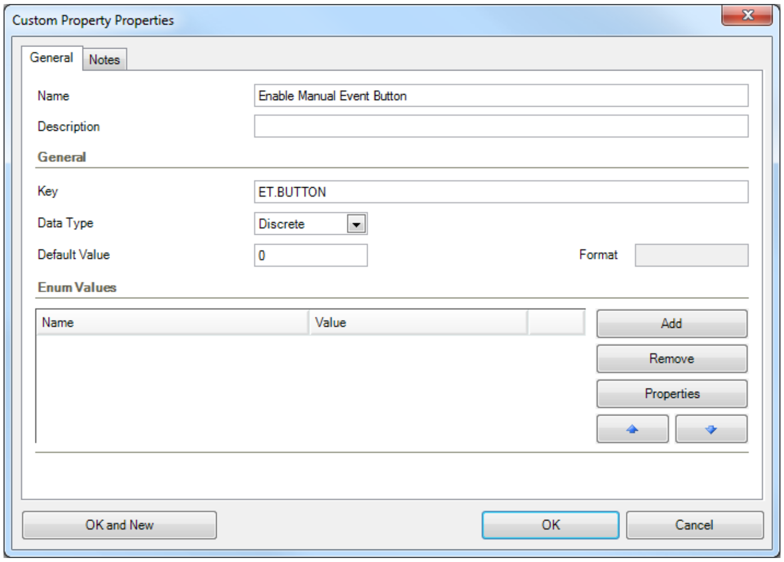
Key: ED

Tab Name: Manual

Apply Scheme To:

- □ Areas
- □ SCRIBETrak Nodes
- □ Event Categories
- □ Event Codes
- ☑ Event Definitions
- □ Event Definition EVENTTrak Nodes
- □ EVENTTrak Nodes
- □ Function Definitions
- □ Journal Categories
- □ LOGICTrak Instances
- □ Materials
- □ KPI Calculations
- □ KPI Count Categories
- □ KPI Counters
- □ Parameter Definitions
- □ Products
- □ Recipies
- □ Recipe Step Definitions
- □ Sample Categories
- □ Sample Definitions
- □ Shifts
- □ Step Categories
- □ Step Codes
- □ Systems
- □ System EVENTTrak Nodes
- □ Tags
- □ Task Definitions
- □ Task Form Items
- □ Teams
- □ Users

Type Filter:



Custom Property Properties

Name: Enable Manual Event Button

Description:

General

Key: ET.BUTTON

Data Type: Discrete

Default Value: 0

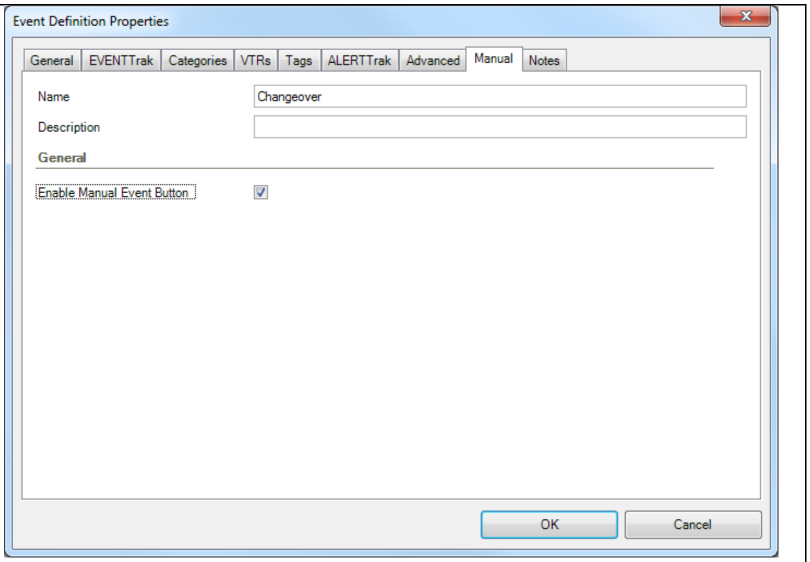
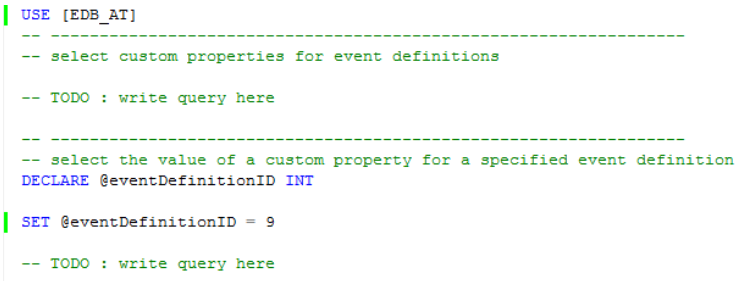
Format:

Enum Values

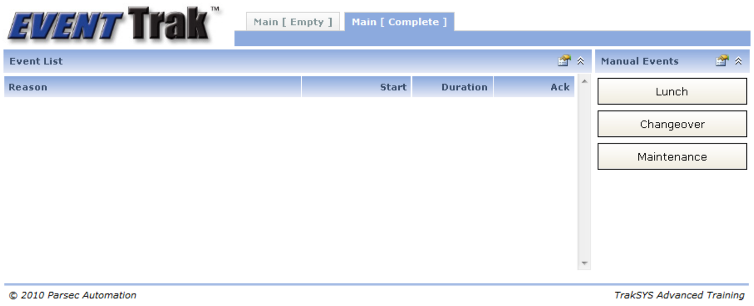
Name	Value

Add Custom Property Values

1. Go to the **Areas and Systems** panel in the MODELTrak tree view.
2. Expand the tree view to find the **Line 1** System under the **Packaging** Area and select it. In the detail view, right-click the **Changeover** Event Definition and select **Properties**.
3. In the **Event Definition Properties** page, select the **Manual** tab, check the box for **Enable Manual Event Button**, and click OK.

	 Event Definition Properties General EVENTTrak Categories VTRs Tags ALERTTrak Advanced Manual Notes OK Cancel Name: Changeover Description: General Enable Manual Event Button ☑ 4. Repeat steps 2 and 3 for the **Lunch** and **Maintenance** Event Definitions.
Open SQL Server Management Studio	1. Go to **Start | All Programs | Microsoft SQL Server 2005 | SQL Server Management Studio**.
Execute a Query	1. From the **File** menu, select **Open | File…**, then select the file **Custom Property Queries 1 [Empty].sql** from the **Files\Custom Properties** folder on the computer desktop. 2. If prompted by the Management Studio, connect to the local SQL Server instance on the computer. 3. Write a **SQL query** in the open query window that lists the following fields for Event Definitions. Use the database view **vwCustomPropertyEventDefinition** to return these fields. • **EventDefinitionID** • **EventDefinitionName** • **ED.ET.BUTTON** 4. Write a **SQL query** in the open query window that returns the value for **Enable Manual Event Button** for a specific Event Definition. Use the **EventDefinitionID** provided in the query template. 

```
USE [EDB_AT]
-- -----------------------------------------------------------------
-- select custom properties for event definitions

-- TODO : write query here

-- -----------------------------------------------------------------
-- select the value of a custom property for a specified event definition
DECLARE @eventDefinitionID INT

SET @eventDefinitionID = 9

-- TODO : write query here
```

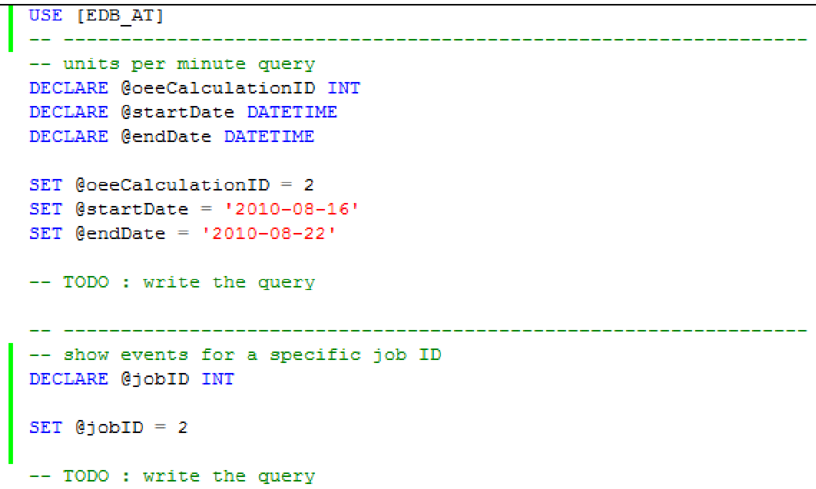
	5. A possible solution for steps 3 and 4 may be found in the file **Custom Property Queries 2 [Complete].sql**, located in the **Files\Custom Properties** folder on the computer desktop.
Use Custom Properties in a Dashboard Web Part	1. Open **WEBTrak** and navigate to the Report Group **TrakSYS Reports\TrakSYS Advanced Training\Custom Properties**. 2. Open the **Main [Empty]** dashboard report and click the **Edit Dashboard** button on the menubar. 3. Edit the **Manual Events** HTML Content web part and go to the **Source** panel of the Web Part Settings. 4. Write a **SQL Query** for the web part's Data Source that will list the Manual Event buttons for a specific System. Use the database tables **tEventDefinition** and **tTag** in your query, along with the database view **vwCustomPropertyEventDefinition**. The query should return the following fields, where the Tag fields are based on the Trigger Tag for a given Event Definition: - EventDefinitionID - EventDefinitionName - TagID - TagName - TagValue 5. A possible solution can be found in the **Main [Complete]** dashboard.  EVENT Trak™ Main [Empty] Main [Complete] Event List Reason Start Duration Ack Manual Events Lunch Changeover Maintenance © 2010 Parsec Automation TrakSYS Advanced Training

Lab 02: TrakSYS Data Structures

Objectives	• Create Queries Against the TrakSYS Data Views
Estimated Time to Complete This Lab	25 Minutes

In this lab, you will write different queries to retrieve data from the TrakSYS database using the standard TrakSYS data views for KPI and Event data. These data views are used frequently in custom KPI and report generation.

Tasks	Detailed Steps
Create Tag Groups	1. Go to **Start | All Programs | Microsoft SQL Server 2005 | SQL Server Management Studio**. 2. Connect to the local database server using **Windows Authentication**. 3. From the **File** menu, select **Open | File…**, then select the file **TrakSYS Data Structures 1 [Empty].sql** from the **Files\TrakSYS Data Structures** folder on the computer desktop. 4. If prompted by the Management Studio, connect to the local SQL Server instance on the computer. 5. Use the document **TrakSYS(TM) Data View Descriptions.pdf** to create a **SQL query** in the open query window for a Custom KPI called **Units per Minute**. • The KPI should be the **Total** count divided by the **Net Operating Time** for a specified date range. • Use the SQL parameters for **OeeCalculationID**, **StartDate**, and **EndDate** defined in the query template. • Use the data view **vwOeeInterval** to complete this query. 6. Create a **SQL query** in the open query window that **Event** data for a specific **Job ID**. • Use the **JobID** provided in the query template. • Use the data view **vwEvent** to complete this query. The query result set should contain the following fields: • EventID • EventStartDateTime • EventEndDateTime • SystemName • SubSystemName • EventDefinitionName • ShiftName • ProductName



```
USE [EDB_AT]
-- ----------------------------------------------------------------
-- units per minute query
DECLARE @oeeCalculationID INT
DECLARE @startDate DATETIME
DECLARE @endDate DATETIME

SET @oeeCalculationID = 2
SET @startDate = '2010-08-16'
SET @endDate = '2010-08-22'

-- TODO : write the query

-- ----------------------------------------------------------------
-- show events for a specific job ID
DECLARE @jobID INT

SET @jobID = 2

-- TODO : write the query
```

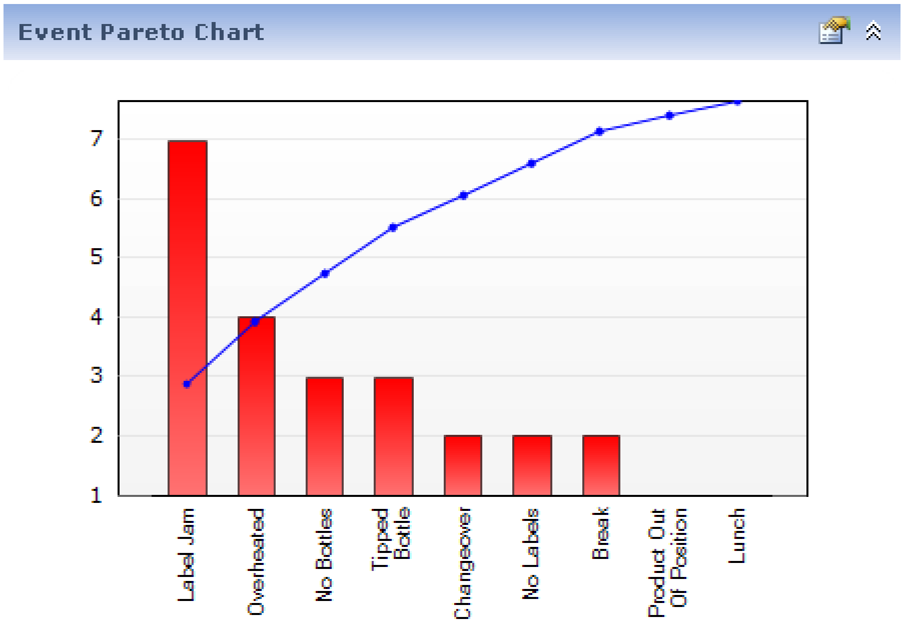
7. A possible solution for steps 4 and 5 may be found in the file **TrakSYS Data Structures 2 [Complete].sql** from the **Files\TrakSYS Data Structures** folder on the computer desktop.

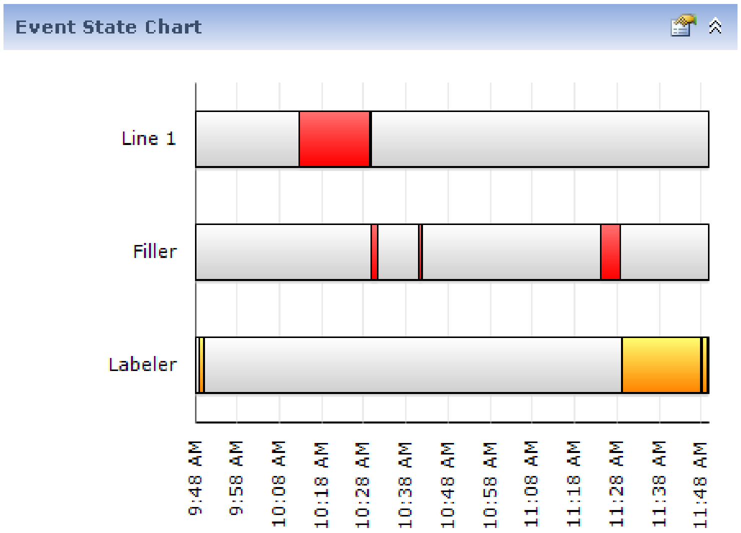
Lab 03: WEBTrak – Dashboard Data Sources

Objectives	• Configure Event Chart Web Parts • Configure a Custom Web Part Source Query
Estimated Time to Complete This Lab	35 minutes

In this lab, you will create a dashboard in WEBTrak that contains an Event Pareto web part. The Pareto Chart will be populated using a custom query that retrieves Event data from the TrakSYS database.

Tasks	Detailed Steps
Edit a Dashboard Report	1. Open **WEBTrak** from **Start | All Programs | Parsec | TrakSYS | WEBTrak**. 2. Click the **Login** button on the menubar to login as the **Adminstrator** (**admin / sa**). Check the **Remember Me** box to cache the login. 3. Navigate to the Report Group **TrakSYS Reports\TrakSYS Advanced Training\Dashboard Data Sources**. 4. Open the **Data Sources [Empty]** dashboard report and click the **Edit Dashboard** button on the menubar.
Add an Event Pareto Chart Web Part with a Custom Query	1. Click the **Add a Web Part** button above the left **Zone** on row 2 of the dashboard. 2. From the 1st column of the web part catalog, select the **Event Pareto Chart** web part. 3. Go to the settings for this web part. In the **Appearance** panel: a. For **Title** type **Event Pareto Chart**. b. Select **Title Only** for **Border Style**. c. Enter **450** for the **Width**. d. Enter **300** for the **Height**. 4. In the **Data** panel: a. Select **User Defined** for **Date/Time Source**. b. For **Group By** pick **Event Definition**. c. For **Metric Type** select **By Count**. 5. In the **Advanced** panel: a. Enter **65** for **X Axis Label Height**. 6. In the **Source** panel: a. Select **SQL** for **Mode**. 7. Write a **SQL Query** for the web part's Data Source that returns all events grouped by **Event Definition** for a specified date range. a. The value for each bar should be the total event count for that Event Definition. b. The custom query should filter the events by the starting and ending date/times of the assigned picker. c. The query for the Pareto Chart must include the following fields: • DataLabel : String • DataValue : Numeric (Integer or Float) • DataColor : null 8. Click **OK** to apply these settings.

	9. Click **View Dashboard** to see the report in view mode.  Event Pareto Chart 7 6 5 4 3 2 1 Label Jam Overheated No Bottles Tipped Bottle Changeover No Labels Break Product Out Of Position Lunch
Add an Event State Chart Web Part with a Custom Query	1. Click the **Add a Web Part** button above the right **Zone** on row 2 of the dashboard. 2. From the 1st column of the web part catalog, select the **Event State Chart** web part. 3. Go to the settings for this web part. In the **Appearance** panel: a. Select **Title Only** for the **Border Style**. b. Enter **450** for the **Width**. c. Enter **300** for the **Height**. d. Select **No** for **Chart Legend Visible**. 4. In the **Data** panel: a. Select **Current Day** for the **Date/Time Source**. 5. In the **Advanced** panel: a. Enter **55** for **X Axis Height**. b. Enter **LightGray** for **Default Slice Color**. 6. In the **Source** panel: a. Select **SQL** for **Mode**. 7. Write a **SQL Query** for the web part's Data Source that returns all events grouped by **Sub System** for a specified date range. a. The custom query should filter the events by the starting and ending date/times of the assigned picker. b. Use the document **TrakSYS 7.0 Web Part Data Interface.pdf** to determine the fields that the query must return for the Event State Chart. 8. Click **OK** to apply these settings. 9. Click **View Dashboard** to see the report in view mode.

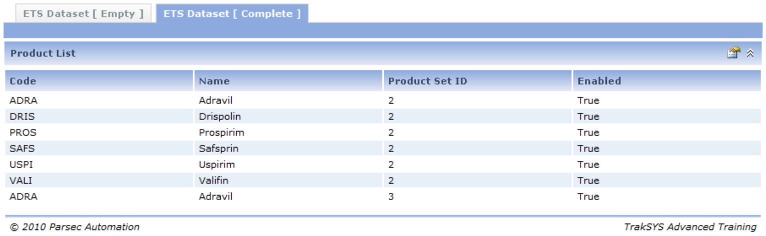


10. A possible solution can be found in the **Data Sources [Complete]** dashboard.

Lab 04: WEBTrak – Dashboard Content Expressions

Objectives	• Use <ets_dataset> with a custom query • Use <ets_repeat> to output custom content
Estimated Time to Complete This Lab	25 minutes

In this lab, you will use content expressions to output tabular content from a custom query. The query is to be embedded within an HTML Content web part using the <ets_dataset> Tag. The output from the query will be displayed in the web part using the <ets_repeat> Tag to loop through the query results.

<table>
<tr><th>Tasks</th><th>Detailed Steps</th></tr>
<tr><td>Edit a Dashboard Report</td><td>1. Open <b>WEBTrak</b> and navigate to the Report Group <b>TrakSYS Reports\TrakSYS Advanced Training\Dashboard Content Expressions</b>.
2. Open the <b>ETS Dataset [Empty]</b> dashboard report and click the <b>Edit Dashboard</b> button on the menubar.</td></tr>
<tr><td>Add an HTML Content Web Part with a Custom Query</td><td>1. Add an <b>HTML Content</b> web part to an empty <b>Zone</b> on the Dashboard.
2. Open the <b>Web Part Settings</b> for the HTML Content web part and click the <b>Editor</b> or <b>Text</b> button to edit the HTML.
3. Add a custom query to the HTML using the <b><ets_dataset></b> Tag. The custom query should return data for all <b>Products</b> listed in the TrakSYS database. The custom query should return the following fields:
• ProductCode
• Name
• ProductSetID
• Enabled
4. Add an <b>HTML table</b> to the web part with 4 columns, one for each field in the custom query.
5. Use the <b><ets_repeat></b> Tag with the HTML table to loop through all of the rows returns by the custom query.
6. Click <b>OK</b> to save the HTML content and custom query and to apply all settings to the web part.
7. Click <b>View Dashboard</b> to view the results from the custom query.
8. A possible solution can be found in the <b>ETS Dataset [Complete]</b> dashboard.


Product List<table><tr><th>Code</th><th>Name</th><th>Product Set ID</th><th>Enabled</th></tr><tr><td>ADRA</td><td>Adravil</td><td>2</td><td>True</td></tr><tr><td>DRIS</td><td>Drispolin</td><td>2</td><td>True</td></tr><tr><td>PROS</td><td>Prospirim</td><td>2</td><td>True</td></tr><tr><td>SAFS</td><td>Safsprin</td><td>2</td><td>True</td></tr><tr><td>USPI</td><td>Uspirim</td><td>2</td><td>True</td></tr><tr><td>VALI</td><td>Valifin</td><td>2</td><td>True</td></tr><tr><td>ADRA</td><td>Adravil</td><td>3</td><td>True</td></tr></table>© 2010 Parsec Automation
TrakSYS Advanced Training</td></tr>
</table>

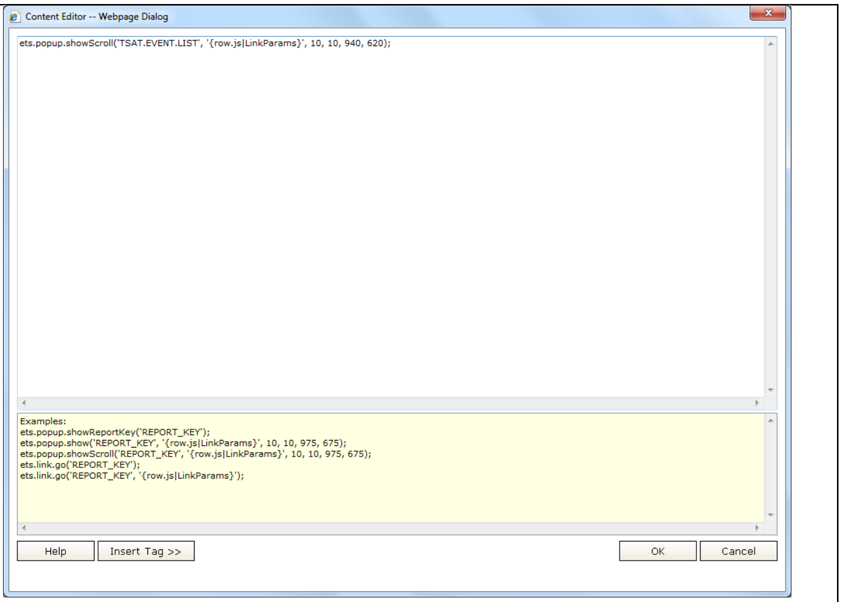
Lab 05: WEBTrak – Advanced Dashboard Techniques

Objectives	• Add a report link to an existing Event Pareto Chart • Add a report link to an existing Event State Chart
Estimated Time to Complete This lab	30 minutes

In this lab, you will learn how to create drilldown links for different web parts on a dashboard. You will add Links to existing Event Charts so that clicking a bar or slice on the chart will open a drilldown report with additional details. This lab will utilize Report Keys to identify the report to open from a web part link.

Tasks	Detailed Steps
Edit a Dashboard Report	1. Open **WEBTrak** and navigate to the Report Group **TrakSYS Reports\TrakSYS Advanced Training\Dashboard Advanced Techniques**. 2. Open the **Report Linking [Empty]** dashboard report and click the **Edit Dashboard** button on the menubar.
Add a Link to an Event Pareto Chart	1. Click the **Edit/Properties** button on the Event Pareto Chart to open the **Web Part Settings** list. 2. In the **Advanced Panel**, click the "…" button next to the **Link** setting to open a **Content Editor** window. 3. Create a drilldown link for the chart by copy-and-pasting one of the example links listed at the bottom of the Content Editor window. For the new link: • Replace the default '**REPORT_KEY**' with the value '**TSAT.EVENT.LIST**'. This will be the Report Key assigned to the drilldown report. • If the example link used contains settings to size the window, modify these settings so that the drilldown report opens in a window that is **940x620** pixels. 4. Click the **OK** button to close the **Content Editor** window and save the new drilldown value to the Link setting. 5. Click **OK** to apply the new settings to the web part.

<table>
<tr>
<td></td>
<td>

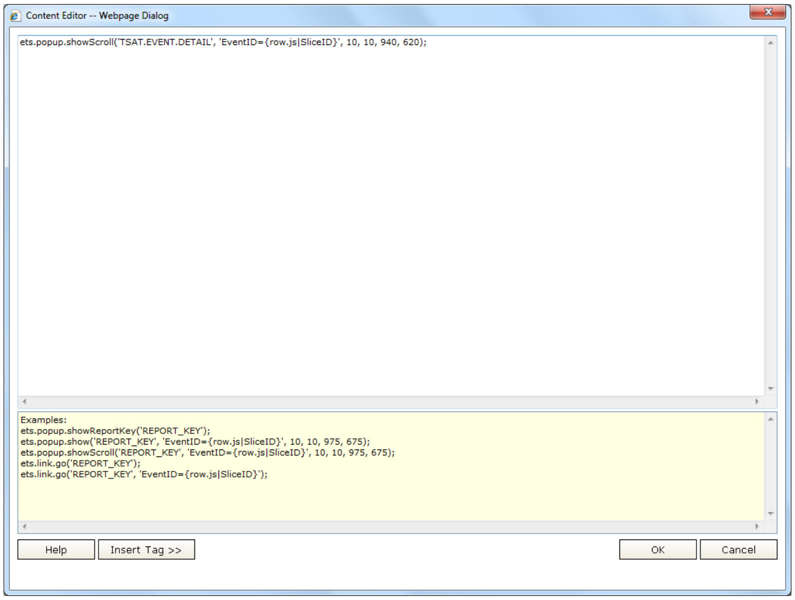


```
ets.popup.showScroll('TSAT.EVENT.LIST', '{row.js|LinkParams}', 10, 10, 940, 620);
```

```
Examples:
ets.popup.showReportKey('REPORT_KEY');
ets.popup.show('REPORT_KEY', '{row.js|LinkParams}', 10, 10, 975, 675);
ets.popup.showScroll('REPORT_KEY', '{row.js|LinkParams}', 10, 10, 975, 675);
ets.link.go('REPORT_KEY');
ets.link.go('REPORT_KEY', '{row.js|LinkParams}');
```

</td>
</tr>
<tr>
<td>Add a Link to an Event State Chart</td>
<td>

1. Repeat steps 1-5 listed above to add a link to the **Event State Chart**.
2. Link the Event State Chart to a dashboard with a **Report Key** of ‘**TSAT.EVENT.DETAIL**’.



```
ets.popup.showScroll('TSAT.EVENT.DETAIL', 'EventID={row.js|SliceID}', 10, 10, 940, 620);
```

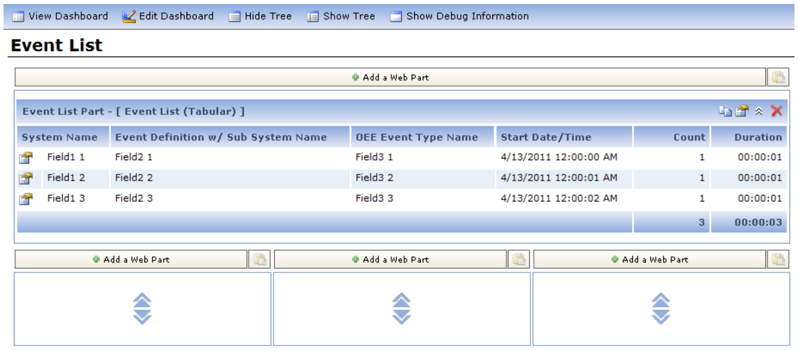
```
Examples:
ets.popup.showReportKey('REPORT_KEY');
ets.popup.show('REPORT_KEY', 'EventID={row.js|SliceID}', 10, 10, 975, 675);
ets.popup.showScroll('REPORT_KEY', 'EventID={row.js|SliceID}', 10, 10, 975, 675);
ets.link.go('REPORT_KEY');
ets.link.go('REPORT_KEY', 'EventID={row.js|SliceID}');
```

</td>
</tr>
<tr>
<td>Create Event List Dashboard</td>
<td>

1. In the **[Dialog]** Report Group under **Dashboard Advanced Techniques**, add a new **Dashboard** report.
2. In the **Report Design** window for the new dashboard, enter **Event List** for the **Name**.
3. In the **Dashboard (Advanced)** section, enter **TSAT.EVENT.LIST** for the **Key**.
4. In the **General** section, set **Show Header** to **No**.
5. Click **Save** to save all changes to the report. Click **Open** to view the dashboard.

</td>
</tr>
</table>

6. Edit the dashboard and add an **Event List (Tabular)** web part to an empty zone.
7. In the **Web Part Settings** for the Event List (Tabular) control, enter **TSAT.EVENT.DETAIL** for **Link Report Key**, and set **Use Linked Parameters** to **Yes**.
8. Click **OK** to save all changes to the dashboard.



Event List

Event List Part - [Event List (Tabular)]

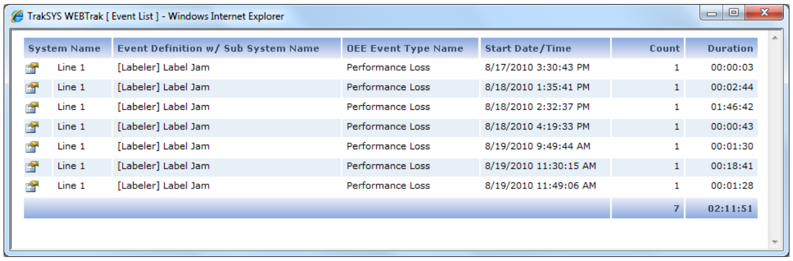
	System Name	Event Definition w/ Sub System Name	OEE Event Type Name	Start Date/Time	Count	Duration
	Field1 1	Field2 1	Field3 1	4/13/2011 12:00:00 AM	1	00:00:01
	Field1 2	Field2 2	Field3 2	4/13/2011 12:00:01 AM	1	00:00:01
	Field1 3	Field2 3	Field3 3	4/13/2011 12:00:02 AM	1	00:00:01
					3	00:00:03

Create Event Detail Dashboard

1. In the **[Dialog]** Report Group under **Dashboard Advanced Techniques**, add a new **Dashboard** report.
2. In the **Report Design** window for the new dashboard, enter **Event Detail** for the **Name**.
3. In the **Dashboard (Advanced)** section, enter **TSAT.EVENT.DETAIL** for the **Key**.
4. In the **General** section, set **Show Header** to **No**.
5. Click **Save** to save all changes to the report. Click **Open** to view the dashboard.
6. Edit the dashboard and add an **Event Detail (Tabular)** web part to an empty zone.
7. Click **OK** to save all changes to the dashboard.

Test the Report Link

1. Open the **Report Linking [Empty]** dashboard and click any of the bars on the **Event Pareto Chart** to test the report link. The **Event List** dashboard should open with data for the bar that was clicked.
2. Click a slice on the **Event State Chart** to test the report link. The **Event Detail** dashboard should open with data for the event that was clicked.
3. A possible solution can be found in the **Report Linking [Complete]** dashboard.



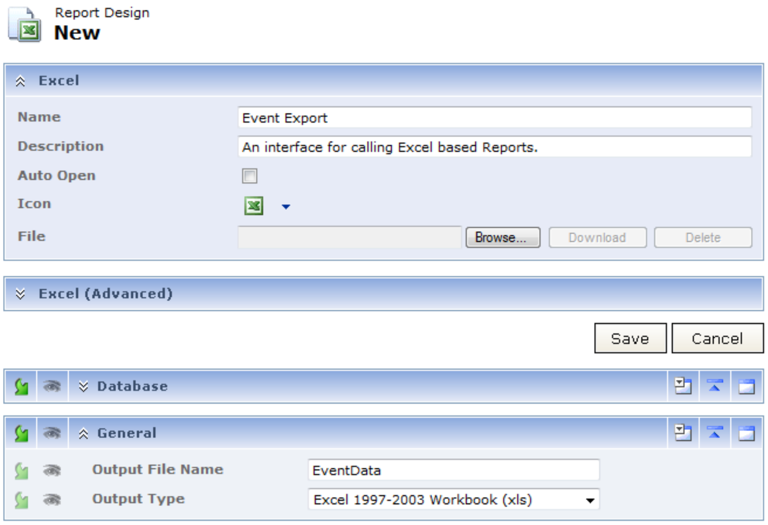
TrakSYS WEBTrak [Event List] - Windows Internet Explorer

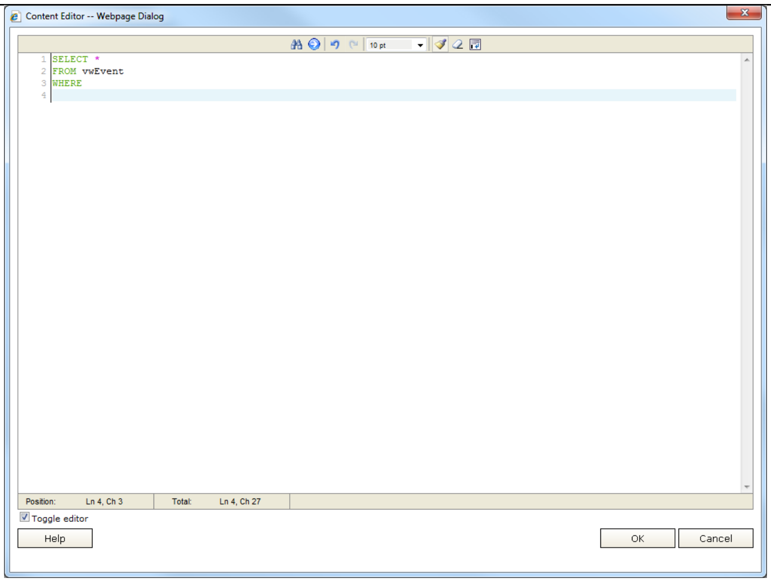
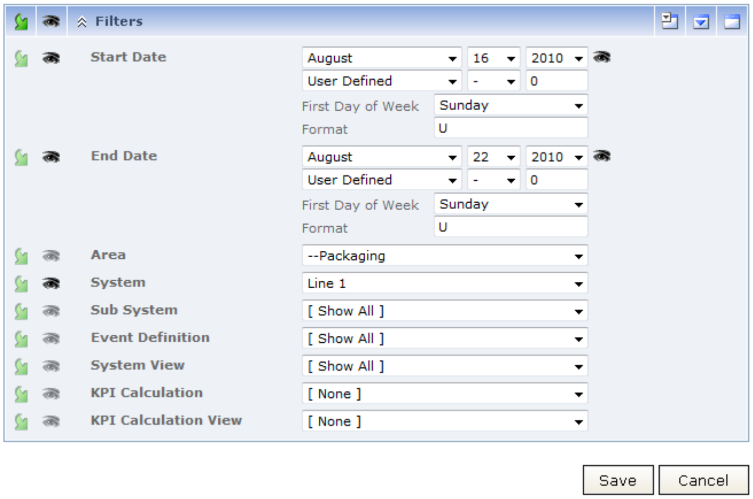
	System Name	Event Definition w/ Sub System Name	OEE Event Type Name	Start Date/Time	Count	Duration
	Line 1	[Labeler] Label Jam	Performance Loss	8/17/2010 3:30:43 PM	1	00:00:03
	Line 1	[Labeler] Label Jam	Performance Loss	8/18/2010 1:35:41 PM	1	00:02:44
	Line 1	[Labeler] Label Jam	Performance Loss	8/18/2010 2:32:37 PM	1	01:46:42
	Line 1	[Labeler] Label Jam	Performance Loss	8/18/2010 4:19:33 PM	1	00:00:43
	Line 1	[Labeler] Label Jam	Performance Loss	8/19/2010 9:49:44 AM	1	00:01:30
	Line 1	[Labeler] Label Jam	Performance Loss	8/19/2010 11:30:15 AM	1	00:18:41
	Line 1	[Labeler] Label Jam	Performance Loss	8/19/2010 11:49:06 AM	1	00:01:28
					7	02:11:51

Lab 06: WEBTrak – Excel Reports

Objectives	• Create an Excel Report to export data • Create a formatted Excel Report
Estimated Time to Complete This lab	20 minutes

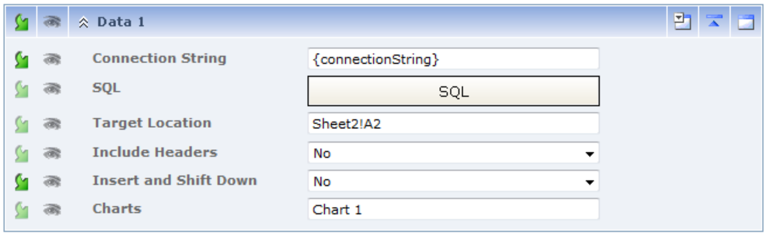
In this lab, you will learn how to use the Excel Report type in WEBTrak to export data from TrakSYS or from another external database. You will also import a pre-existing Excel file with a formatted report and merge it with a custom query from TrakSYS.

Tasks	Detailed Steps
Create a New Excel Report	1. Open **WEBTrak** and navigate to the Report Group **TrakSYS Reports\TrakSYS Advanced Training\Excel Reports**. 2. Add a new **Excel** report to the Report Group. 3. In the Report Design window for the new report, enter **Event Export** for the **Name**. 4. In the **General** section, enter **EventData** for the **Output File Name**, and set **Output Type** to **Excel 1997-2003 Workbook (xls)**.  Report Design New Excel Name: Event Export Description: An interface for calling Excel based Reports. Auto Open: □ Icon File Browse... Download Delete Excel (Advanced) Save Cancel Database General Output File Name: EventData Output Type: Excel 1997-2003 Workbook (xls)
Add a SQL Query to an Excel Report	1. In the **Data 1** section, click the **SQL** button to open a **Content Editor** dialog. 2. In the Content Editor dialog, enter a query that returns all **Event** records from database view **vwEvent** for a specific **System** and **Date Range**. • Use the values for **System**, **Start Date**, and **End Date** from the Report Designer's **Data** section in the query. • Add the parameters to the query using **Content Expressions**. • Return all fields from the database view. 3. Click the **OK** button to save the query to the report.

	
	Content Editor -- Webpage Dialog
	`SELECT *`
	`FROM vwEvent`
	`WHERE`
	Position: Ln 4, Ch 3 Total: Ln 4, Ch 27
	☑ Toggle editor
	4. In the **Filters** section:
	a. Select **August 16, 2010** for **Start Date**.
	b. Select **August 22, 2010** for **End Date**.
	c. Select **--Packaging** for **Area**.
	d. Select **Line 1** for **System**.
	e. Hide the parameters for **Area**, **Sub System**, **Event Definition**, **System View**, **KPI Calculation**, and **KPI Calculation View** using the **Show/Hide Parameter** icons next to each parameter.
	5. Click the **Save** button to save the new report to the WEBTrak tree view.
	
	Filters
	Start Date: August 16 2010; User Defined - 0; First Day of Week: Sunday; Format: U
	End Date: August 22 2010; User Defined - 0; First Day of Week: Sunday; Format: U
	Area: --Packaging
	System: Line 1
	Sub System: [Show All]
	Event Definition: [Show All]
	System View: [Show All]
	KPI Calculation: [None]
	KPI Calculation View: [None]

Add a Formatted Excel Template to an Excel Report	1. In the **Excel Reporting** Report Group, right-click the **Excel Chart [Empty]** Excel report and select **Design**. 2. In the **Excel Chart [Empty]** section, click the **Browse…** button and upload the file **ExcelChart.xls** from the **Files\Excel Reports** folder on the computer desktop.

3. In the **Data 1** section, click the **SQL** button to add a new query to the Excel report.
   a. The query should return the **RecordedDateTime** and **TagValueFloat** fields from the database table **tTagHistory**.
   b. The query should use the **Start Date** and **End Date** parameters from the **Report Designer** as filters in the WHERE clause.
4. In the **Data 1** section:
   a. Enter **Sheet2!A2** for **Target Location**.
   b. Set **Include Headers** to **No**.
   c. Set **Insert and Shift Down** to **No**.
   d. Enter **Chart 1** for **Charts**.
5. Click the **Save** button to save the report changes.

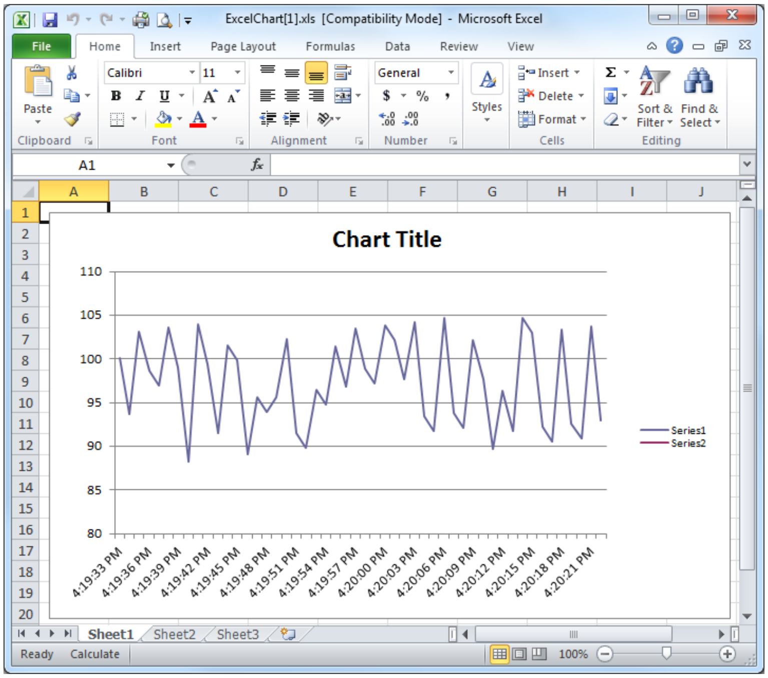


Data 1

Connection String	{connectionString}
SQL	SQL
Target Location	Sheet2!A2
Include Headers	No
Insert and Shift Down	No
Charts	Chart 1

View Reports

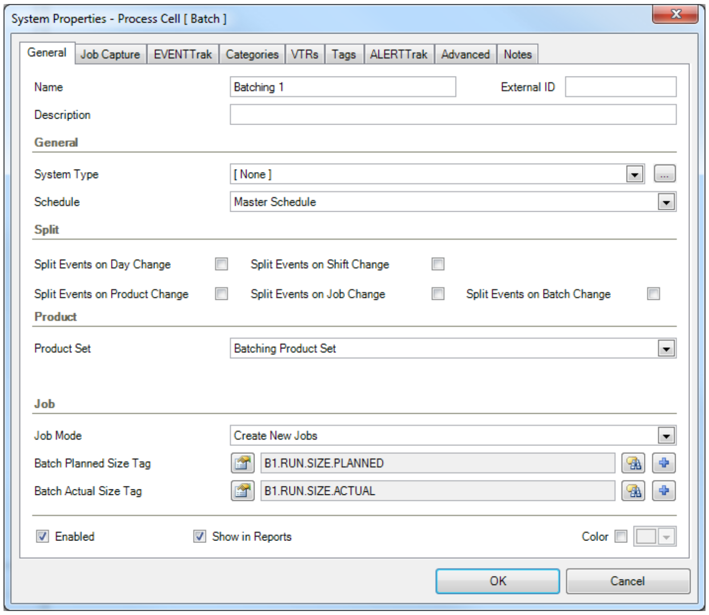
1. Click on and open each of the Excel reports to view them. Possible solutions for each report can be found in the **Event Report [Complete]** and **Excel Chart [Complete]** Excel reports.



Lab 07: MODELTrak – Batch Systems and Recipes

Objectives	• Configure a Batch System • Create Product Recipes
Estimated Time to Complete This lab	25 minutes

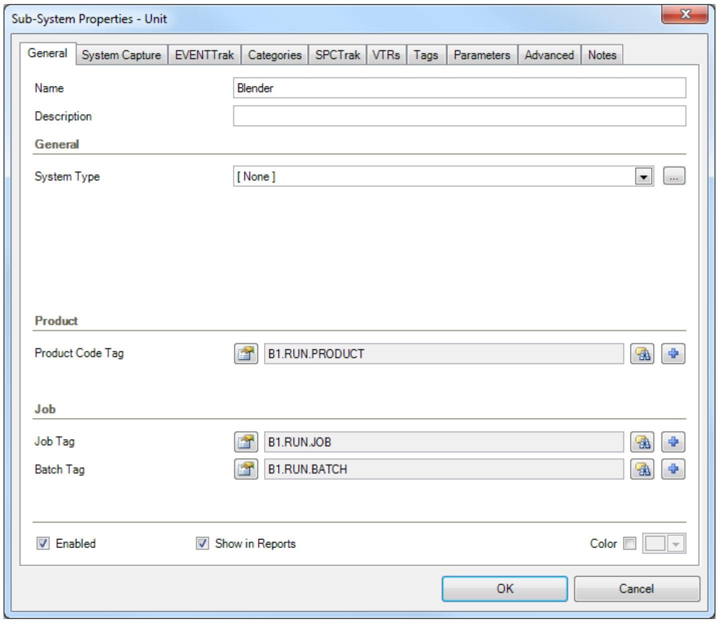
In this lab you will configure TrakSYS to record data for a batch process. You will create a new Batch System in MODELTrak and utilize Function Definitions, Parameters, and Tags to define what the process can do. You will then create a Recipe for an existing Product in PRODUCTTrak that specifies the sequence and duration of steps required to produce the Product.

Tasks	Detailed Steps
Open TrakSYS MODELTrak	1. Go to **Start | All Programs | Parsec | TrakSYS | TrakSYS MODELTrak**.
Create a System and Sub-System	1. Go to the **Areas and Systems** panel of MODELTrak. 2. Right-click on the **Batching** Area and select **New System [Batch]** from the popup menu. 3. In the **System Properties** page, type **Batching 1** for the **Name**. 4. Select **Master Schedule** for the **Schedule**. 5. Select **Batching Product Set** for the **Product Set**. 6. For the **Batch Planned Size Tag**, find and select **B1.RUN.SIZE.PLANNED**. 7. For the **Batch Actual Size Tag**, find and select **B1.RUN.SIZE.ACTUAL**. 8. Click **OK** to save this System.  System Properties - Process Cell [Batch] General Job Capture EVENTTrak Categories VTRs Tags ALERTTrak Advanced Notes Name: Batching 1 External ID: Description: General System Type: [None] Schedule: Master Schedule Split Split Events on Day Change □ Split Events on Shift Change □ Split Events on Product Change □ Split Events on Job Change □ Split Events on Batch Change □ Product Product Set: Batching Product Set Job Job Mode: Create New Jobs Batch Planned Size Tag: B1.RUN.SIZE.PLANNED Batch Actual Size Tag: B1.RUN.SIZE.ACTUAL ☑ Enabled ☑ Show in Reports Color □ 9. Right-click on the Batching 1 System in the tree view and select **New Sub-System [Unit]**. 10. In the **Sub-System Properties** page, type **Blender** for the **Name**.

11. Find and select the required tags for each of the following settings.

Setting	Tag Name
Product Code Tag	**B1.RUN.PRODUCT**
Job Tag	**B1.RUN.JOB**
Batch Tag	**B1.RUN.BATCH**

12. Click **OK** to save this Sub-System.



Sub-System Properties - Unit

Name: Blender

Description:

General

System Type: [None]

Product

Product Code Tag: B1.RUN.PRODUCT

Job

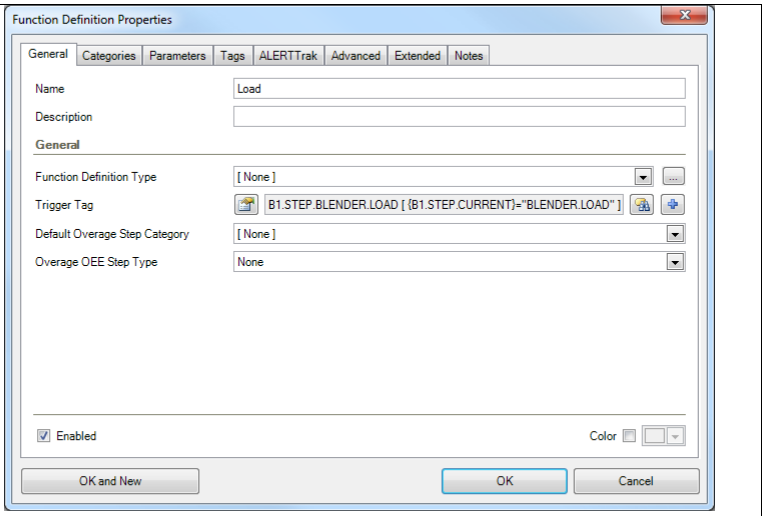
Job Tag: B1.RUN.JOB

Batch Tag: B1.RUN.BATCH

☑ Enabled ☑ Show in Reports Color □

Create Function Definitions

1. Right-click the **Blender** Sub-System and select **New Function Definition** from the menu list.
2. In the **Function Definition Properties,** type **Load** for the **Name** of the Function.
3. For the **Trigger Tag**, find and select **B1.STEP.BLENDER.LOAD**.
4. Click **OK** to save this Function Definition.



Name: Load

Description:

General

Function Definition Type: [None]

Trigger Tag: B1.STEP.BLENDER.LOAD [{B1.STEP.CURRENT}="BLENDER.LOAD"]

Default Overage Step Category: [None]

Overage OEE Step Type: None

☑ Enabled

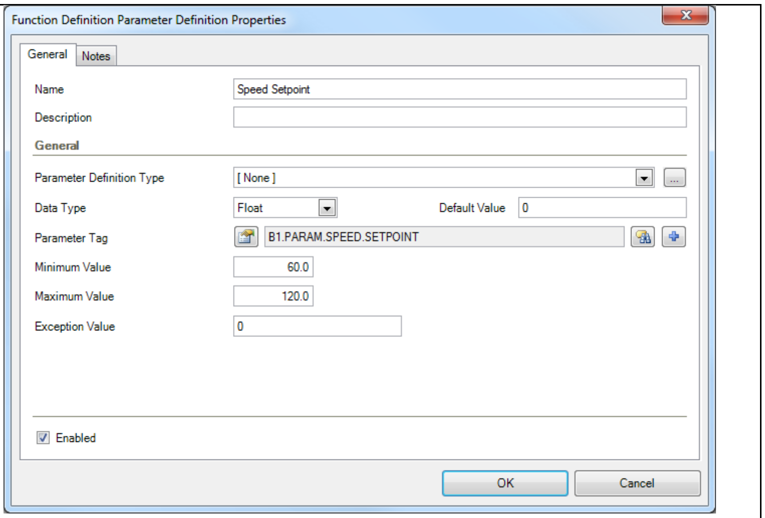
Color □

5. Repeat steps 1-4 to create the following Function Definitions.

Function Definition Name	Trigger Tag
Blend	**PL1.STEP.BLENDER.BLEND**
Unload	**PL1.STEP.BLENDER.UNLOAD**
Clean	**PL1.STEP.BLENDER.CLEAN**

Add a Parameter to a Function Definition

1. Right-click on the **Blend** Function Definition for the **Blender** Sub-System and select **Properties**.
2. In the **Function Definition Properties,** click the **Parameters** tab.
3. In the Parameters section, click the **New** button to add a new Parameter to the Function.
4. In the **Function Definition Parameter Definition Properties**, enter **Speed Setpoint** for the **Name**.
5. Select **Float** for the **Data Type**.
6. Enter **60.0** for the **Minimum Value** and **120.0** for the **Maximum Value**.
7. For the **Parameter Tag**, find and select **B1.PARAM.SPEED.SETPOINT**.
8. Click **OK** to save this Parameter Definition. Click OK again to save changes to the Function Definition.



Function Definition Parameter Definition Properties

General | Notes

Name: Speed Setpoint

Description:

General

Parameter Definition Type: [None]

Data Type: Float

Default Value: 0

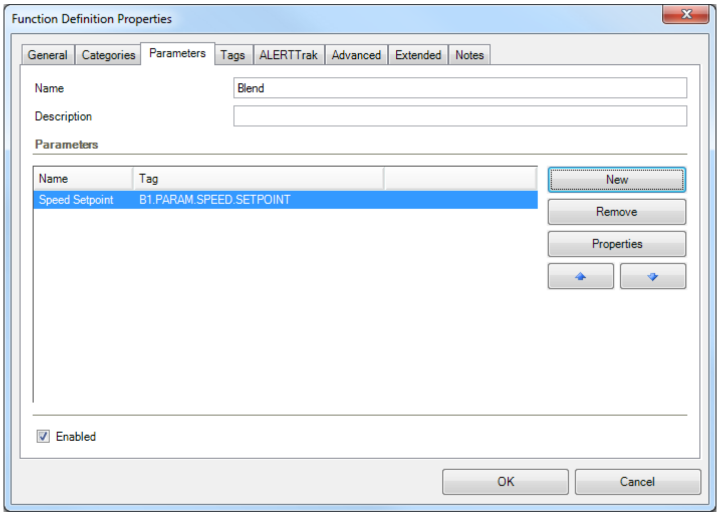
Parameter Tag: B1.PARAM.SPEED.SETPOINT

Minimum Value: 60.0

Maximum Value: 120.0

Exception Value: 0

☑ Enabled



Function Definition Properties

General | Categories | Parameters | Tags | ALERTTrak | Advanced | Extended | Notes

Name: Blend

Description:

Parameters

Name	Tag
Speed Setpoint	B1.PARAM.SPEED.SETPOINT

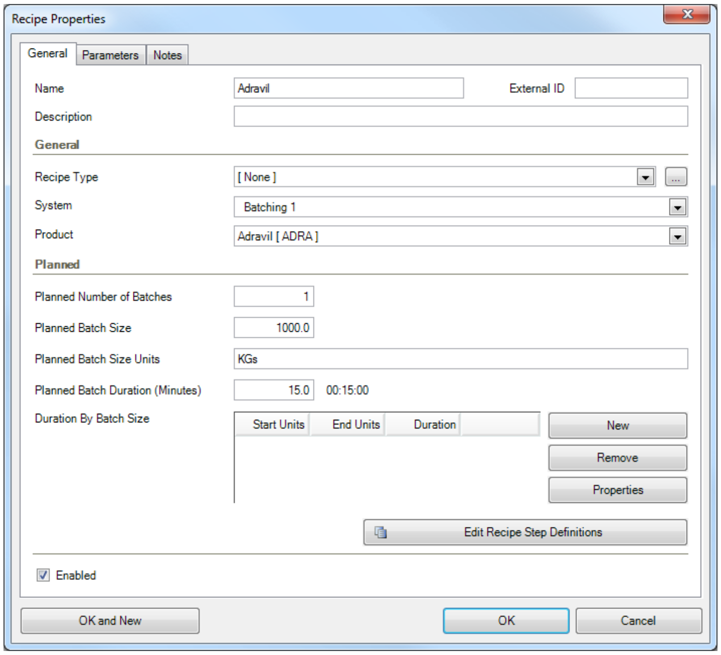
☑ Enabled

Create a Product Recipe

1. In the **PRODUCTTrak** panel of **MODELTrak**, under the **Packaging Product Set**, right-click the **Recipes** folder and select **New Recipe** from the menu list.
2. In the **Recipe Properties**, enter Adravil for the Name.
3. Configure the following settings with the specified values.

Setting	Value
System	**Batching 1**
Product	**Adravil [ADRA]**
Planned Number of Batches	**1**
Planned Batch Size	**1000.0**
Planned Batch Size Units	**KGs**
Planned Batch Duration (Minutes)	**15.0**

4. Click the **Edit Recipe Step Definitions** button to add Steps to the Recipe.



Recipe Properties

Name: Adravil
External ID:
Description:

General

Recipe Type: [None]
System: Batching 1
Product: Adravil [ADRA]

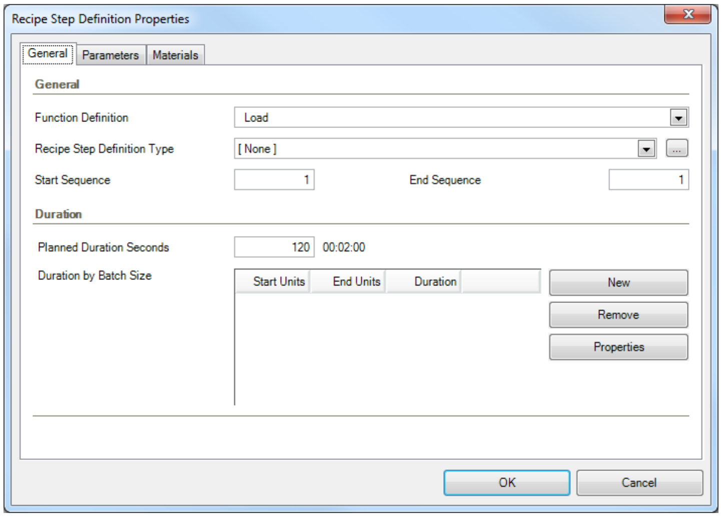
Planned

Planned Number of Batches: 1
Planned Batch Size: 1000.0
Planned Batch Size Units: KGs
Planned Batch Duration (Minutes): 15.0 00:15:00
Duration By Batch Size

Start Units	End Units	Duration

☑ Enabled

5. In the **Edit Recipe Step Definitions** page, click the **New** button to add a new Recipe Step.
6. In the **Recipe Step Definition Properties**, select **Load** for the **Function Definition**.
7. Enter **120** for the **Planned Duration Seconds**.
8. Enter 1 for the Start Sequence and the **End Sequence**.
9. Click **OK** to save this Recipe Step.



Recipe Step Definition Properties

General

Function Definition: Load
Recipe Step Definition Type: [None]
Start Sequence: 1
End Sequence: 1

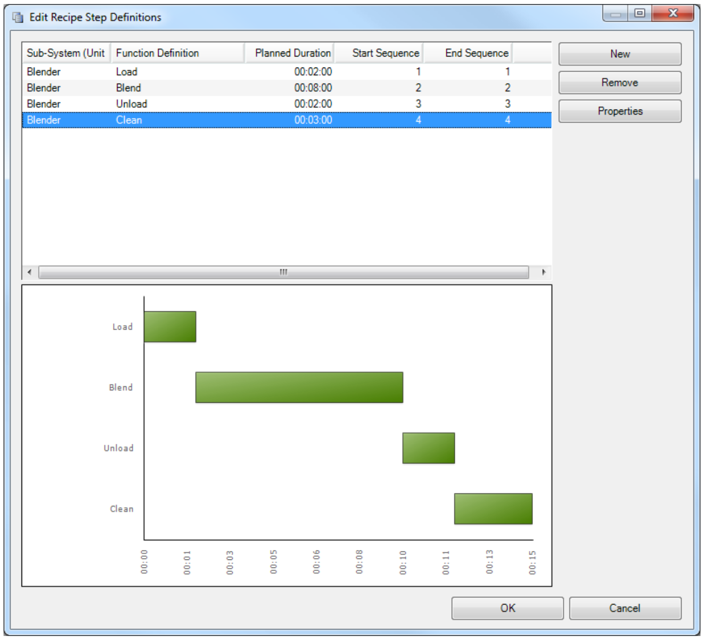
Duration

Planned Duration Seconds: 120 00:02:00
Duration by Batch Size

Start Units	End Units	Duration

10. Add 3 more Recipe Steps with the following settings.

Function Definition	Planned Duration Seconds	Start Sequence / End Sequence
Blend	**480**	**2**
Unload	**120**	**3**
Clean	**180**	**4**



Edit Recipe Step Definitions

Sub-System (Unit	Function Definition	Planned Duration	Start Sequence	End Sequence
Blender	Load	00:02:00	1	1
Blender	Blend	00:08:00	2	2
Blender	Unload	00:02:00	3	3
Blender	Clean	00:03:00	4	4

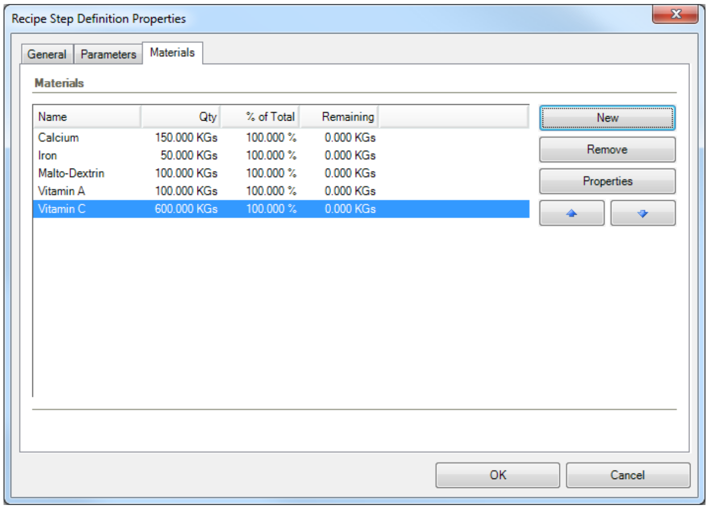
11. Click **OK** on the Recipe Properties to save the new Recipe.

Add Materials to a Recipe Step

1. Right-click on the **Adravil** Recipe and select **Properties**.
2. Click **Edit Recipe Step Definitions**.
3. Select the **Load** Recipe Step, the click the **Properties** button.
4. In the **Recipe Step Definition Properties**, click the **Materials** tab.
5. Click the New button to add a new Material to the Recipe Step.
6. In the **Recipe Step Definition Material Properties**, select **Calcium** for the **Material**, enter **150.0** for the **Quantity**, and click **OK**.
7. Repeat steps 5 and 6 to add additional Materials to the Load Recipe Step with the following properties.

Material	Quantity
Iron	**50.0**
Malto-Dextrin	**100.0**
Vitamin A	**100.0**
Vitamin C	**600.0**

8. Click OK to save changes to this Recipe Step.



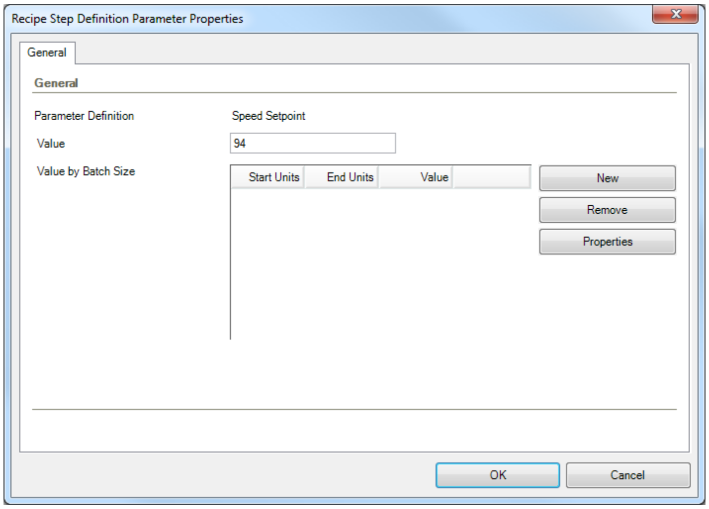
Recipe Step Definition Properties

Materials

Name	Qty	% of Total	Remaining
Calcium	150.000 KGs	100.000 %	0.000 KGs
Iron	50.000 KGs	100.000 %	0.000 KGs
Malto-Dextrin	100.000 KGs	100.000 %	0.000 KGs
Vitamin A	100.000 KGs	100.000 %	0.000 KGs
Vitamin C	600.000 KGs	100.000 %	0.000 KGs

Add Parameters to a Recipe Step

1. In the **Edit Recipe Step Definitions** page, select the **Blend** Recipe Step and click the **Properties** button.
2. In the **Recipe Step Definition Properties**, click the **Parameters** tab.
3. Select the Parameter **Speed Setpoint** and click the **Properties** button.
4. In the **Recipe Step Definition Parameter Properties**, enter **94.0** for the **Value** and click **OK**.



Recipe Step Definition Parameter Properties

General

Parameter Definition: Speed Setpoint

Value: 94

Value by Batch Size

Start Units	End Units	Value

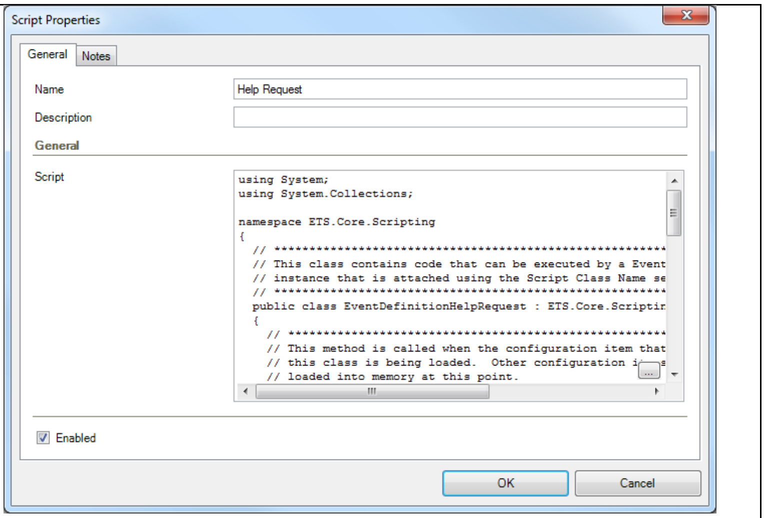
5. Click **OK** to save changes to the Recipe Step Definition, **OK** again to save changes to all Recipe Steps, and **OK** a third time to save changes to the Recipe.

Lab 08: MODELTrak – Entity Script Classes

Objectives	• Create an Entity Script Class • Add Custom Functionality to an Event Definition
Estimated Time to Complete This lab	30 minutes

In this lab you will utilize C#.NET to create Entity Script Classes in MODELTrak. Script Classes are used to add custom functionality to TrakSYS that can be triggered by standard configuration entities, such as Systems, Event Definitions, and Tags. You will then assign the Script Classes to specific Event Definitions that will run the custom functionality whenever they are started by their Trigger Tags.

Tasks	Detailed Steps
Open TrakSYS MODELTrak	1. Go to **Start | All Programs | Parsec | TrakSYS | TrakSYS MODELTrak**.
Create an Entity Script Class	1. Go to the **Administration** panel of MODELTrak. 2. Under the **Script Library** group, right-click on the **Entity Script Classes** group and select **New Script | Event Definition Script Class**. 3. In the **Script Properties**, enter **Help** Request for the **Name**. 4. Click the ... button to open the **Script Editor** dialog. 5. In the **Script Editor** dialog, change the name of the class from **EventDefinitionScript** to **EventDefinitionHelpRequest**. 6. Edit the method **PostScanEventStart(context)** to perform the following custom functionality whenever an Event starts. a. Insert a new record into the table **_tHelpRequest**. b. The values for the table field **RequestDateTime** should be the current date and time. c. The values for the table field **RequestMessage** should contain the ID for the current Event. The ID can be retrieved using the object property **context.EventID**. d. Use the TrakSYS Script API function **context.Execute(sql)** to insert data into the table. 7. Click the **Test Compile** button to check the script for syntax errors. Click **OK** to save this script. 8. Click **OK** again to save the new Entity Script Class.



Name: Help Request

Description:

General

Script:

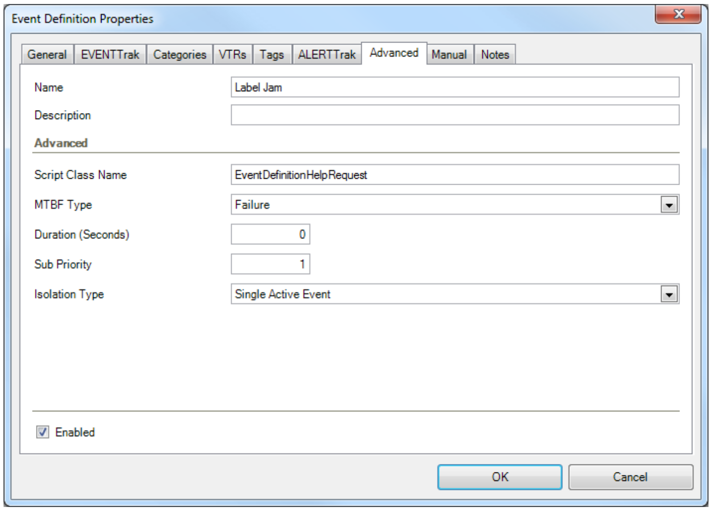
```
using System;
using System.Collections;

namespace ETS.Core.Scripting
{
  // *************************************************************
  // This class contains code that can be executed by a Event
  // instance that is attached using the Script Class Name se
  // *************************************************************
  public class EventDefinitionHelpRequest : ETS.Core.Scriptin
  {
    // ***********************************************************
    // This method is called when the configuration item that
    // this class is being loaded.  Other configuration i
    // loaded into memory at this point.
```

☑ Enabled

9. A possible solution can be found in the **Help Request [Complete]** Entity Script Class.

Assign the Entity Script to an Event Definition

1. Go to the **Areas and Systems** panel of MODELTrak.
2. Under the **Packaging** Area, expand the **Line 1** System and click the **Labeler** Sub-System. In the List View for the Labeler, right-click on the **Label Jam** Event Definition and select **Properties** from the menu.
3. In the **Event Definition Properties**, click the **Advanced** tab.
4. On the Advanced tab, enter **EventDefinitionHelpRequest** for the **Script Class Name**.
5. Click OK to save this Event Definition.



Name: Label Jam

Description:

Advanced

Script Class Name: EventDefinitionHelpRequest

MTBF Type: Failure

Duration (Seconds): 0

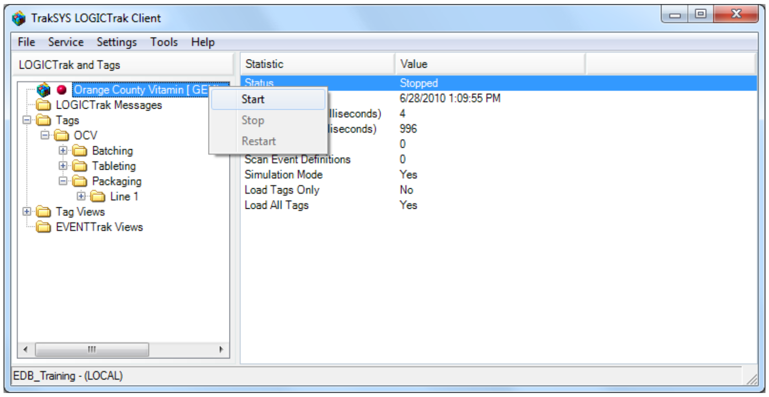
Sub Priority: 1

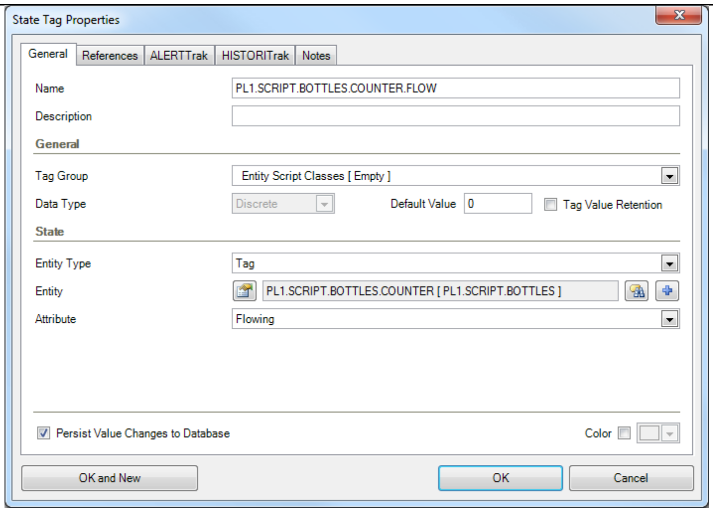
Isolation Type: Single Active Event

☑ Enabled

Run LOGICTrak Service

1. Go to **Start | All Programs | Parsec | TrakSYS | TrakSYS LOGICTrak Client**.
2. Run the LOGICTrak service by right-clicking on the LOGICTrak icon in the tree view and selecting **Start**. Notice the red icon

	becomes green once the service has started.  TrakSYS LOGICTrak Client File Service Settings Tools Help LOGICTrak and Tags Orange County Vitamin [GE LOGICTrak Messages Tags OCV Batching Tableting Packaging Line 1 Tag Views EVENTTrak Views Start Stop Restart EDB_Training - (LOCAL) Statistic — Value Status — Stopped 6/28/2010 1:09:55 PM lliseconds) — 4 liseconds) — 996 0 Scan Event Definitions — 0 Simulation Mode — Yes Load Tags Only — No Load All Tags — Yes
Change Tag Values in LOGICTrak	1. In the LOGICTrak tree view under the **Line 1** Tag Group, click the **Labeler** group. 2. Double-click on the **PL1.LABELER.LABEL.JAM** Tag to open the Update Tag Value dialog Box. 3. Change the value of the Tag from **0** to **1** to trigger a **Label Jam** Event, click the **Save** button and wait for a few seconds. 4. Change the Tag value back to 0 to make the Event inactive.
Run SQL Server Management Studio	1. Go to **Start | All Programs | Microsoft SQL Server 2005 | SQL Server Management Studio**. 2. Connect to the local server using **Windows Authentication**. 3. Check the table **_tHelpRequest** to confirm that a new record was written to the table when the **Label Jam** Event was triggered.
Create a State Tag [*Optional*]	1. Go to the **Tags** panel of MODELTrak. 2. Navigate to the Tag Group **TrakSYS Training Assignments\Entity Script Classes [Empty]**. 3. Right-click on the Entity Script Classes [Empty] group and select **New Tag | New State Tag** from the menu. 4. In the **State Tag Properties**, enter **PL1.SCRIPT.BOTTLES.COUNTER.FLOW** for the **Name**. 5. Select **Tag** for the **Entity Type**. 6. Find and assign the Tag **PL1.SCRIPT.BOTTLES.COUNTER** for the **Entity**. 7. Select **Flowing** for the **Attribute**. 8. Click **OK** to save this Tag.

	 State Tag Properties General References ALERTTrak HISTORITrak Notes Name: PL1.SCRIPT.BOTTLES.COUNTER.FLOW Description: General Tag Group: Entity Script Classes [Empty] Data Type: Discrete Default Value: 0 □ Tag Value Retention State Entity Type: Tag Entity: PL1.SCRIPT.BOTTLES.COUNTER [PL1.SCRIPT.BOTTLES] Attribute: Flowing ☑ Persist Value Changes to Database Color □ OK and New OK Cancel
Create an Entity Script Class [*Optional*]	1. Go to the **Administration** panel of MODELTrak. 2. Add a new **System Script Class** to the **Entity Script Classes** group. 3. In the **Script Properties**, enter **Manual Events** for the **Name**. 4. Open the **Script Editor**, change the name of the class from **SystemScript** to **SystemManualEvents**, and modify the method **PostScan(context)** to perform the following functionality. a. Reset the value of all Virtual Trigger Tags for the System's Event Defintions to zero (0) when the Tag PL1.SCRIPT.BOTTLES.COUNTER.FLOW is turns true (changes from zero (0) to one (1)). 5. Save the new System Script Class and assign it to the Packaging **Line 1** System. 6. A possible solution can be found in the **Manual Events [Complete]** Entity Script Class.
Reset Manual Events [*Optional*]	1. Go to **LOGICTrak Client** and restart the **LOGICTrak Service** for your changes to take effect. 2. Navigate to the Tag Group **TrakSYS Training Assignments\Entity Script Classes [Empty]** and set the Tag **PL1.SCRIPT.BOTTLES.COUNTER.CONTROL** to **0**. 3. Navigate to the Tag Group **OCV\Packaging\Line 1\[Event]** and change one or more Tags in this group to a value of 1. 4. Return to the **Entity Script Classes [Empty]** group and change the Tag **PL1.SCRIPT.BOTTLES.COUNTER.CONTROL** back to **1**. 5. Return to the **[Event]** group and confirm that all of the Event Tags have been reset to 0.

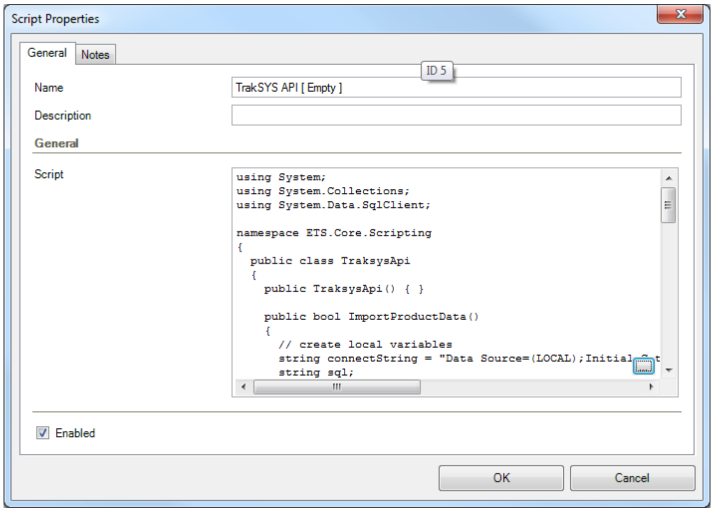
Lab 09: TrakSYS API

Objectives	• Use TrakSYS API classes and objects in a custom script.
Estimated Time to Complete This lab	20 minutes

In this lab you will use classes from the TrakSYS API to import Product data from an external database. A custom table in the TrakSYS database will be used as the data source for this lab. You will create this import functionality inside of a TrakSYS Script Class and trigger it manually from an Advanced Script Tag.

<table>
<tr><th>Tasks</th><th>Detailed Steps</th></tr>
<tr><td>Create a Script Class</td><td>

1. Go to the Administration panel of MODELTrak.
2. In the TrakSYS API Script Group, right-click the Script Class TrakSYS API [Empty] and select Properties from the menu.
3. Click the ... button to open the Script Editor.



Name: TrakSYS API [Empty]

Description:

General

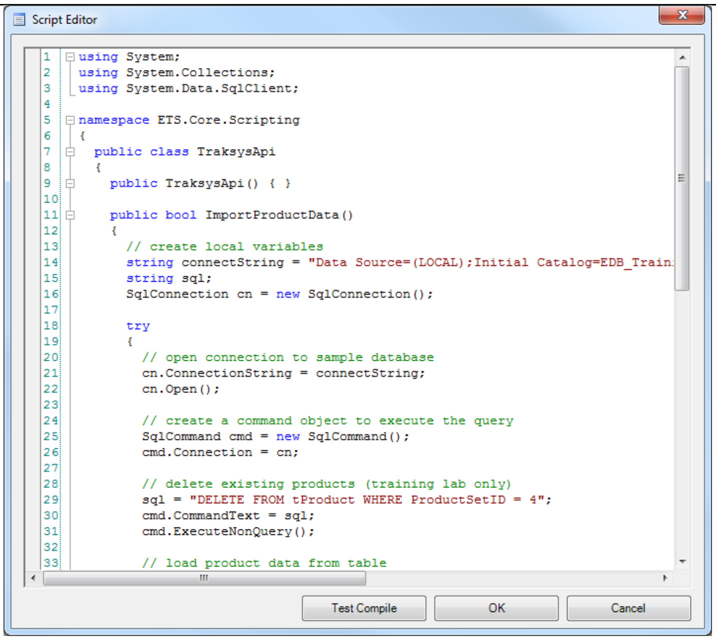
Script:

```
using System;
using System.Collections;
using System.Data.SqlClient;

namespace ETS.Core.Scripting
{
  public class TraksysApi
  {
    public TraksysApi() { }

    public bool ImportProductData()
    {
      // create local variables
      string connectString = "Data Source=(LOCAL);Initial Cat
      string sql;
```

☑ Enabled

4. Modify the existing script class to write product data from an existing query into the TrakSYS database using classes from the TrakSYS API.
    a. Loop through the SqlDataReader rs, which has been populated with the following Product data from an external database: **ProductCode**, **Name**, **Description**, **ProductSetID**.
    b. Create a **Data Model** object for Products (**ETS.Core.Api.Models.Data.DbProduct**) that can be populated with data from the SqlDataReader object.
    c. Use the existing **ApiService** object available from the **Global.Context** object (**Global.Context.Api**).
5. Click **OK** to save changes to this Script Class.

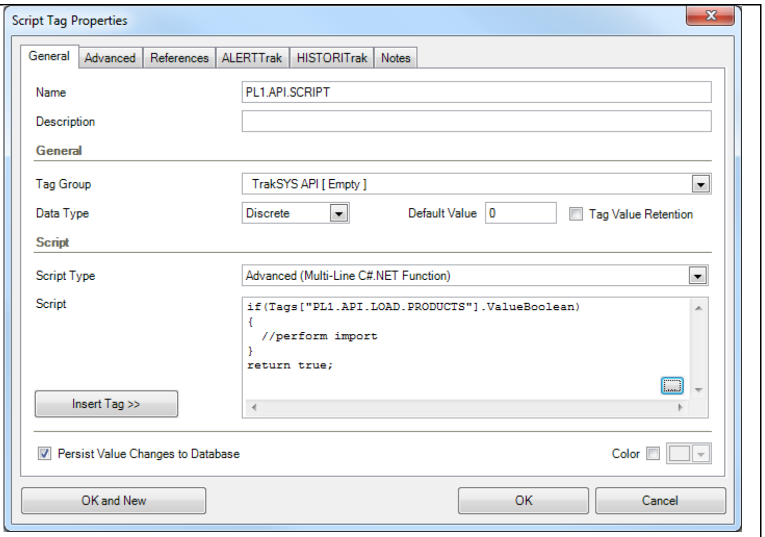
</td></tr>
</table>



```
using System;
using System.Collections;
using System.Data.SqlClient;

namespace ETS.Core.Scripting
{
  public class TraksysApi
  {
    public TraksysApi() { }

    public bool ImportProductData()
    {
      // create local variables
      string connectString = "Data Source=(LOCAL);Initial Catalog=EDB_Train
      string sql;
      SqlConnection cn = new SqlConnection();

      try
      {
        // open connection to sample database
        cn.ConnectionString = connectString;
        cn.Open();

        // create a command object to execute the query
        SqlCommand cmd = new SqlCommand();
        cmd.Connection = cn;

        // delete existing products (training lab only)
        sql = "DELETE FROM tProduct WHERE ProductSetID = 4";
        cmd.CommandText = sql;
        cmd.ExecuteNonQuery();

        // load product data from table
```

Create an Advanced Script Tag

1. Go to the **Tags** panel of MODELTrak.
2. Navigate to the Tag Group **TrakSYS Training Assignments\TrakSYS API [Empty]**, right-click on the Tag Group, and select **New Script Tag** from the menu.
3. In the Script Tag Properties, enter PL1.API.SCRIPT for the Name.
4. Select Advanced (Multi-Line C#.NET Function) for the Script Type.
5. Click the ... button to open the Script Editor.
6. Enter a script block that executes the method **TraksysApi.ImportProductData()** whenever the Tag **PL1.API.LOAD.PRODUCTS** is true.
   a. Create an instance of the **TraksysApi** class (this is not a static class).
   b. Use the function **Global.Context.QueueTagNameEqualsForAssignment (tagName, value)** to reset the Tag **PL1.API.LOAD.PRODUCTS** to false after the import is complete.
   c. Be sure that the script returns a true/false value as the value for the Script Tag.



Name: PL1.API.SCRIPT

Description:

General

Tag Group: TrakSYS API [Empty]

Data Type: Discrete

Default Value: 0

□ Tag Value Retention

Script

Script Type: Advanced (Multi-Line C#.NET Function)

Script:

```
if(Tags["PL1.API.LOAD.PRODUCTS"].ValueBoolean)
{
  //perform import
}
return true;
```

☑ Persist Value Changes to Database

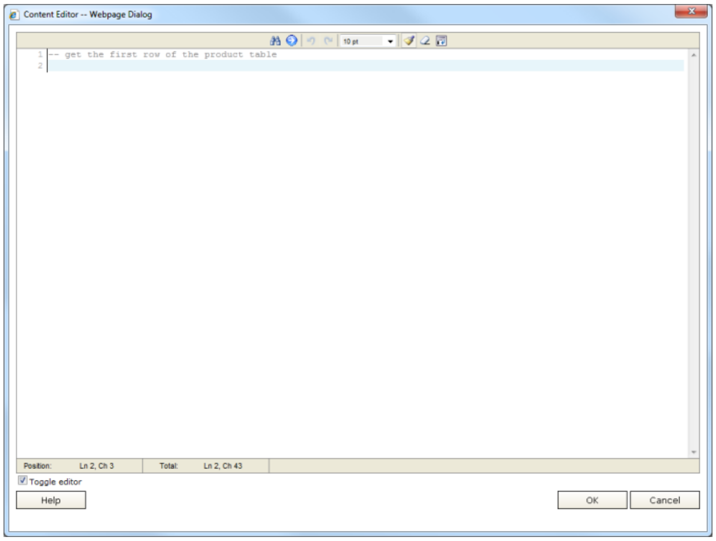
Color □

7. A possible solution can be found in the Script Tag **PL1.API.SCRIPT.COMPLETE**, under the Tag Group **TrakSYS API [Complete].**

Lab 10: WEBTrak – HTML Content (with Post Back)

Objectives	• Create a user form with the HTML Content (with Post Back) Web Part
Estimated Time to Complete This lab	30 minutes

In this lab you will create a custom user form in a WEBTrak Dashboard using the HTML Content (with Post Back) Web Part. This form will allow a user to edit data for an existing Product in a PRODUCTrak Product Set. You will create the form using HTML, populate it with default data from the TrakSYS database, and execute an update query when the form is submitted to save any changes.

Tasks	Detailed Steps
Edit a Dashboard	1. Open **WEBTrak** from **Start | All Programs | Parsec | TrakSYS | WEBTrak**. 2. Click the **Login** button on the menubar to login as the **Adminstrator** (**admin** / **sa**). Check the **Remember Me** box to cache the login. 3. Navigate to the Report Group **TrakSYS Reports\TrakSYS Advanced Training\HTML Content (with Post Back)**. 4. Open the **HTML Post Back [Empty]** dashboard report and click the **Edit Dashboard** button on the menubar.
Edit the HTML Content (with Post Back) Web Part	1. Click the button on the **Product Editor – [HTML Content (with Post Back)]** to edit the shared Web Part. 2. Under **Web Part Settings**, go to the **Source** panel. 3. For **SQL Query**, enter a query to retrieve the first row in the table **tProduct** from the TrakSYS database.  Content Editor -- Webpage Dialog `-- get the first row of the product table` Position: Ln 2, Ch 3 Total: Ln 2, Ch 43 ☑ Toggle editor 4. In the **Settings** panel, click the **Text** button to create an HTML form for the Web Part using the **HTML Editor** dialog. a. The form should have one (1) hidden input for storing the

ID of the Product.

b. The form should have two (2) text inputs for editing the following values from the Source **SQL Query**: **ProductCode**, **Name**.

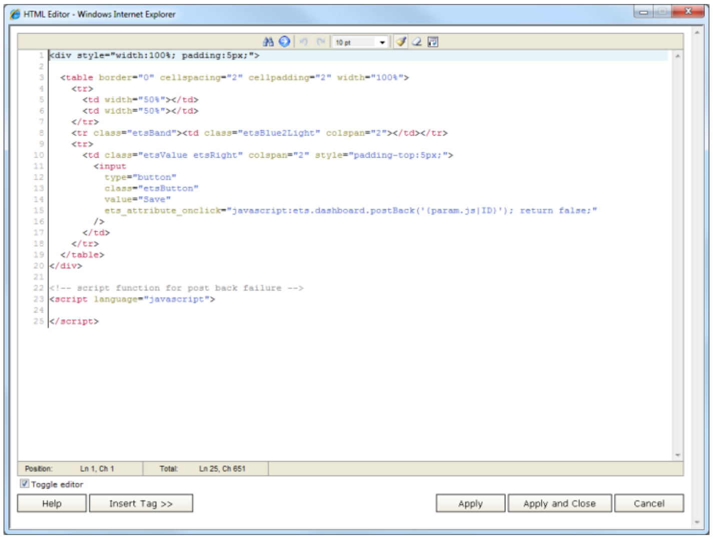
c. For the text inputs on the form, initialize the input values using the WEBTrak HTML attribute **ets_attribute_etsinitvalue**.

d. A **Save** button has already been added to the HTML form for posting the form data back to the server for processing.

e. Inside the existing **<script>** tag at the bottom of the form, add the following JavaScript function:

```
function postBackFailure(array) {
   alert('Update Failed!');
}
```

f. Click **Apply and Close** when you are finished editing the HTML content for the Web Part to close the HTML Editor dialog.

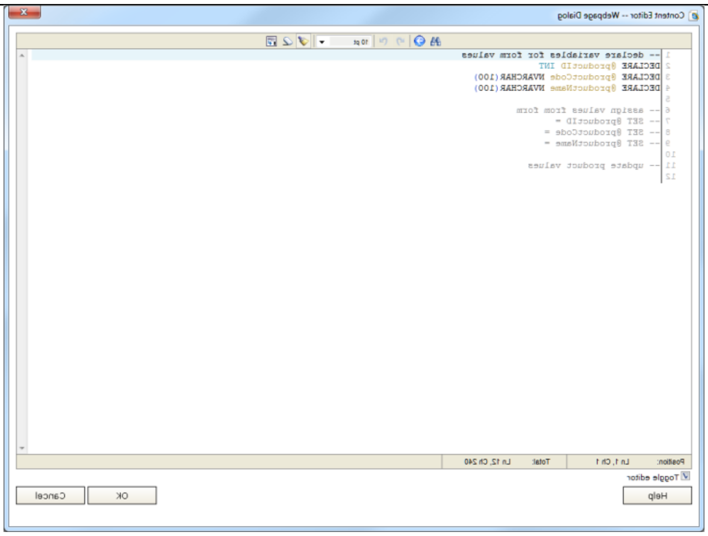


```
<div style="width:100%; padding:5px;">

  <table border="0" cellspacing="2" cellpadding="2" width="100%">
    <tr>
      <td width="50%"></td>
      <td width="50%"></td>
    </tr>
    <tr class="etsBand"><td class="etsBlue2Light" colspan="2"></td></tr>
    <tr>
      <td class="etsValue etsRight" colspan="2" style="padding-top:5px;">
        <input
          type="button"
          class="etsButton"
          value="Save"
          ets_attribute_onclick="javascript:ets.dashboard.postBack('{param.js|ID}'); return false;"
        />
      </td>
    </tr>
  </table>
</div>

<!-- script function for post back failure -->
<script language="javascript">

</script>
```

5. In the **Advanced** panel, for **Post Back SQL**, enter a query that updates the table _tProduct with new values from the HTML form for a specific product.

   a. Load data from the HTML form into SQL variables using **Content Expressions** (**{form|FieldName}**).

   b. Be sure to only update rows that match the selected ID.

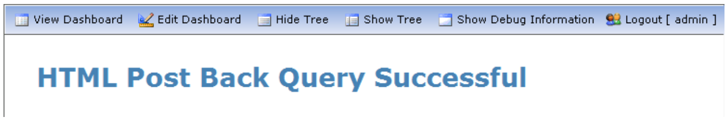


```
-- declare variables for form values
DECLARE @productID INT
DECLARE @productCode NVARCHAR(100)
DECLARE @productName NVARCHAR(100)

-- assign values from form
-- SET @productID =
-- SET @productCode =
-- SET @productName =

-- update product values
```

6. Enter the following value for **On Success Redirect URL**: http://localhost/webtrak/report/view.aspx?ReportKey=HTML.SUCCESS
7. Click OK to save changes to this Web Part.

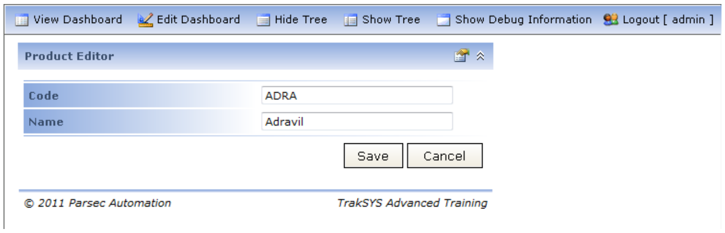
Edit a Product

1. Click **View Dashboard** on the menubar to view the completed form. Change the **ProductCode** or **Name** values for the selected product (**Adravil**) and click the **Save** button to update the TrakSYS database.
2. You will be redirected to the following web page if the update was successful.



HTML Post Back Query Successful

3. A possible solution can be found in the **HTML Post Back [Complete]** dashboard.



Product Editor

Code: ADRA

Name: Adravil

© 2011 Parsec Automation TrakSYS Advanced Training