FactoryPMI Reporting Introduction

Welcome to FactoryPMI Reporting!

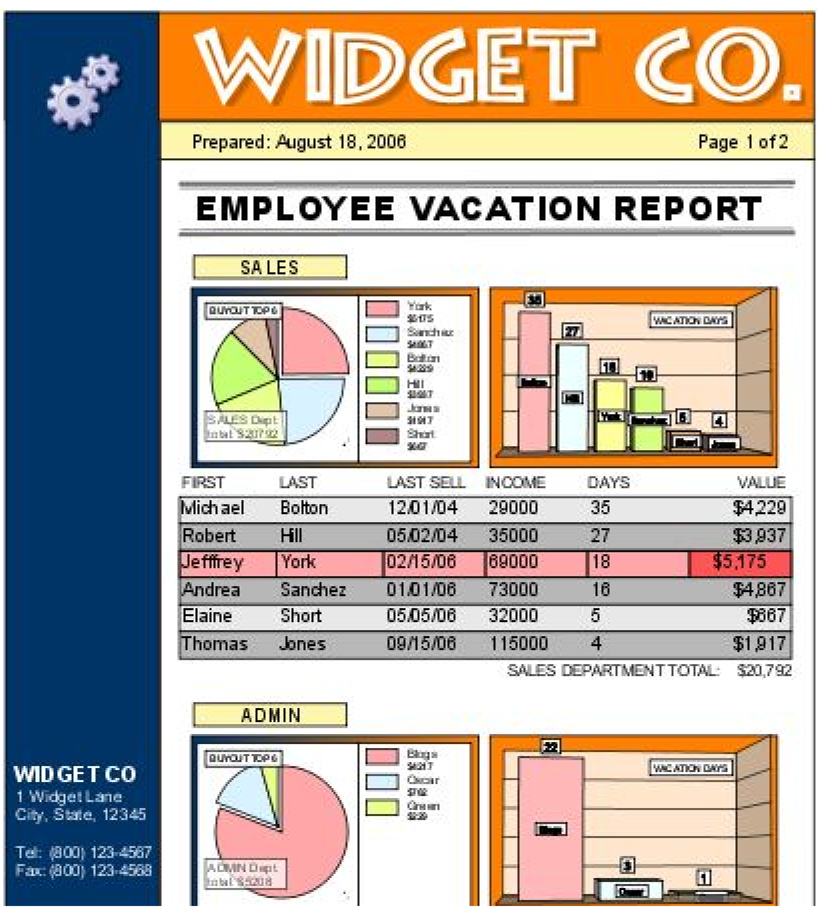
FactoryPMI Reporting is a plugin for creating dynamic reports! These reports may be generated from existing Adobe Acrobat (pdf) files or created totally from scratch. Data is introduced through FactoryPMI, providing access to any SQL database!

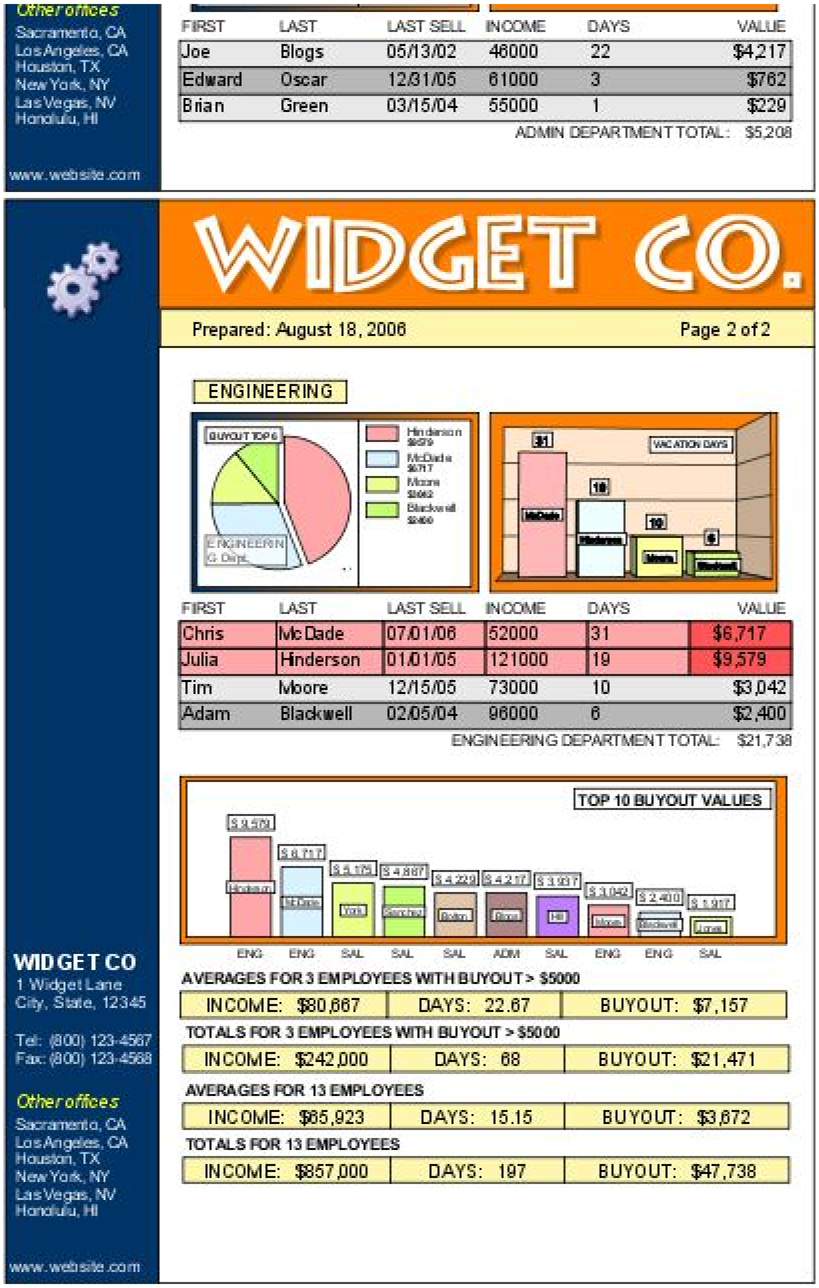
FactoryPMI Reporting makes creating professional reports easy with a rich library including: images, graphs, tables, and a variety of basic shape tools. Access to reports is web based via the FactoryPMI runtime, a Java application, providing authenticated users access from anywhere, all based on networking standards that your IT department can support. Reports are printer friendly and can easily be exported to a variety of formats including pdf! Here are some common uses of dynamic reports:

- Production Management
- Efficiency Monitoring
- Downtime Tracking
- Statistical Process Control
- Quality Assurance
- Overall Equipment Effectiveness (OEE) Management
- Historical Data analysis

Benefits

FactoryPMI Reporting enables managers to increase productivity, decrease waste, reduce costs, and increase quality with existing resources by providing an view of their manufacturing process. Managers often save time by automating reporting processes that were once done by hand. Often valuable man hours that went into creating spreadsheets or reports can be recovered! These reports are trivial to manage since they are generated on the fly from existing SQL database data.




Other offices
Sacramento, CA
Los Angeles, CA
Houston, TX
New York, NY
Las Vegas, NV
Honolulu, HI

www.website.com

FIRST	LAST	LAST SELL	INCOME	DAYS	VALUE
Joe	Blogs	05/13/02	46000	22	$4,217
Edward	Oscar	12/31/05	61000	3	$762
Brian	Green	03/15/04	55000	1	$229

ADMIN DEPARTMENT TOTAL: $5,208

WIDGET CO.

Prepared: August 18, 2006 Page 2 of 2

ENGINEERING

FIRST	LAST	LAST SELL	INCOME	DAYS	VALUE
Chris	McDade	07/01/06	52000	31	$6,717
Julia	Hinderson	01/01/05	121000	19	$9,579
Tim	Moore	12/15/05	73000	10	$3,042
Adam	Blackwell	02/05/04	96000	6	$2,400

ENGINEERING DEPARTMENT TOTAL: $21,738

WIDGET CO
1 Widget Lane
City, State, 12345

Tel: (800) 123-4567
Fax: (800) 123-4568

Other offices
Sacramento, CA
Los Angeles, CA
Houston, TX
New York, NY
Las Vegas, NV
Honolulu, HI

www.website.com

AVERAGES FOR 3 EMPLOYEES WITH BUYOUT > $5000

INCOME: $80,667	DAYS: 22.67	BUYOUT: $7,157

TOTALS FOR 3 EMPLOYEES WITH BUYOUT > $5000

INCOME: $242,000	DAYS: 68	BUYOUT: $21,471

AVERAGES FOR 13 EMPLOYEES

INCOME: $65,923	DAYS: 15.15	BUYOUT: $3,672

TOTALS FOR 13 EMPLOYEES

INCOME: $857,000	DAYS: 197	BUYOUT: $47,738

Report created in Tutorial #2.

Help File Organization

This help file is organized in 4 main sections:

Introduction
Provides the basic information needed to get started, including how to install and activate. Goes over FactoryPMI reporting features and how it works with the FactoryPMI system.

Tutorials
Introduces you to the Report Designer by stepping you through creating simple, yet powerful example reports. This is a great place to consider what can be created with FactoryPMI Reporting.

Components
Broken down to two sections. *FactoryPMI Components* interface directly with your FactoryPMI project. *ReportViewer Components* are strictly used within the Report Designer. They are the tools that allow the creation of professional reports.

Concepts
User Interface concepts go over the way that report customization works within the Report Designer. *Basic* and *Advanced* concepts discuss various aspects of FactoryPMI Reporting. This is a good section to begin reading early to learn what Reporting is all about. Re-read this section after you have become proficient with the interface to gain a full understanding of how Report Generation works.

Extended Help

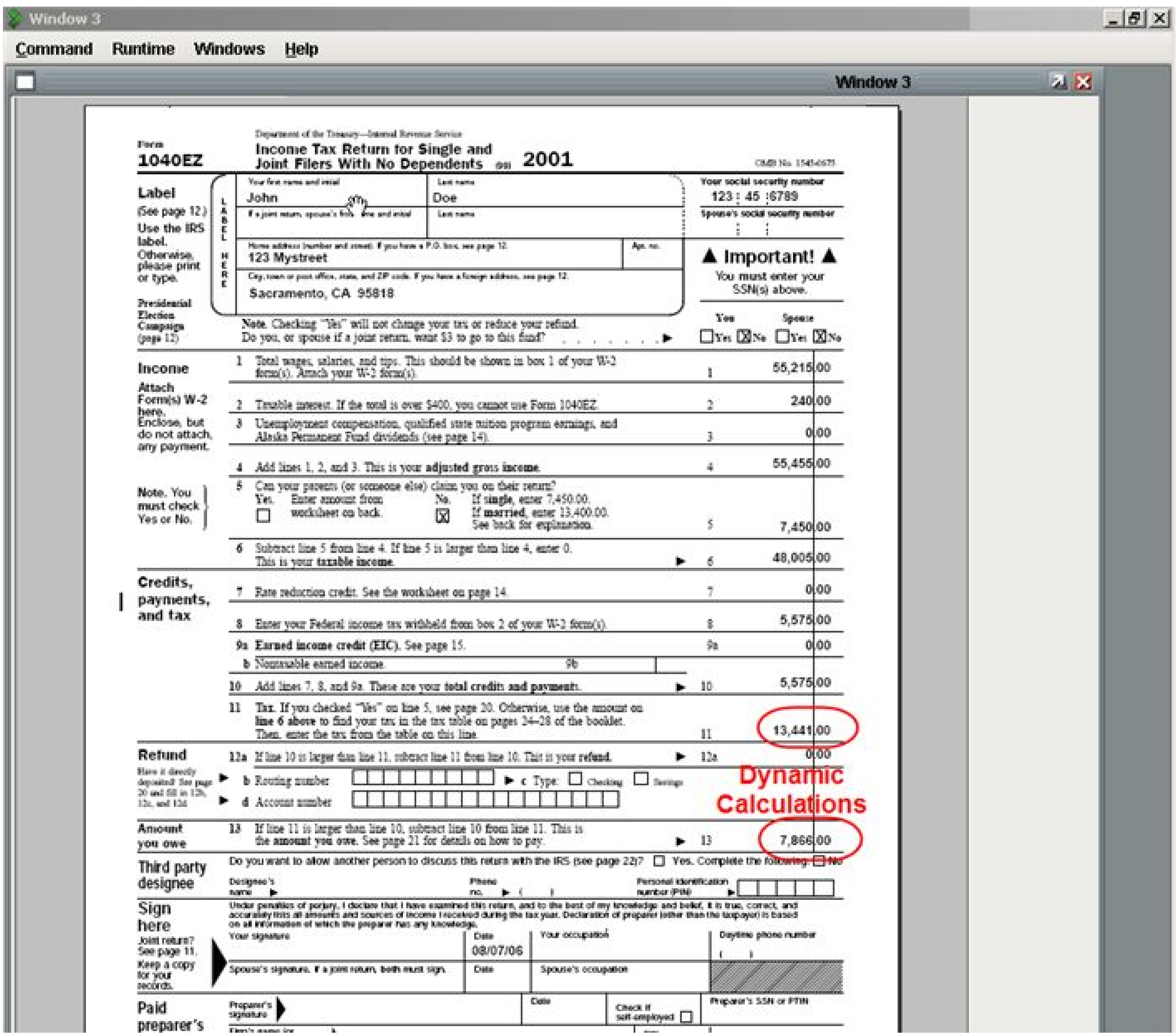
As no manual can fully cover every conceivable situation or topic, it's important that you know where to go for answers. The first and best place is the Inductive Automation Web site and the Inductive Automation Forum, where you can peruse the issues and questions that other users have encountered. We will respond to your posts by the next business day.
From there, you may E-mail Us. We strive to provide a quick turn around on answers - usually within 24 hours. Finally, registered users may call us toll-free at 1-800-266-7798.

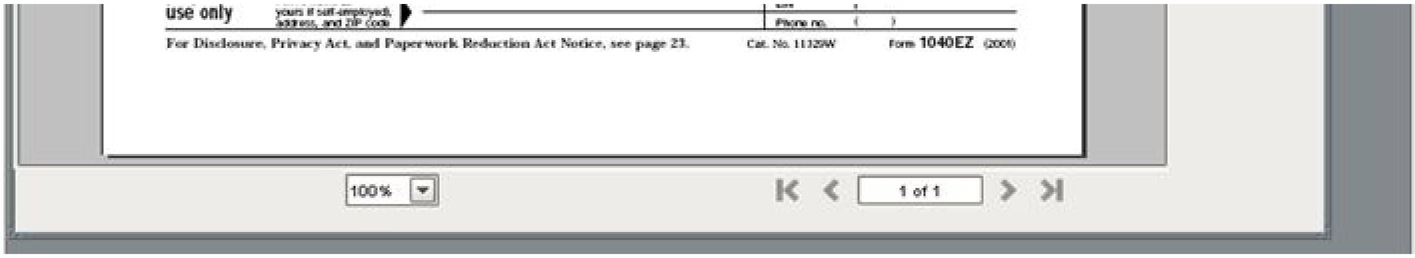
FactoryPMI Reporting Features

The most noteworthy feature of *FactoryPMI Reporting* is the fact that is integrated into the FactoryPMI system. This: provides access to FactoryPMI data including any SQL database, allows an unlimited number of concurrent clients via web based access, and shares authentication with your existing FactoryPMI project.

Other features:

- Easy to use WYSIWYG (What you see is what you get) designer that includes an intuitive layout and drawing tools
- Powerful table tool that creates new pages to fit your data. It supports a wide range of features.
- Ability to start with an existing pdf report for automatic form fill-in.
- Reports are printer friendly.
- Every report can be saved by the user in a variety of formats including pdf.
- The Reporting Plugin includes the Row Selector and Column Selector components. Both are very useful when working with DataSets. They work especially well with FactoryPMI graph and table components as well as the Report Viewer.
- The Reporting Plugin includes the File Explorer and PDF File Viewer components. These are very useful for viewing machine maintenance manuals or any other PDFs from within your project.

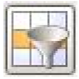
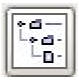


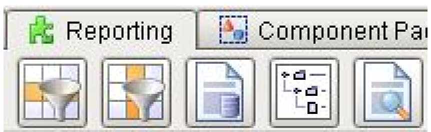


Example report based on existing pdf

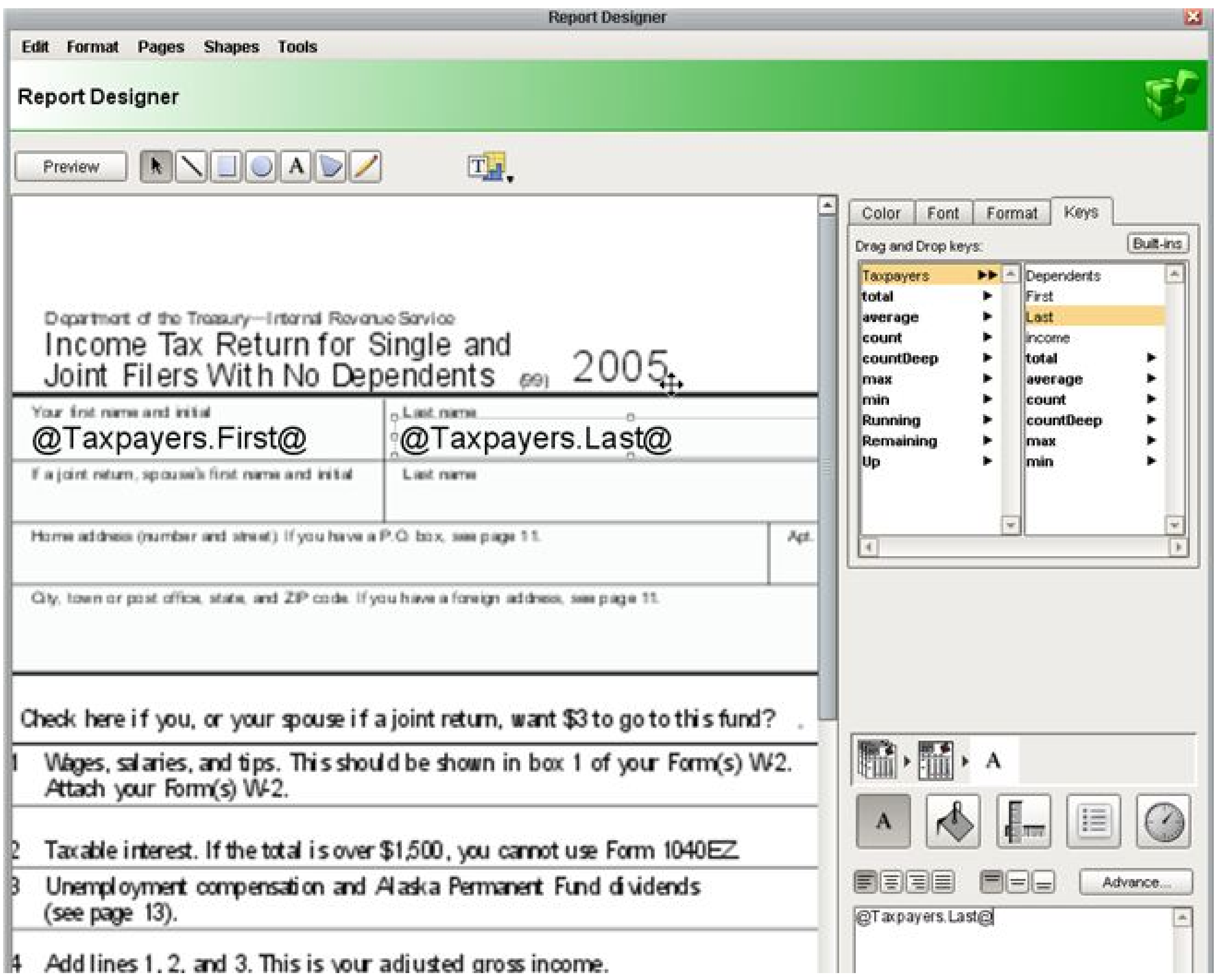
FactoryPMI Reporting Model

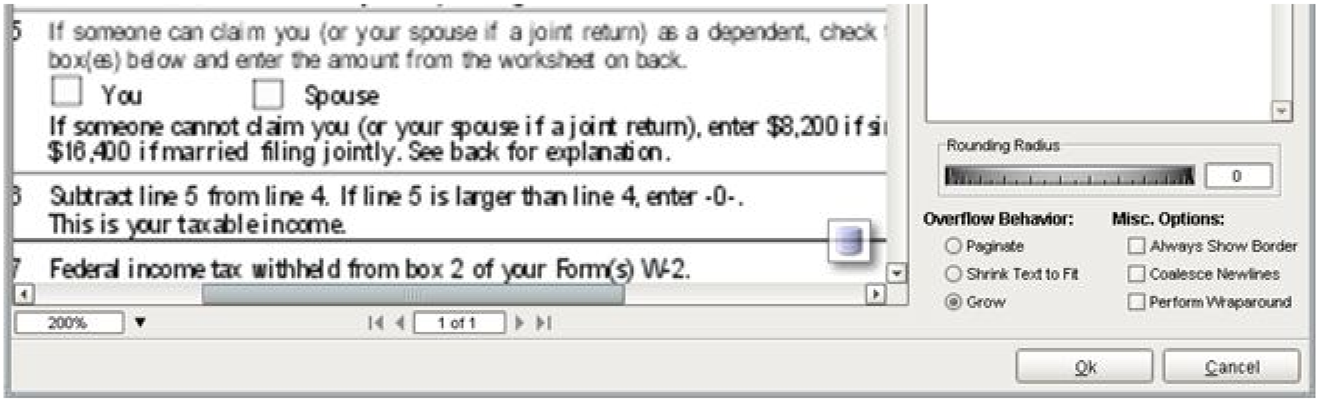
When you install the *FactoryPMI Reporting* plugin a **Reporting** tab appears in the designer that contains the following:

-  Row Selector
-  Column Selector
-  Report Viewer
-  File Explorer
-  PDF Viewer

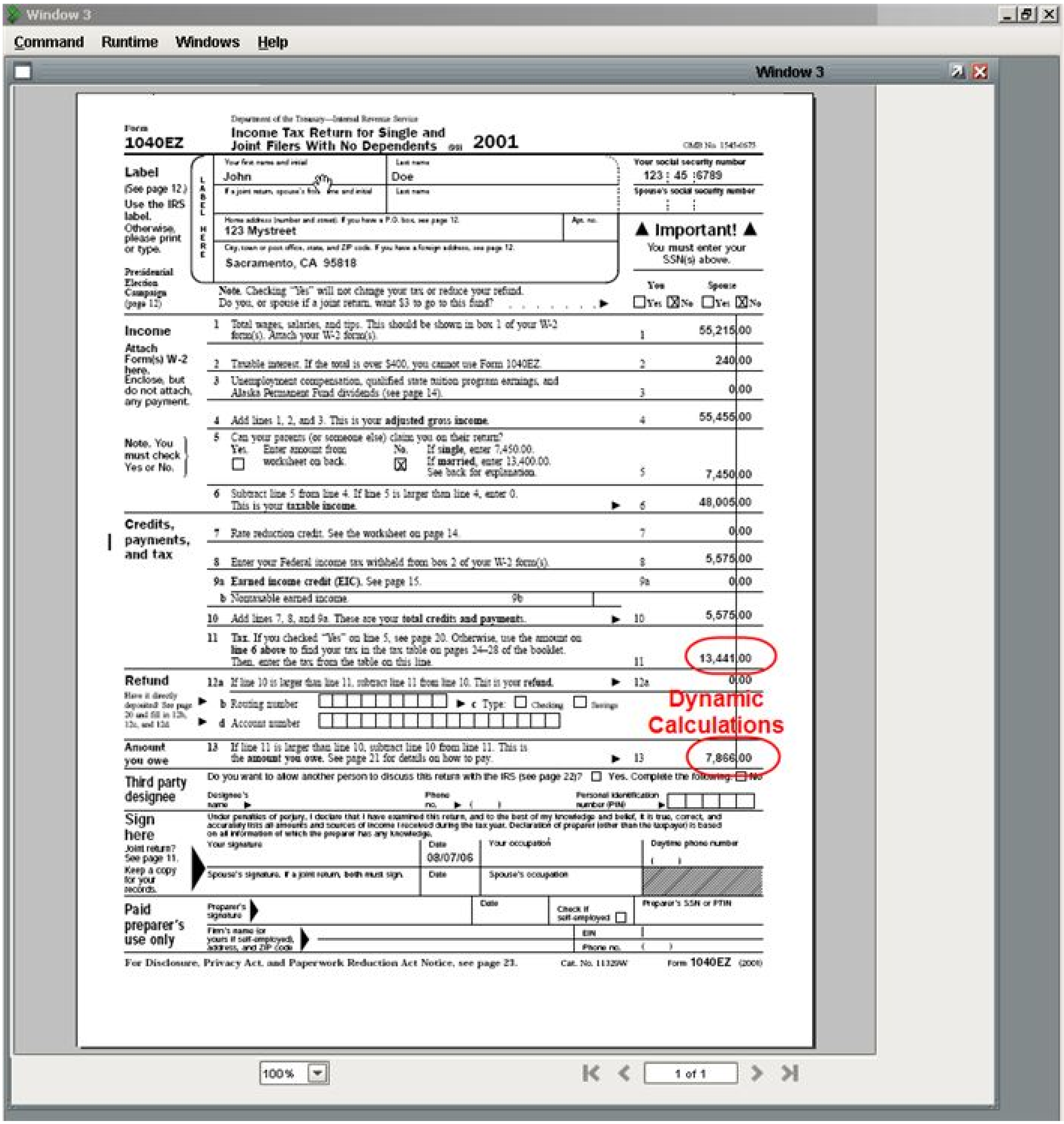


Simply use these objects as you would any FactoryPMI components. The bulk of creating your professional report is done through the Report Designer, which is the customizer (Cntl+U) for the Report Viewer.





Example pdf report in the [Report Designer]



Viewing report in the FactoryPMI Runtime

How to Install The Gateway

Installation

Installing the FactoryPMI Reporting plugin is a simple process done in the FactoryPMI web configuration. From the Gateway Configuration Page do the following

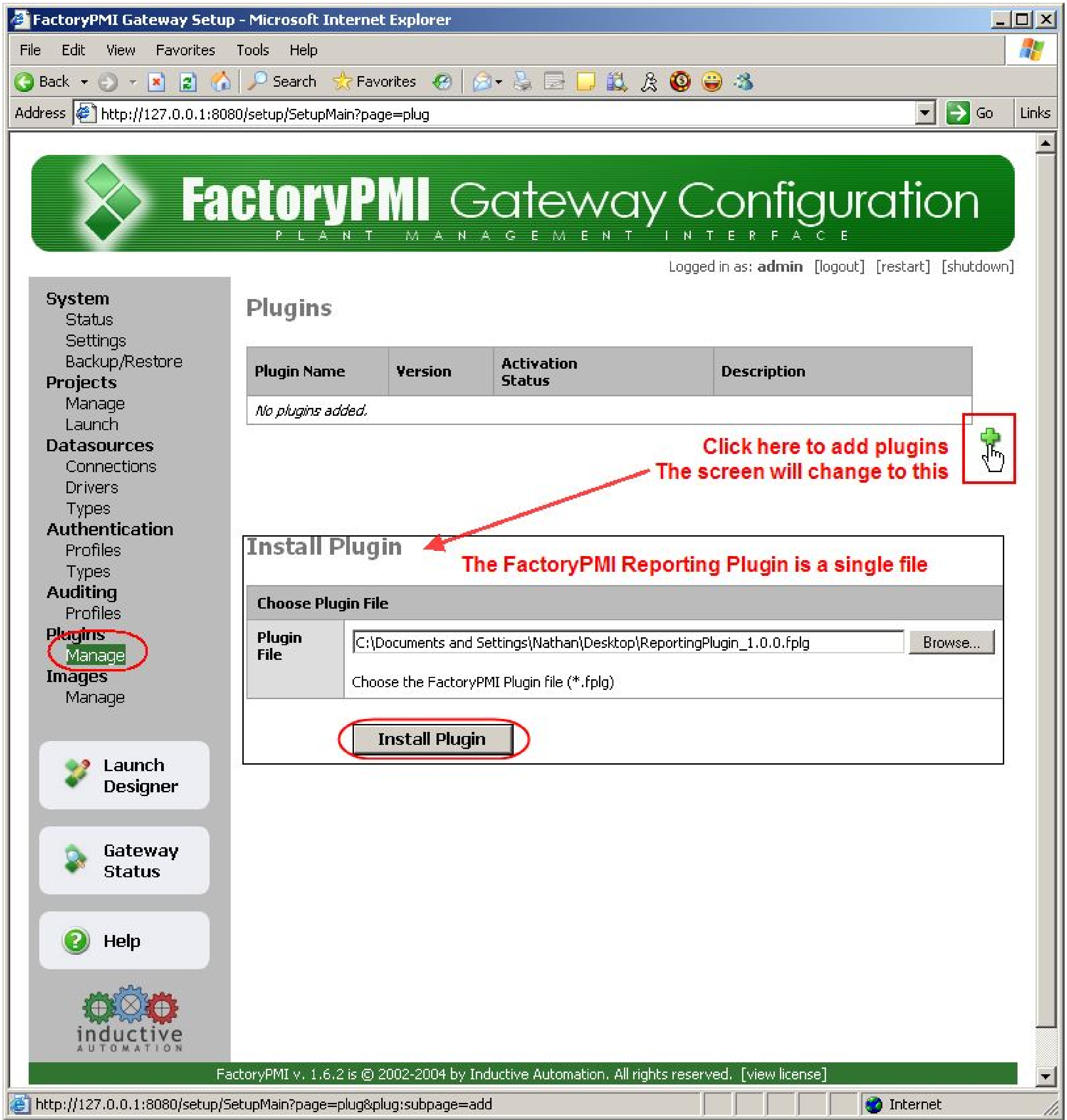


Fig. 1: Go to the Plugins section, then click the "add plugin" icon

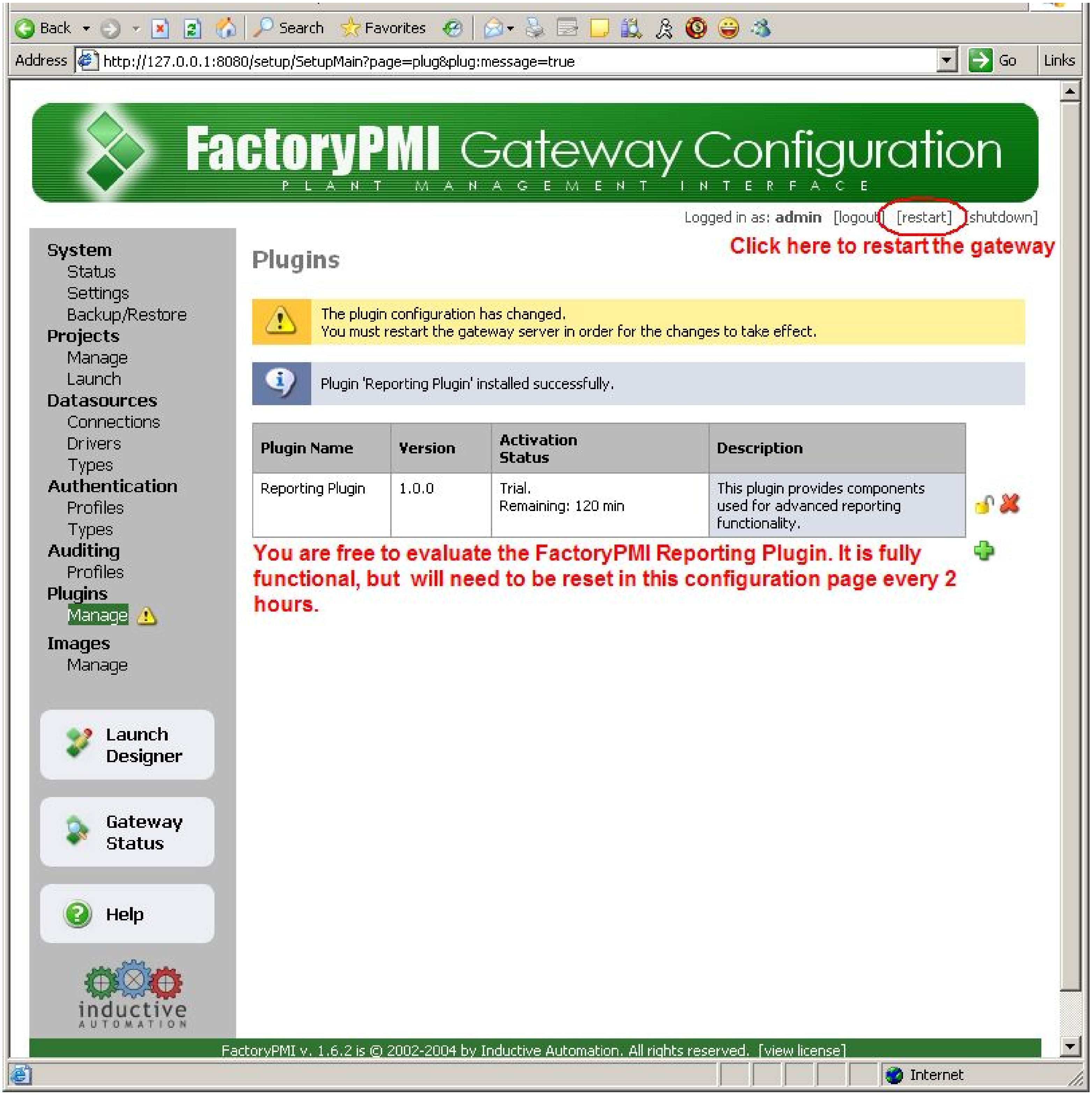


Fig. 2: Select the Reporting Plugin .fplg file. The gateway will need to be restarted

Trial Mode

The FactoryPMI Reporting plugin trial works in a similar fashion as FactoryPMI's trial mode.

The trial mode provides a way for you to try our software without any feature restrictions. This allows you to fully evaluate our software to make sure that it's right for you. In the trial mode, all reports will have a watermark on them displaying the fact that the reporting plugin is being run in trial mode. In addition, after two hours of cumulative runtime, the plugin will 'timeout'. When the plugin times out, the Row Selector and Column selector components will have a watermark on them, and the report component will no longer be able to print or save to PDF. You can log into the FactoryPMI Gateway and reset the plugin, which resets the two-hour timeout period. You can do this as many times as you want, which means that you can evaluate it for as long as you want! This system gives you flexibility to evaluate our product, while making it impractical for industrial use.

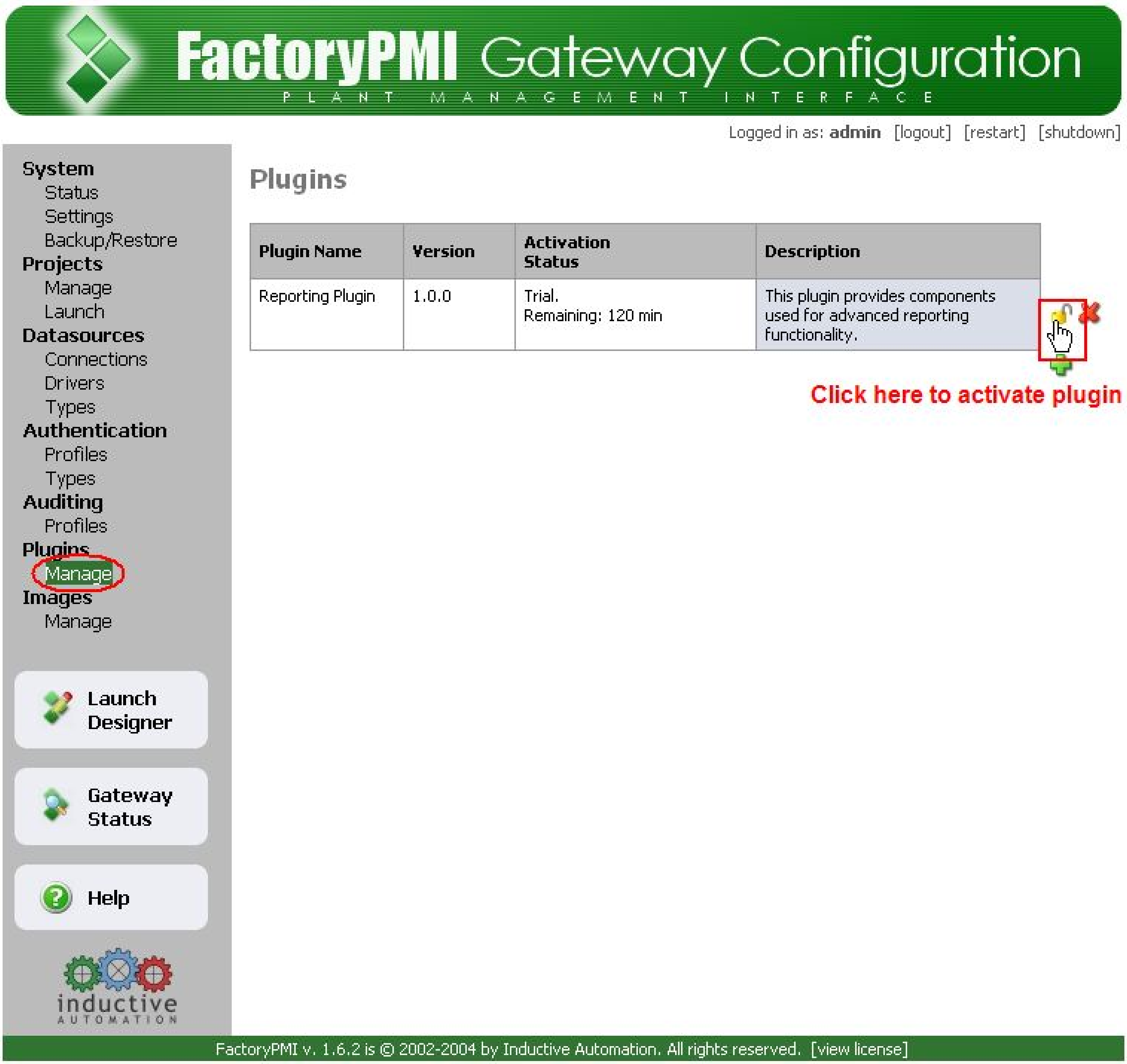
Running FactoryPMI Designer does not cause your trial window to decrease. This means that you can design an entire project on an un-activated FactoryPMI Gateway.

Registration and Activation

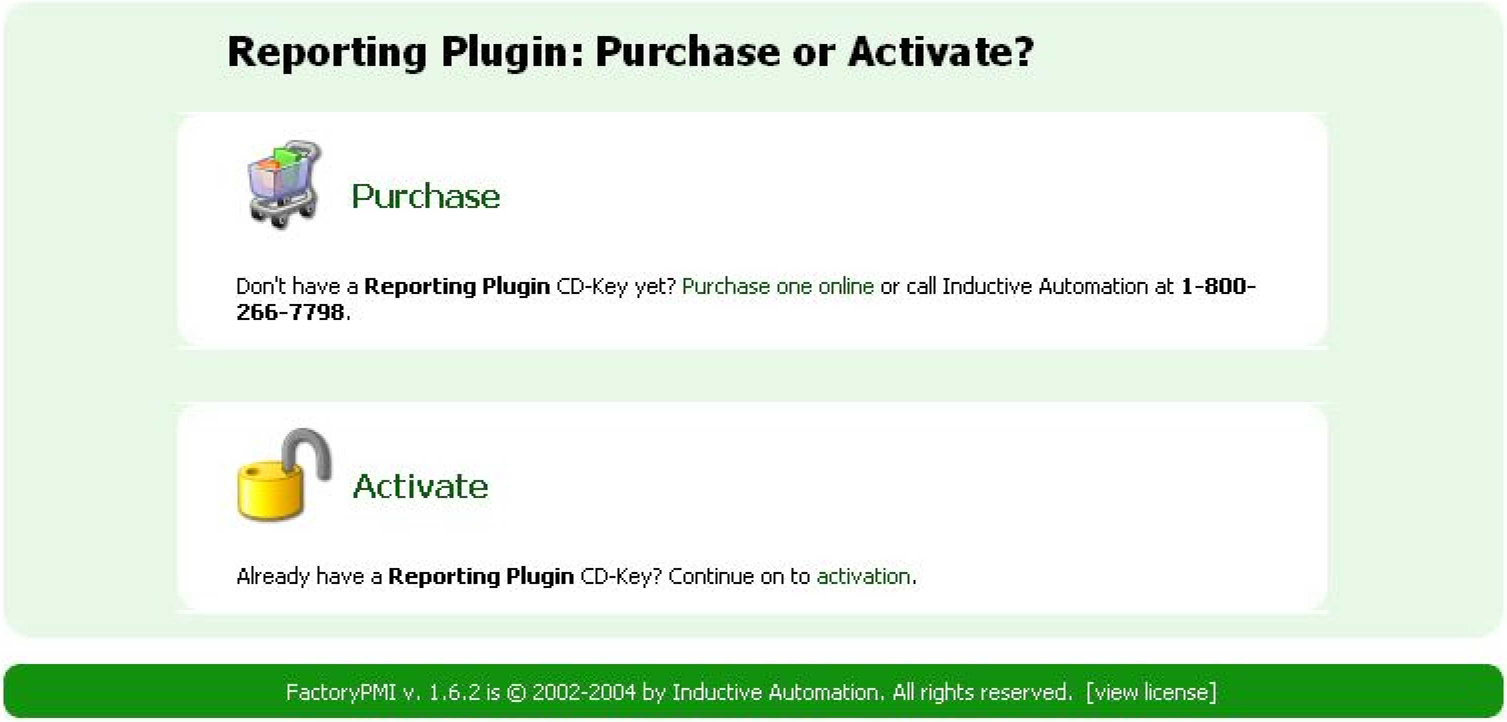
Registration and Activation

Registration and Activation is the process by which you activate your purchased copy of the FactoryPMI Reporting plugin in order to remove the time limitation. **Registration** refers to the act of acquiring a CD-Key, and entering it in the program. **Activation** refers to notifying Inductive Automation that you are using that cd key for an installation, and in most cases can be done instantly over the Internet.

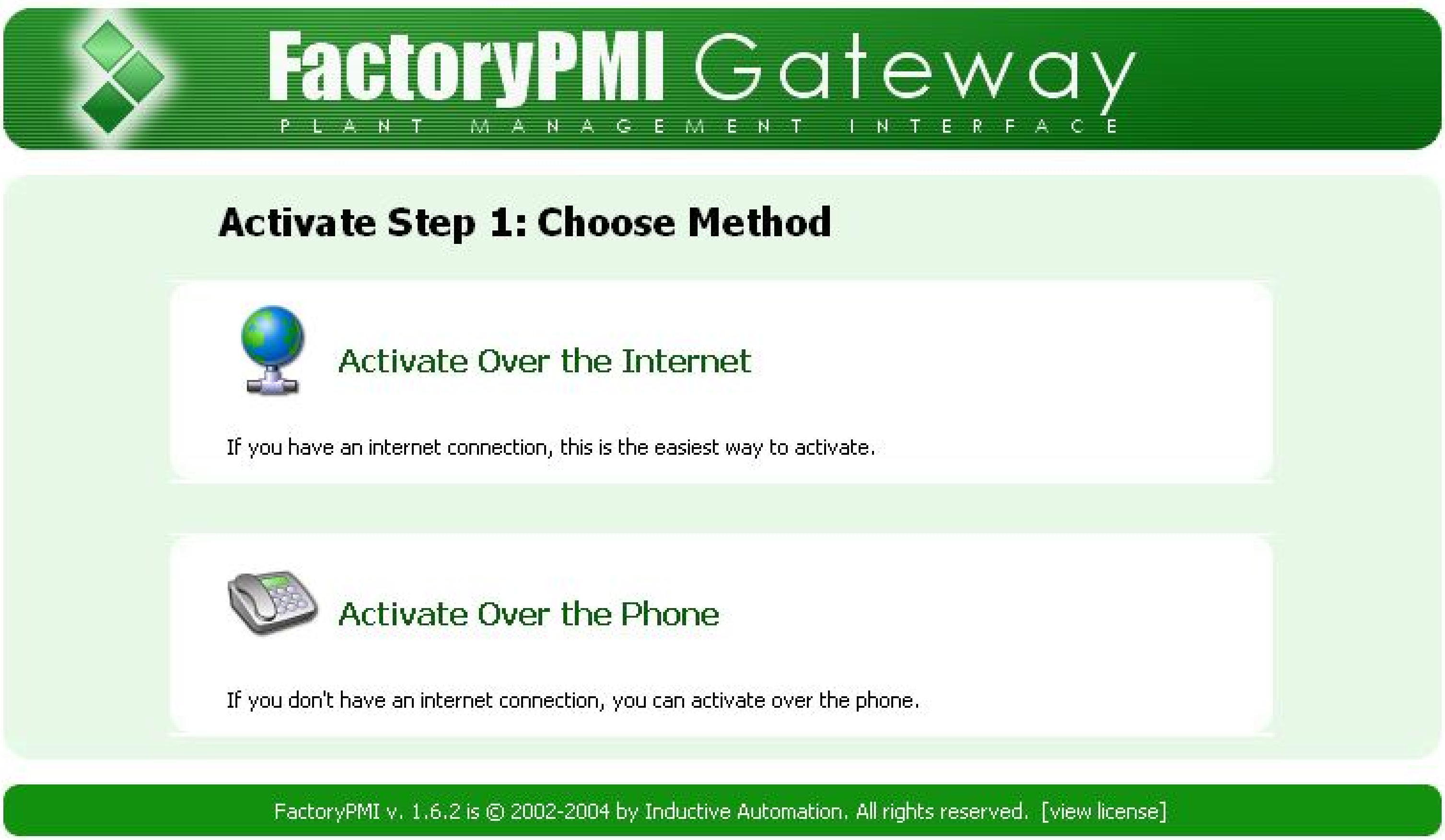
To begin the process: log on to the FactoryPMI Gateway Configuration web page, Click manage plugins, then click the **activate plugin** icon.:



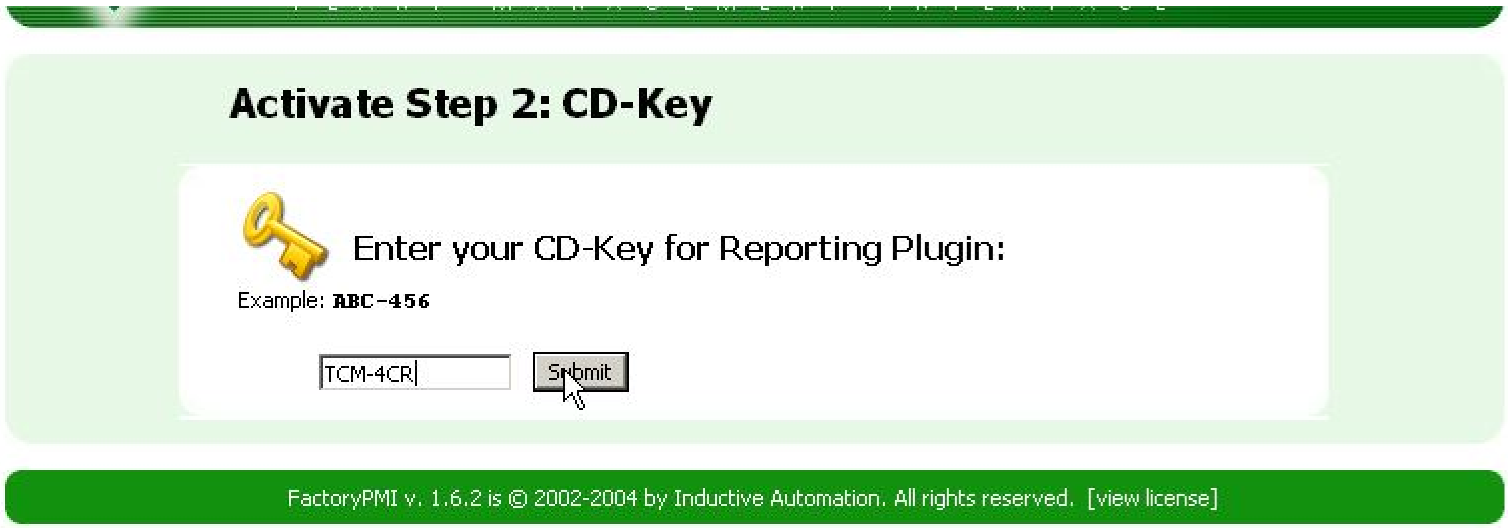
This will bring you to the following page, where you may choose to **purchase** FactoryPMI if you do not have a CD-Key (redirects you to our web site), or click **activate** to continue.



The following screen asks which method you would like to use for activation: Internet or phone. If the computer you are installing on has an Internet connection, choose Internet. It is the quickest way. If the installation computer does not have an Internet connection, click Phone. (NOTE: If an available computer has Internet access, you may complete the Phone activation yourself by going to InductiveAutomation.com/Support/Activate)



Regardless of the method you selected, the following screen will ask you to enter the cd-key. The cd key can be found on the inside insert of the cd case, or if you purchased online, it was displayed at check out and emailed to the address you provided. It is a 6 character string, formatted like XXX-XXX.



Phone Activation

The following screen will appear:

When you call, or when you go to the support web site, you will be asked for the Temporary Install key. This key is different each time you come to this page, so DO NOT CLOSE or refresh the browser window until you have finished, or you will have to start over. After giving this key, you will be given the Activation Key in response. Enter it in the box, and click "Activate".

If everything is successful you will see a thank you screen. Otherwise, you will be given an error message and a hint at what might be wrong.

Internet Activation

Internet activation will try to contact our server and process the request. If everything goes well you will receive a thank-you screen. Otherwise you'll be given an error message.

If Internet activation fails several times, try activating over our web site at support.inductiveautomation.com

Activation Successful

Activation Successful!

Reporting Plugin is now activated! Thank you for your purchase. Please visit Inductive Automation's website or email support@inductiveautomation.com with any questions that you have. Click here to go back to the launch page.

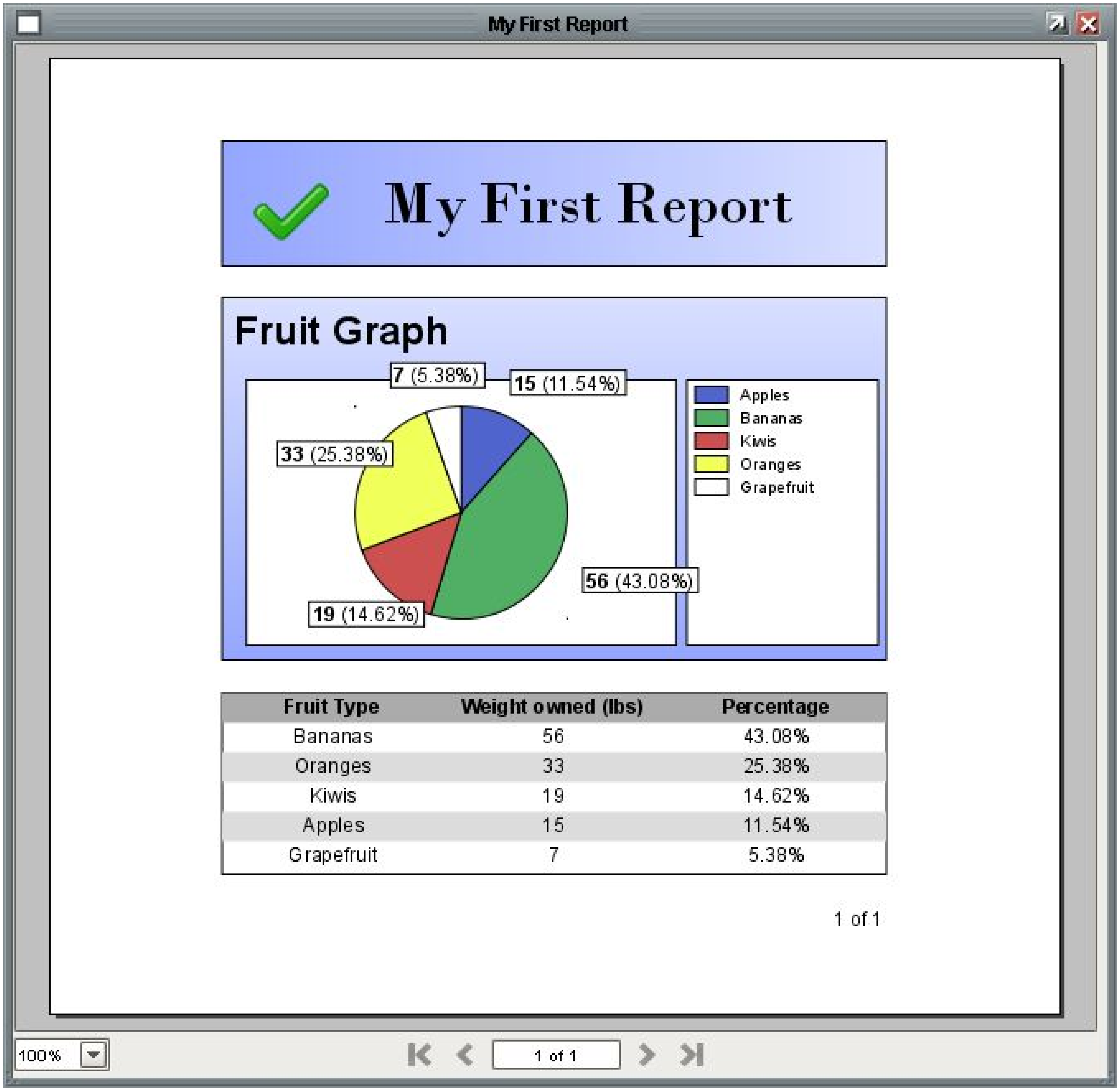
What do I really need to know to get started?

FactoryPMI Reporting is really pretty easy to use. A basic grasp of the following topics, shown in order of precedence, will have you on your way to creating professional reports:

- Understand how data gets into the report via dynamic properties.
- Read how selection works. Pay close attention to superselection. This is **very important**!
- Know that all properties can be modified via the attribute panel or the inspector panel once you select the right object.
- Understand that substitution keys are the way that reports display dynamic data.
- When working with tables and graphs, the DataSet Key defines the FactoryPMI DataSet that will populate the object. Once defined, you may implicitly specify variables under that dataset.
- At this point click through the Quick Start or Tutorial #1.

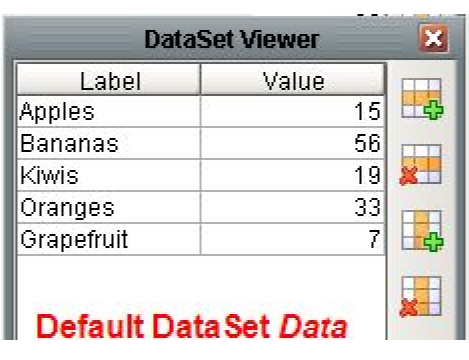
Step by Step Quick Start

This guide steps you through creating basic report that contains a table and pie chart with the default DataSet, *Data*, shown below. Click [here]() to learn how to install the *FactoryPMI Reporting plugin* or [here]() to learn how to populate the report with your own data.



Instructions

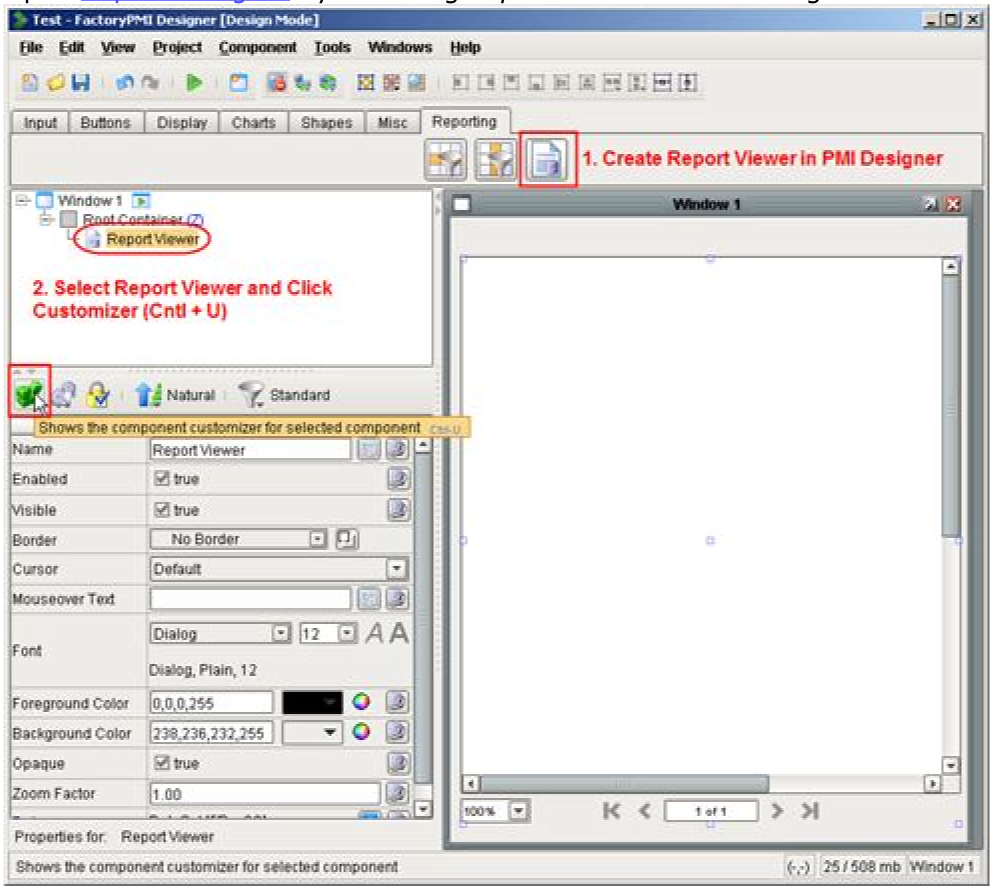
We begin with the default DataSet, *Data*, that comes attached to every [Report Viewer]().



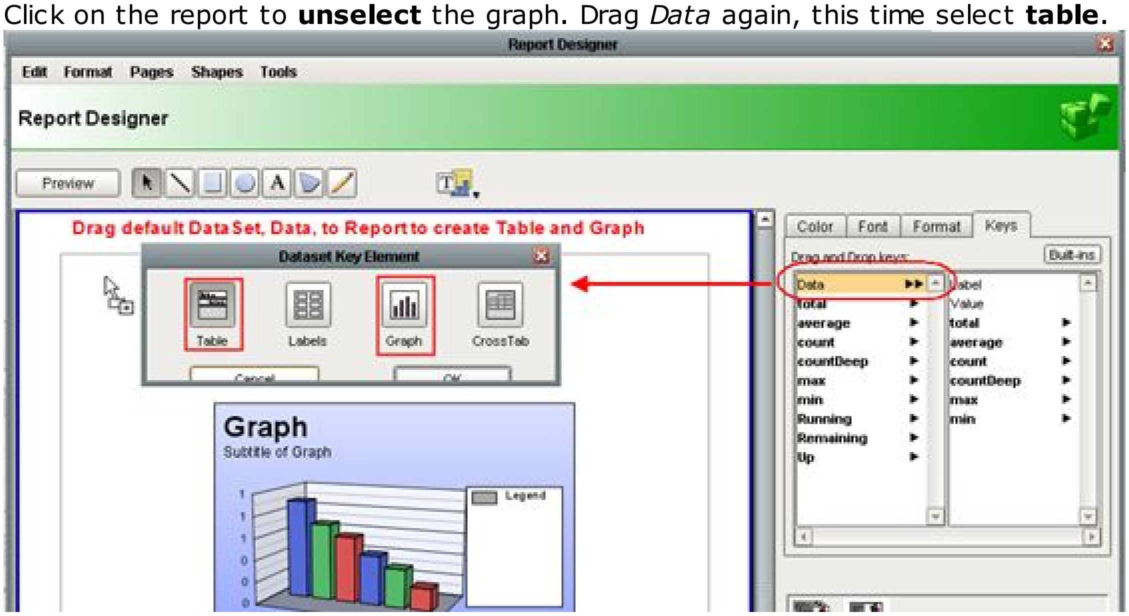
Default DataSet, *Data*

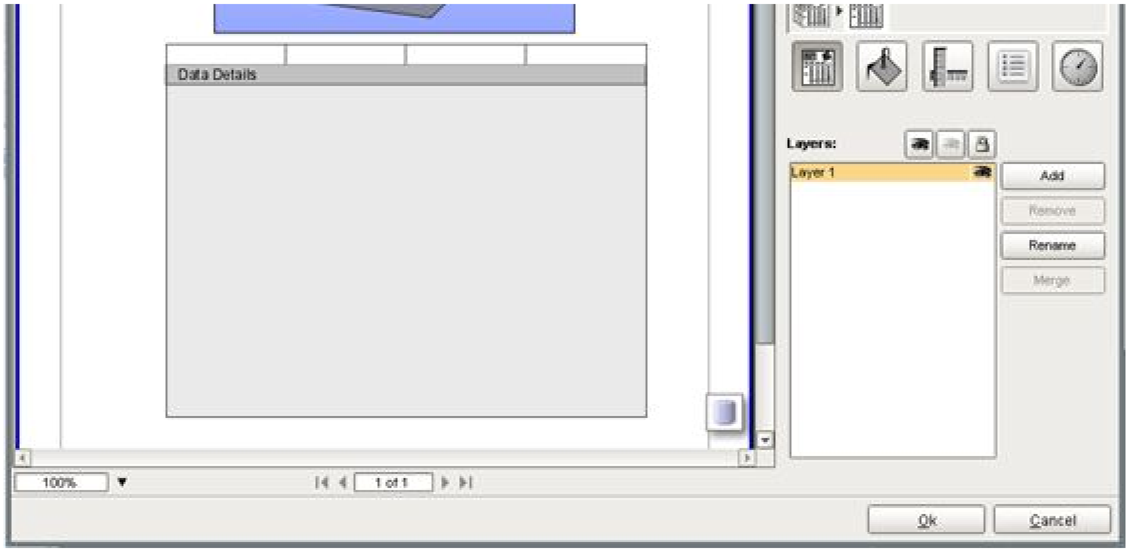
Here are the steps to creating the report:

1. Install FactoryPMI Reporting Plugin
2. Drag *Report Viewer* from Reporting tab into project window
3. Open Report Designer by selecting *Report Viewer* and clicking on the **Customizer**.

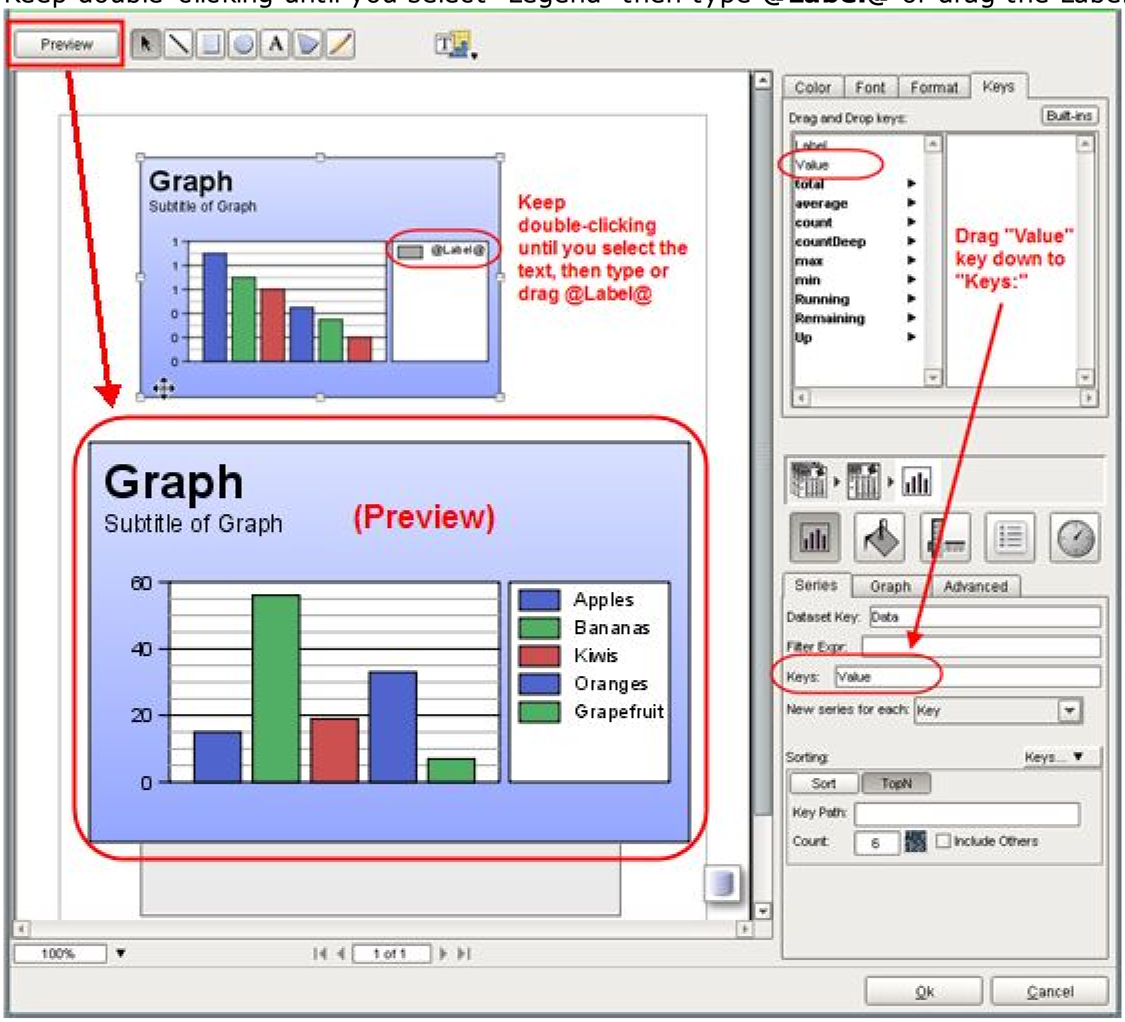


4. Select the keys tab of the Attributes panel and drag *Data* to the report.
5. Select **graph**
6. Click on the report to **unselect** the graph. Drag *Data* again, this time select **table**.

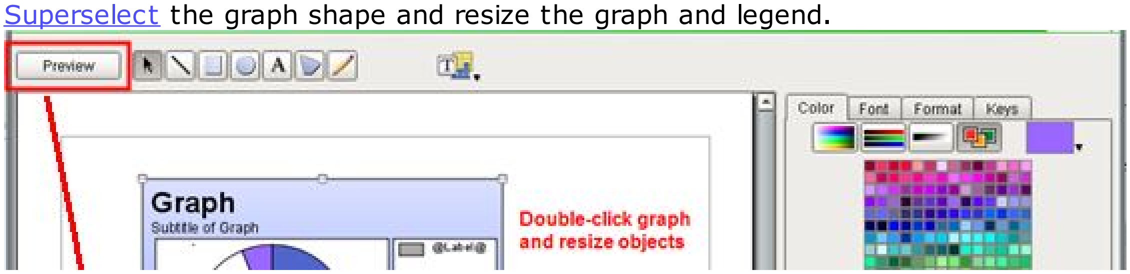


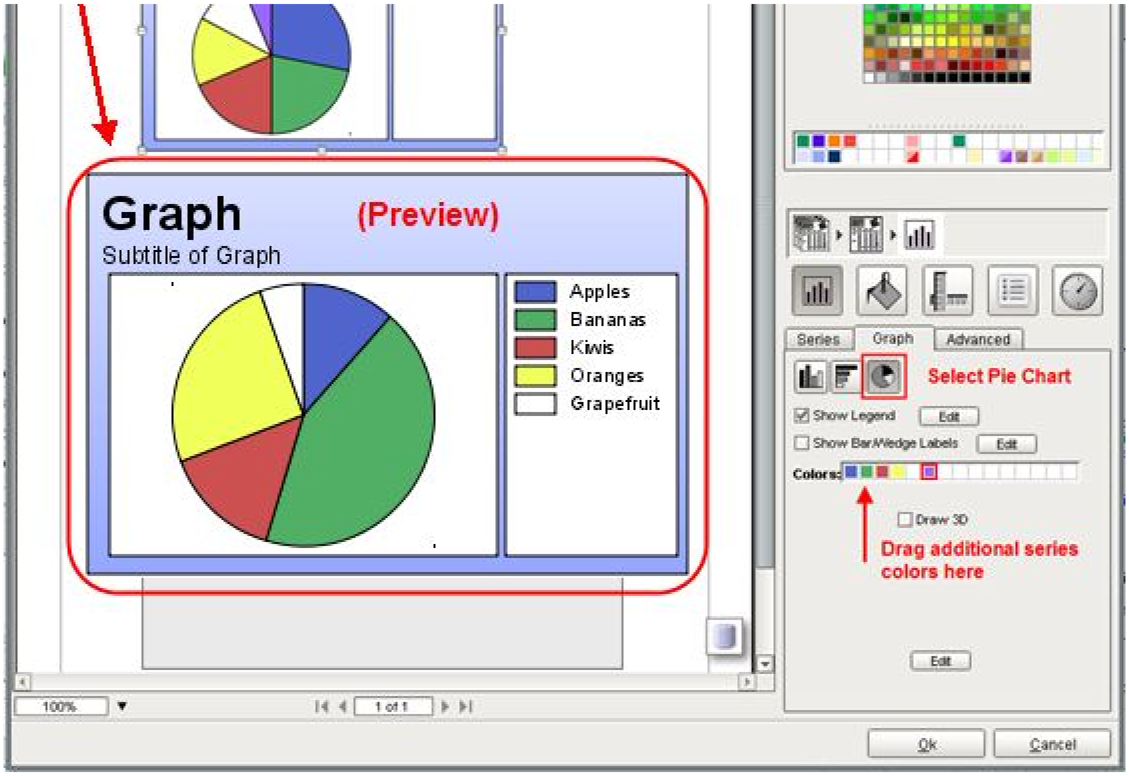


7. Drag *Value* key down to "Keys:" or type **Value**
8. Keep double-clicking until you select "Legend" then type **@Label@** or drag the Label key in.

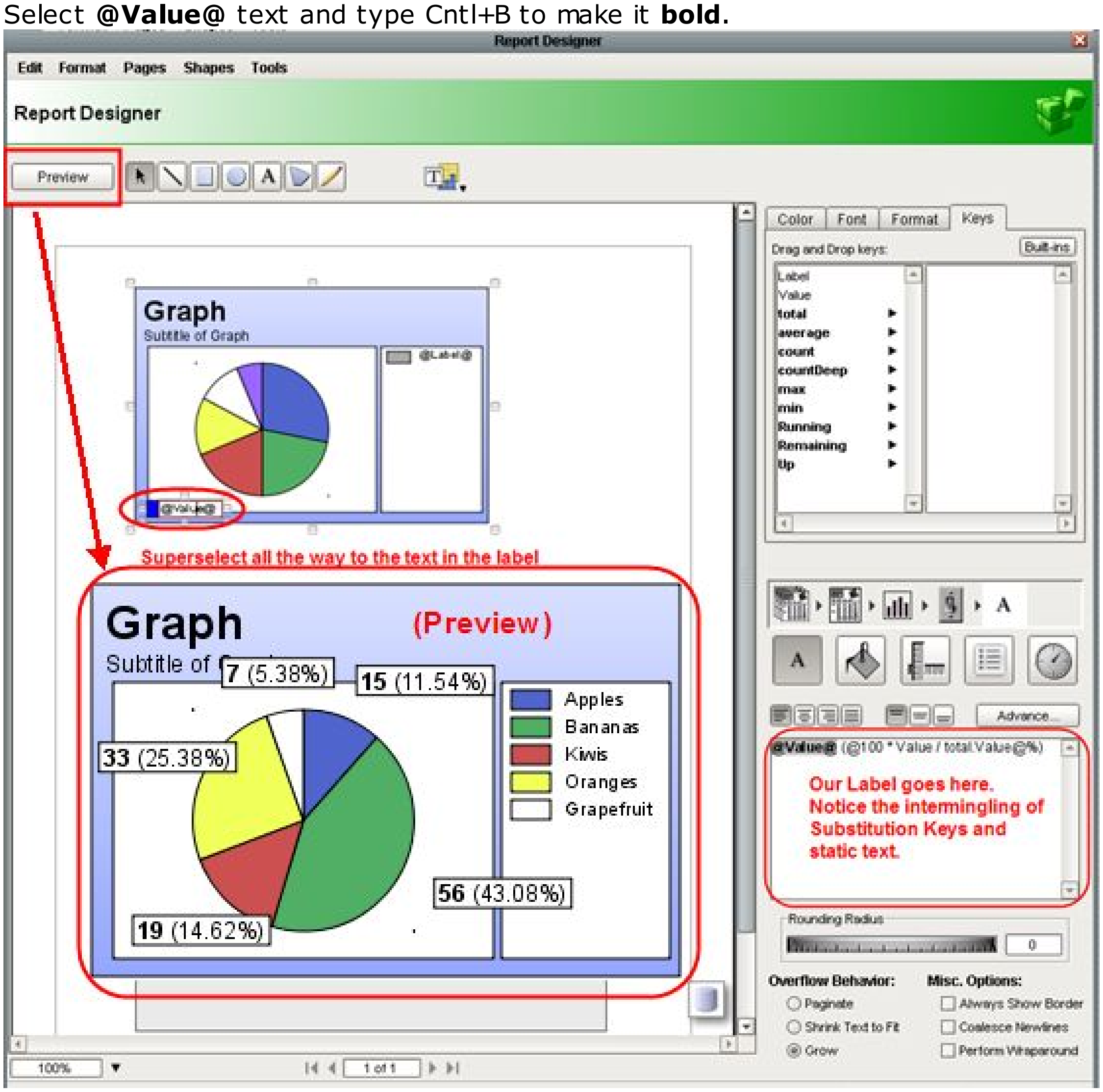


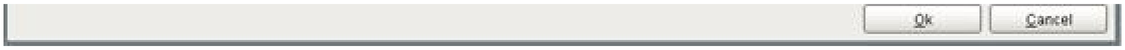
9. On the *Graph* tab of the [Inspector Panel](), select the pie chart icon .
10. Drag colors from the [Color Attribute Panel]() to the graph's series colors.
11. [Superselect]() the graph shape and resize the graph and legend.



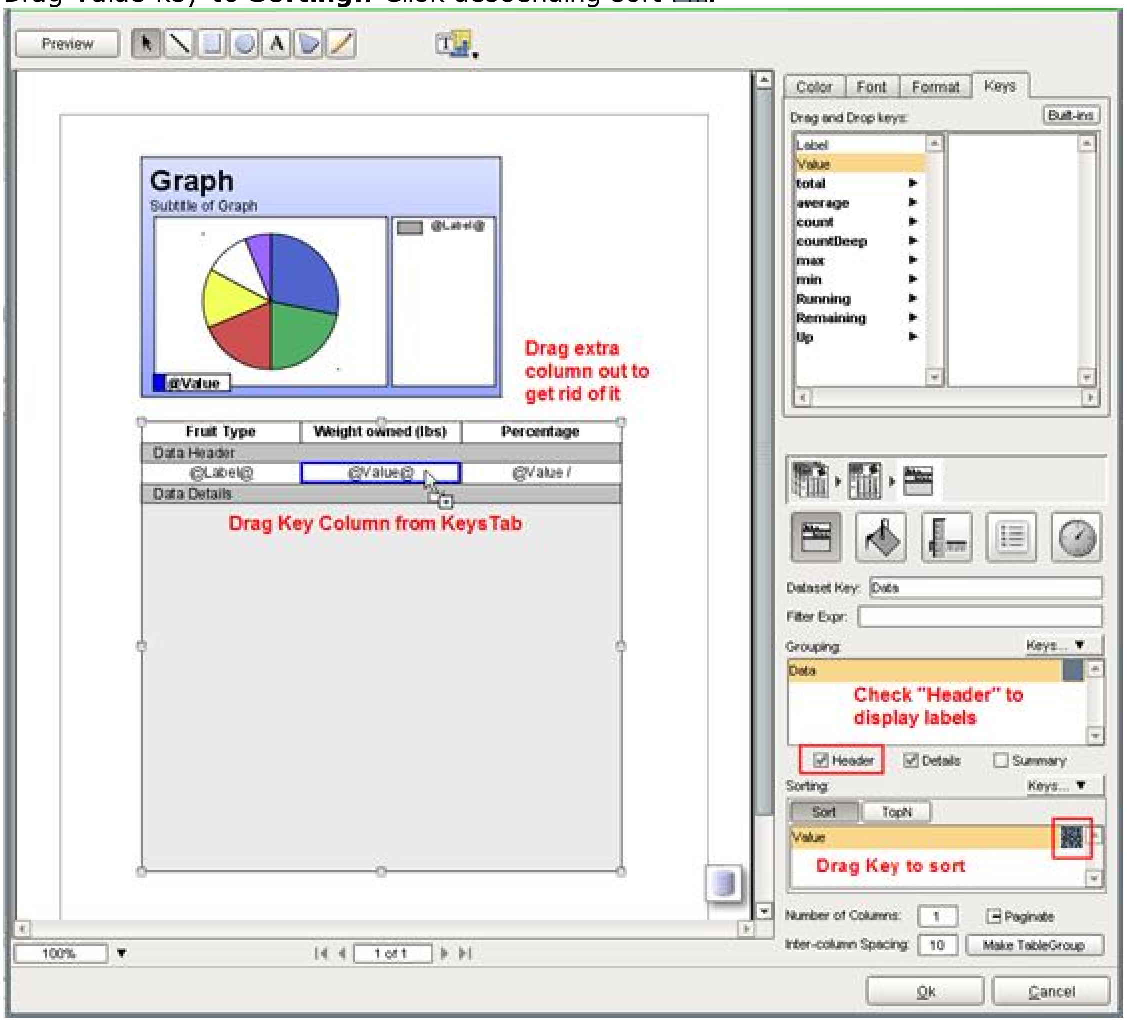


12. On the *Graph* tab, check *Show Bar/Wedge Labels*.
13. Superselect the label text. Change the [font size] to 12 point.
14. Change the text to "**@Value@** (@100 * Value / total.Value@%)". We're intermingling static text and [substitution keys] to display both the value and percentage.
15. Select **@Value@** text and type Cntl+B to make it **bold**.

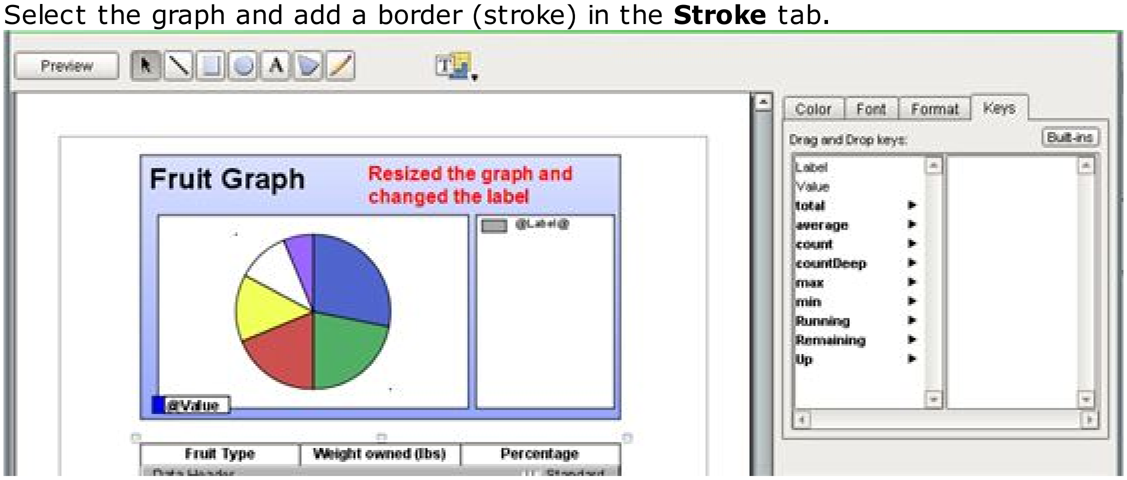


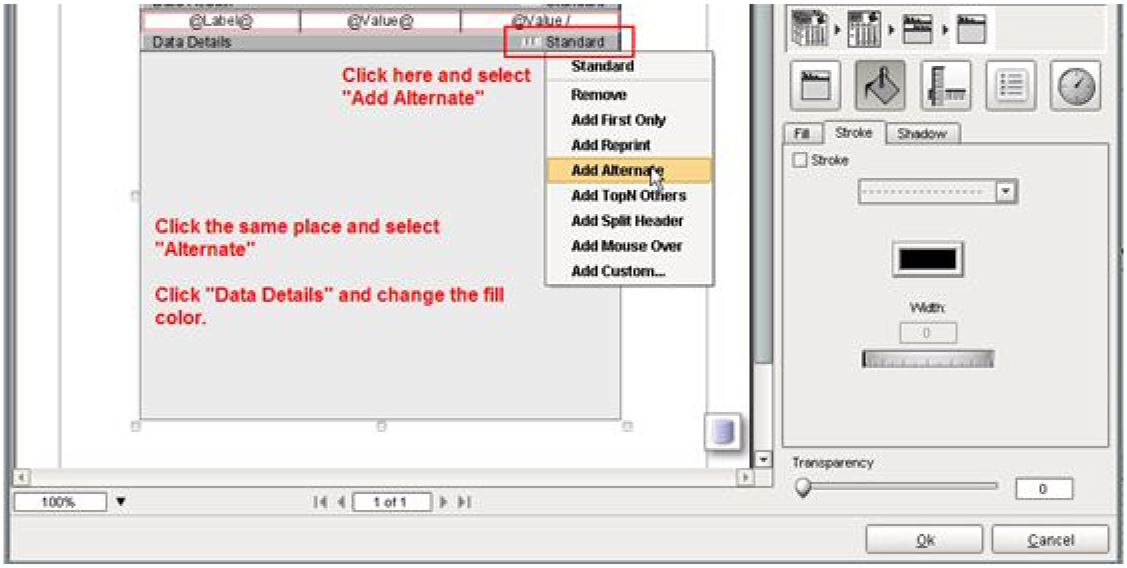


16. Select Table and check **Header**
17. Drag extra column off workspace to get rid of it. This can also be done in the table inspector.
18. Type in headers. In this case we made the text **bold** and centered.
19. Drag key columns to *Data Details* columns.
20. For percentage, we use "@Value / Up.total.Value * 100@%" or "@Value / Data.total.Value * 100@%"
21. Drag Value key to **Sorting:**. Click descending sort.

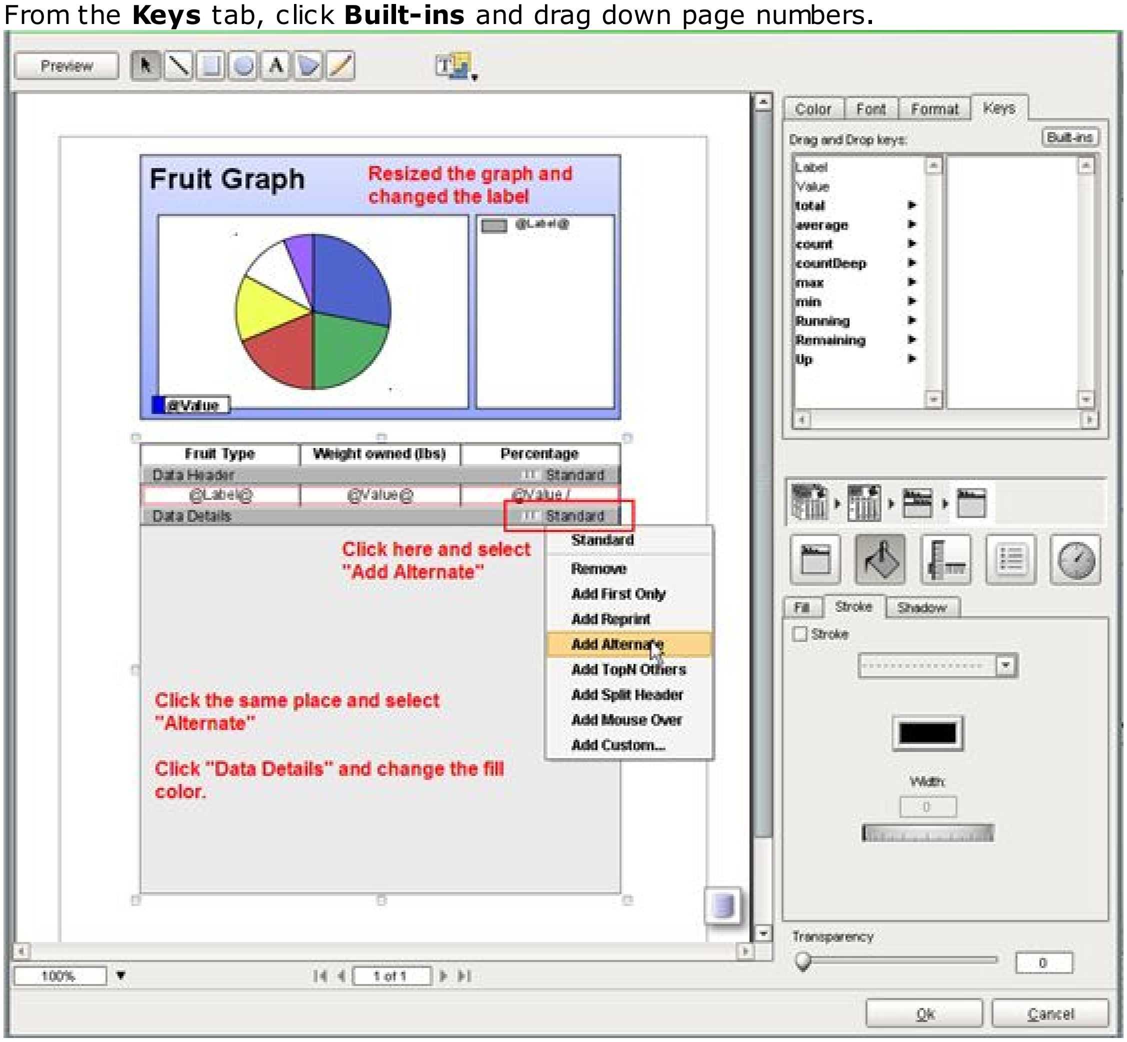


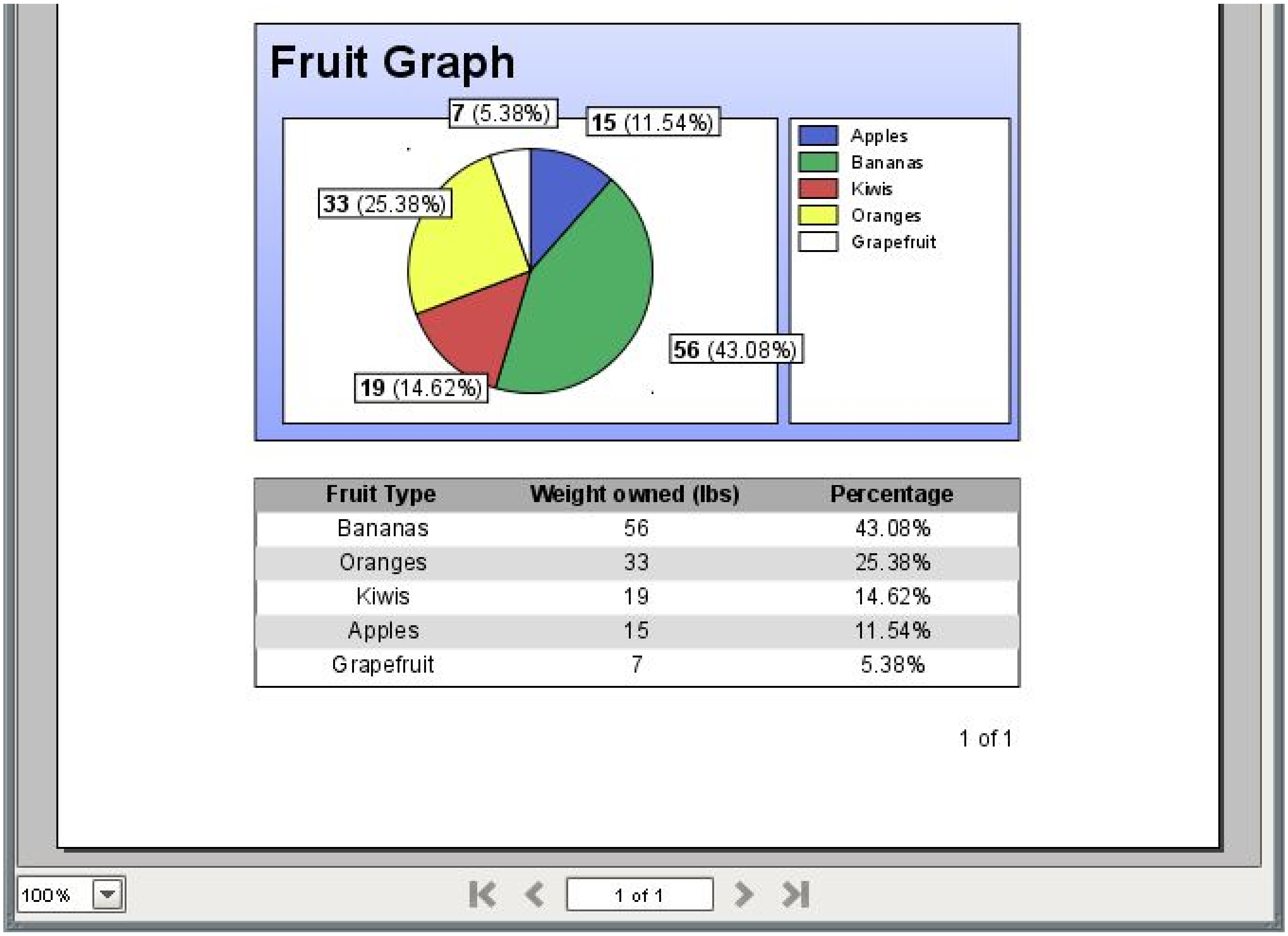
22. Resize the graph and modify the label
23. Double-click the graph, click to add Alternate Row Version
24. Click **Standard** on **Data Details**, select *Alternate*
25. Change the row fill color to gray.
26. Do the same for the *Data Header* **fill color** with a darker gray.
27. Select the graph and add a border (stroke) in the **Stroke** tab.





28. Drag in a gradient filled rectangle, text and the included image *Bultin/icons/48/check2.png* to create our header
29. From the **Keys** tab, click **Built-ins** and drag down page numbers.



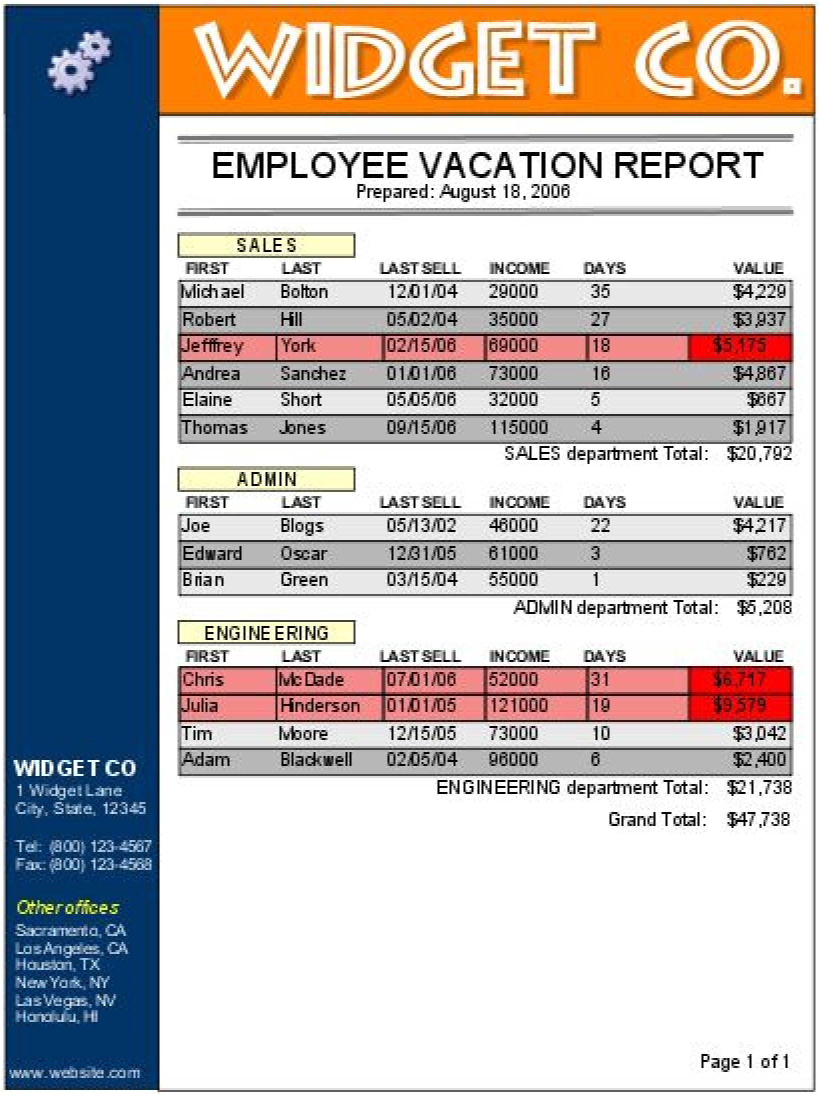


Fruit Type	Weight owned (lbs)	Percentage
Bananas	56	43.08%
Oranges	33	25.38%
Kiwis	19	14.62%
Apples	15	11.54%
Grapefruit	7	5.38%

Our finished first report

Tutorial 1 - Overview

Tutorial #1 takes you through a Widget Co. quarterly employee vacation report. It should give you an idea on how to make a table based report and provide examples of common reporting features. Check out Tutorial 2 for an example of more features.



WIDGET CO.

EMPLOYEE VACATION REPORT

Prepared: August 18, 2006

SALES

FIRST	LAST	LAST SELL	INCOME	DAYS	VALUE
Michael	Bolton	12/01/04	29000	35	$4,229
Robert	Hill	05/02/04	35000	27	$3,937
Jeffrey	York	02/15/06	69000	18	$5,175
Andrea	Sanchez	01/01/06	73000	16	$4,867
Elaine	Short	05/05/06	32000	5	$667
Thomas	Jones	09/15/06	115000	4	$1,917

SALES department Total: $20,792

ADMIN

FIRST	LAST	LAST SELL	INCOME	DAYS	VALUE
Joe	Blogs	05/13/02	46000	22	$4,217
Edward	Oscar	12/31/05	61000	3	$762
Brian	Green	03/15/04	55000	1	$229

ADMIN department Total: $5,208

ENGINEERING

FIRST	LAST	LAST SELL	INCOME	DAYS	VALUE
Chris	McDade	07/01/06	52000	31	$6,717
Julia	Hinderson	01/01/05	121000	19	$9,579
Tim	Moore	12/15/05	73000	10	$3,042
Adam	Blackwell	02/05/04	96000	6	$2,400

ENGINEERING department Total: $21,738

Grand Total: $47,738

WIDGET CO
1 Widget Lane
City, State, 12345

Tel: (800) 123-4567
Fax: (800) 123-4568

Other offices
Sacramento, CA
Los Angeles, CA
Houston, TX
New York, NY
Las Vegas, NV
Honolulu, HI

www.website.com

Page 1 of 1

- Background
- Getting Started
- Basic Layout including headers and footers
- Substitution Keys and Tables including grouping and sorting
- Row Versioning and final touches

Next (Background)

TIP	Create your own report as you go through the tutorial.

Tutorial 1 - Background

WIDGET CO.

Widget Co. is concerned with maximizing the morale of its people. Every employee is entitled 3 days of paid vacation per month. Employees are given the option of selling back their vacation days at 1 1/2 times their normal wage.

The production manager has tasked you with creating a report with the following requirements:

- Look **presentable** - The report will be going to the VP.
- **automatically** display generation date and page numbers. This needs to be a "one click" report.
- **Group** employees by department.
- Be **dynamic** - Widget Co. anticipates rapid growth. The report needs to be able to deal with a large number of employees, possible new departments, and separate pages automatically without cutting off data.
- **Calculate** equations automatically - The manager is interesting in knowing how much money in vacation time each employee is owed, as well as a running total by department.
- **Sort** employees by vacation days. Widget Co. gives preferential approval to the employee with the most days.
- Support **custom row versions**. A special paid vacation is offered when an employees vacation sellback value exceeds $5000. Such employees need to stand out!

Employee data can be retrieved from the accounting database with the following SQL query:

```
SELECT * FROM employees;
```

	first	last	department	lastSell	income	withholdings	vacationdays
1	Joe	Blogs	ADMIN	2002-05-13 00:00:00	46000	5000	22
2	Julia	Hinders...	ENGINEERING	2005-01-01 00:00:00	121000	10000	19
3	Adam	Blackwell	ENGINEERING	2004-02-05 00:00:00	96000	8000	6
4	Robert	Hill	SALES	2004-05-02 00:00:00	35000	200	27
5	Brian	Green	ADMIN	2004-03-15 00:00:00	55000	4000	1
6	Elaine	Short	SALES	2006-05-05 00:00:00	32000	2000	5
7	Tim	Moore	ENGINEERING	2005-12-15 00:00:00	73000	4000	10
8	Chris	McDade	ENGINEERING	2006-07-01 00:00:00	52000	4000	31
9	Andrea	Sanchez	SALES	2006-01-01 00:00:00	73000	5000	16
10	Thomas	Jones	SALES	2006-09-15 00:00:00	115000	12000	4
11	Jefffrey	York	SALES	2006-02-15 00:00:00	69000	1000	18
12	Michael	Bolton	SALES	2004-12-01 00:00:00	29000	500	35
13	Edward	Oscar	ADMIN	2005-12-31 00:00:00	61000	3000	3

We will modify our SQL query to include the derived value **buyout**, the monetary value of employee's vacation days.

CAST is used so that MySQL returns **buyout** as a number instead of a string.

```
SELECT *, CAST(income/360 * 1.5 * vacationdays AS SIGNED) AS buyout FROM employees;
```

	first	last	department	lastSell	income	withholdings	vacationdays	buyout
1	Joe	Blogs	ADMIN	2002-05-13 00:00:00	46000	5000	22	4217
2	Julia	Hinderson	ENGINEERING	2005-01-01 00:00:00	121000	10000	19	9579
3	Adam	Blackwell	ENGINEERING	2004-02-05 00:00:00	96000	8000	6	2400
4	Robert	Hill	SALES	2004-05-02 00:00:00	35000	200	27	3937
5	Brian	Green	ADMIN	2004-03-15 00:00:00	55000	4000	1	229
6	Elaine	Short	SALES	2006-05-05 00:00:00	32000	2000	5	667
7	Tim	Moore	ENGINEERING	2005-12-15 00:00:00	73000	4000	10	3042
8	Chris	McDade	ENGINEERING	2006-07-01 00:00:00	52000	4000	31	6717

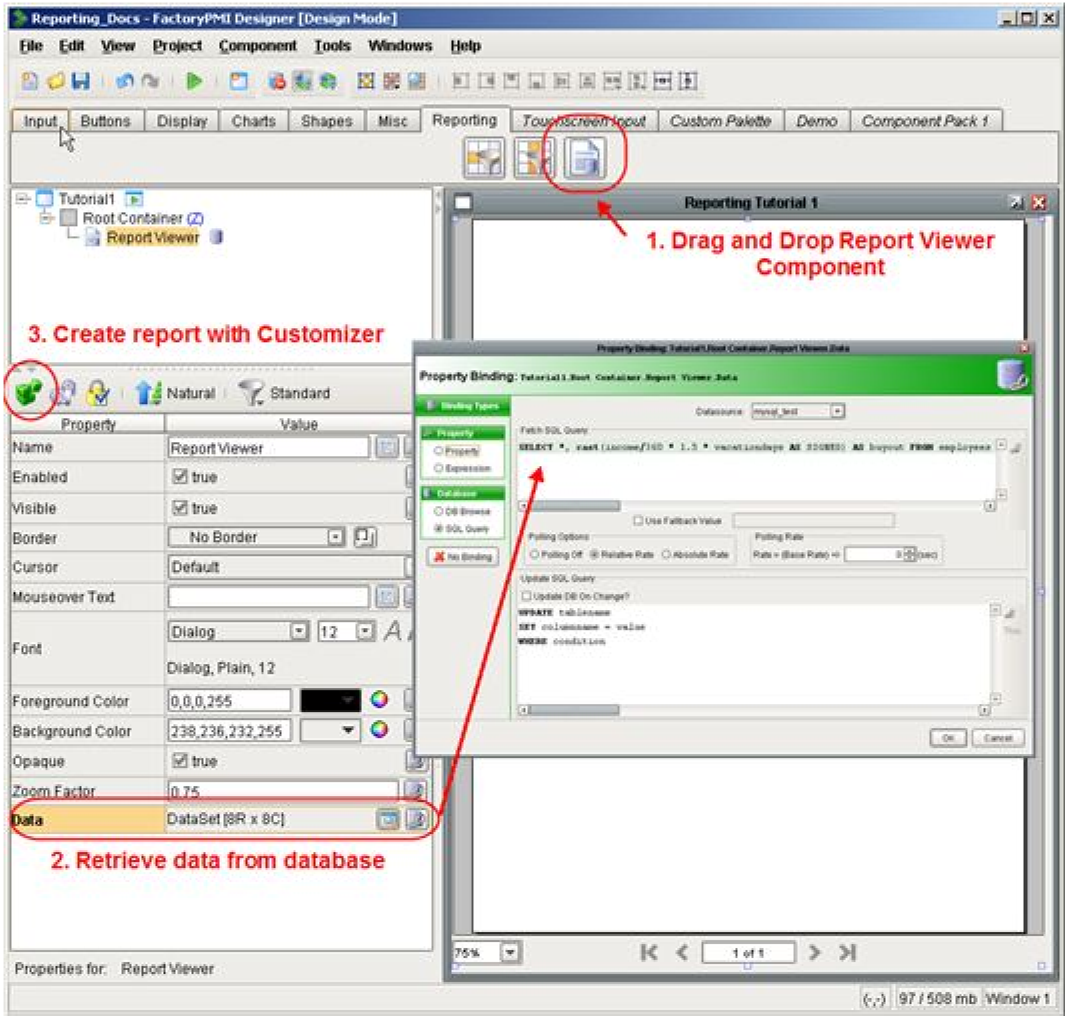
9	Andrea	Sanchez	SALES	2006-01-01 00:00:00	73000	5000	16	4867
10	Thomas	Jones	SALES	2006-09-15 00:00:00	115000	12000	4	1917
11	Jefffrey	York	SALES	2006-02-15 00:00:00	69000	1000	18	5175
12	Michael	Bolton	SALES	2004-12-01 00:00:00	29000	500	35	4229
13	Edward	Oscar	ADMIN	2005-12-31 00:00:00	61000	3000	3	762

Previous (Index)   Next (Getting Started)

TIP	We could use expressions within the report to calculate **buyout**. In this tutorial we use the SQL database because we will be using **buyout** in many places. We will: display it as a column, use it as the basis of our custom row versions, and may want the option of sorting our report based on it. Other reasons include: leveraging the SQL database's rich function library and only needing to change the expression in one place.

Tutorial 1 - Getting Started

We begin by installing the Reporting plugin and creating a report in our project within the FactoryPMI designer.

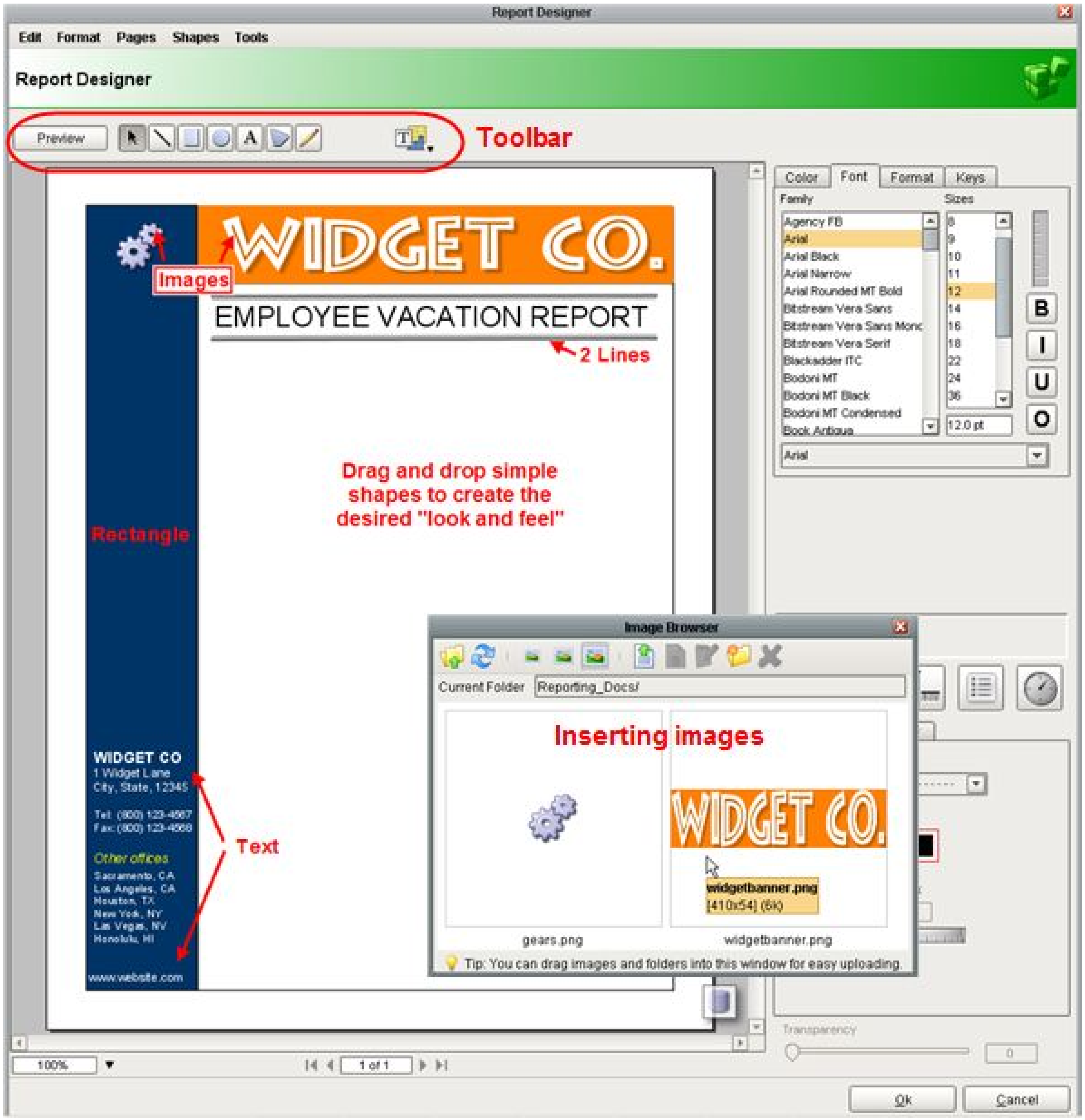


1. Install the FactoryPMI Reporting Plugin
2. Create a new Window, and drag down a Report Viewer from the **Reporting** tab.
3. Populate the **Data** dynamic dataset. Note: You can customize your own Reporting datasets.
4. Select the Report Viewer component and click on the Customizer (Cntl+U). This is where you will be creating the report.

Index Previous (Background) Next (Basic Layout)

Tutorial 1 - Basic Layout

Clicking on the customizer with the Report Viewer opens the **Report Designer** window where we create our report.



Everything here was created with the toolbar.The following steps were taken:

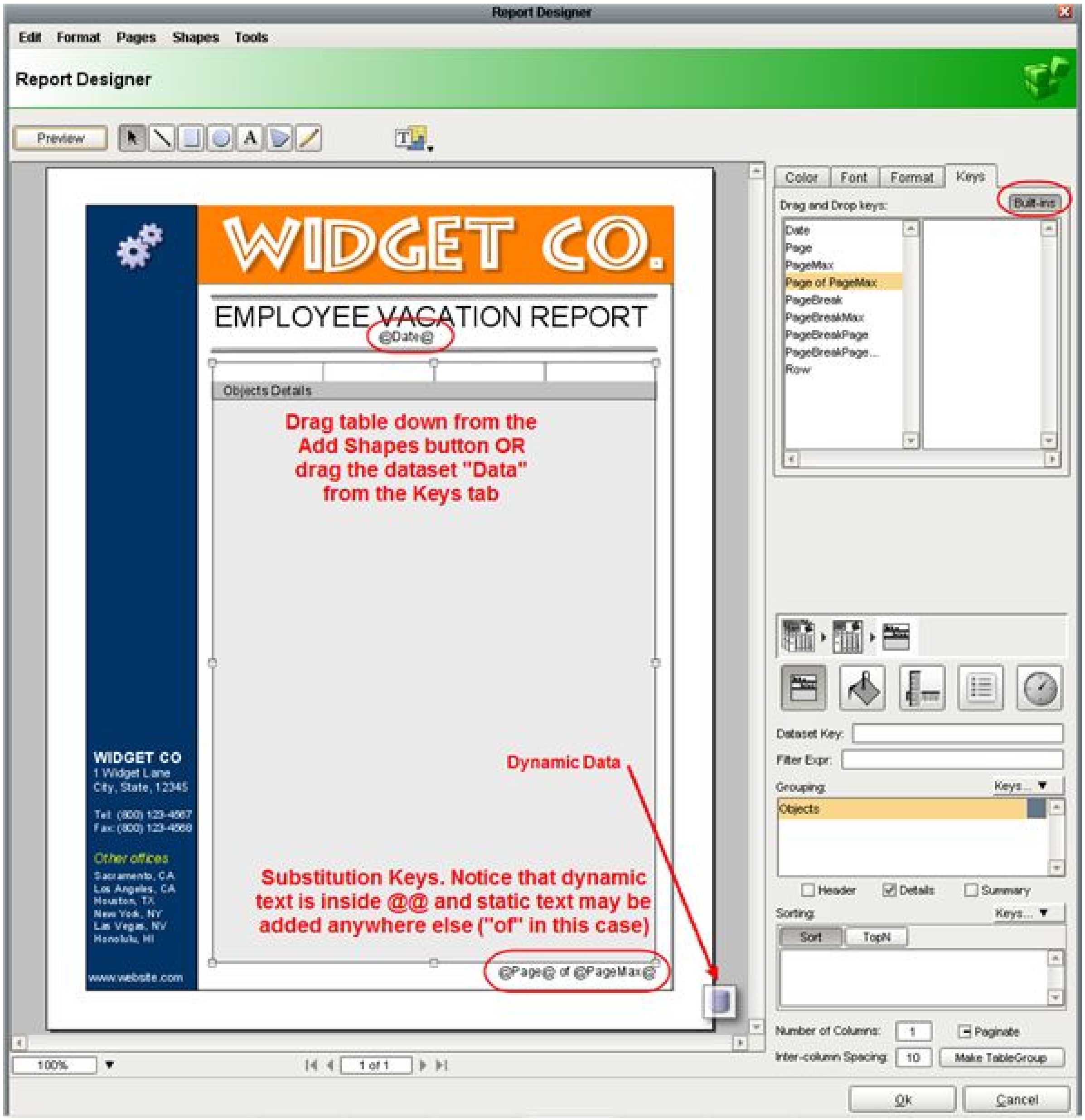
1. Drag the left and top rectangles. Modify their fill property in the attributes panel to create the blue and orange background colors.
2. Drag another border rectangle for good measure.
3. Add Shapes->Image. Repeat for gears and header image.
4. Add Text. Modify applicable properties in their inspector and attributes panels. Double clicking text for superselection is key here!
5. Add 4 Lines. Modify applicable properties in their inspector and attributes panels.

Index   Previous (Getting Started)   Next (Substitution Keys and Tables)

TIP	Borrow the look and feel from existing reports!

Tutorial 1 - Substitution Keys and Tables

The most interesting portion of this report will be a table. It will occupy as much space as the size that we drag it on the screen, creating extra pages as necessary for the data.



Adding Page Numbers

1. Select the keys tab, click the **Built-ins** button, then drag the *Date key* into the report header. The cursor will change to the *Drag Key* icon . Releasing the mouse button will place a label with the text "**@Date@**".
2. Repeat, dragging the builtin key "*Page @Page@ of @PageMax@*" to the footer (bottom) of the report.
3. Format the date by double clicking the text label to superselect the text, then use the formatter to format the date.

Creating Table

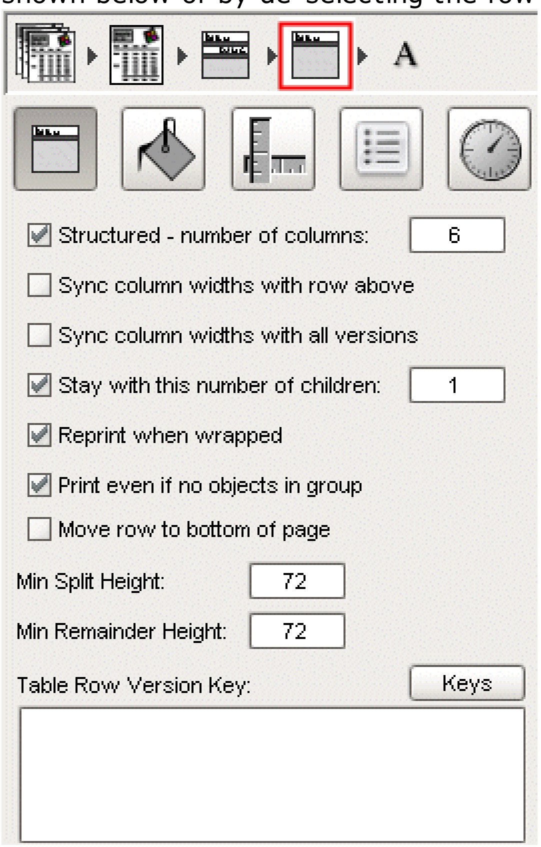
1. Drag the DataSet *Data* from the keys tab and choose table when you see the **Dataset Key Element** window above. Select **table** and click ok.

   Alternatively, create the table from the toolbar then drag down the *Data* key to the *Dataset key* field of the keys attribute panel. Defining the table's DataSet is done automatically when using the step above.

2. Resize and position the window as desired.

Table Customization

1. Select the table, and select to the Table Inspector panel. Clicking on the Shape Specific Inspector will bring up the same panel.
2. Select *Data* under grouping and check: Header, Details, and Summary. This creates a unique header and summary row for each unique department. **Data Details** refers to each employee.
3. Select *department* under grouping and check: Details, and Summary. **Summary** creates a single summary row for the report. **Details** at this level of grouping is just above as the **Data Header** level of *Data* (We could have used either one instead of both for this example). More on table row grouping precedence here.
4. Modify the structured column width property for each row. This defines how many columns, of a fixed user definable width, a given table row will use. We will use 6 columns for **Data Details**, and not use structured columns for the others. Not using structured column width can be set by unchecking the top checkbox shown below or by de-selecting the row's prison icon  when the table is superselected.

   Superselect table, then single click select row to pull up this inspector

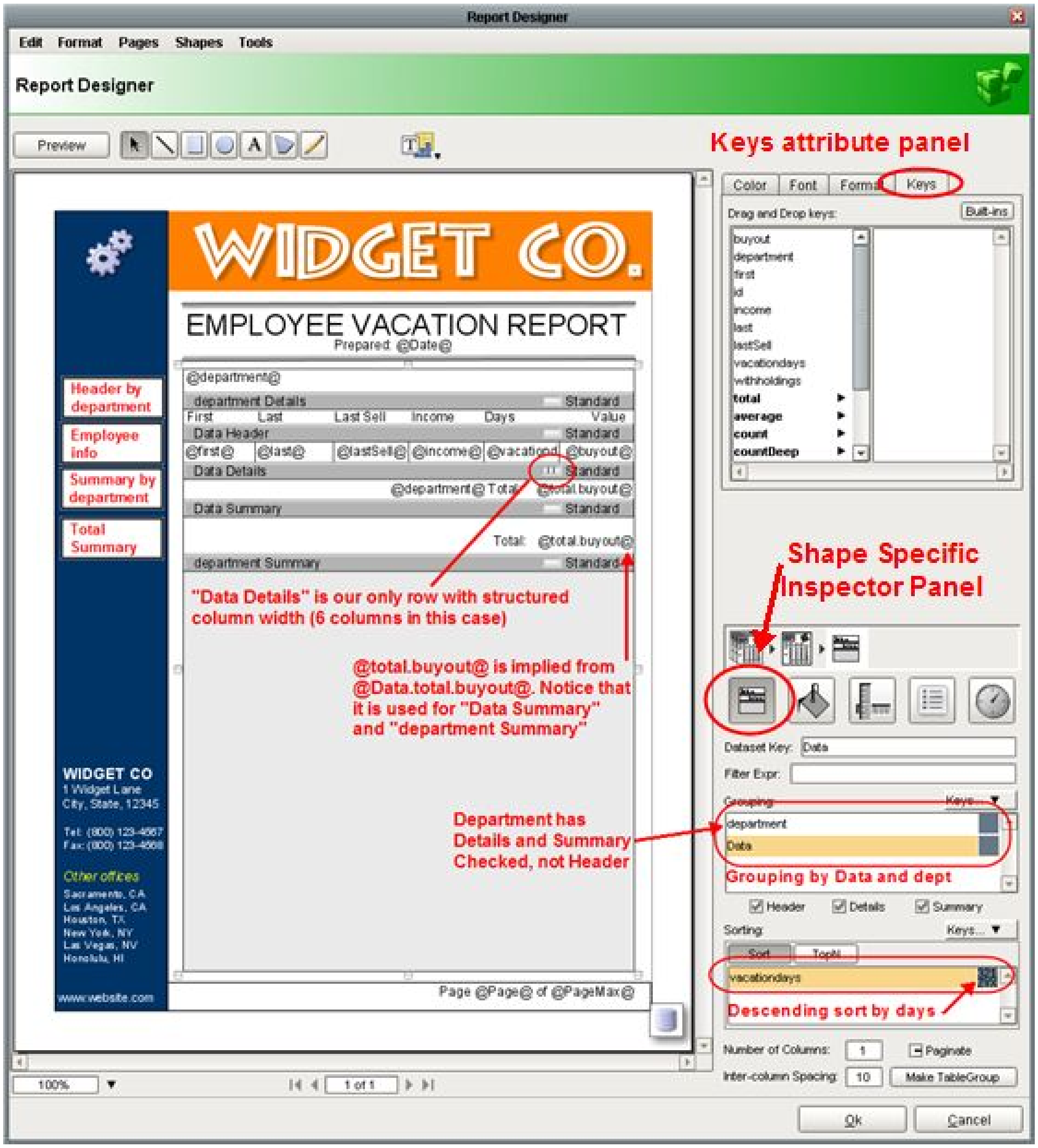
5. Drag keys into table row columns. See the six columns in the **Data Details** row of the screenshot below. Notice the use of text editing, and text formatting.

   The **total** aggregate key, @total.buyout@, is used for both departmental subtotals and the grand total. The difference lies in the level of grouping it is placed in and is explained here.

Sorting and Grouping

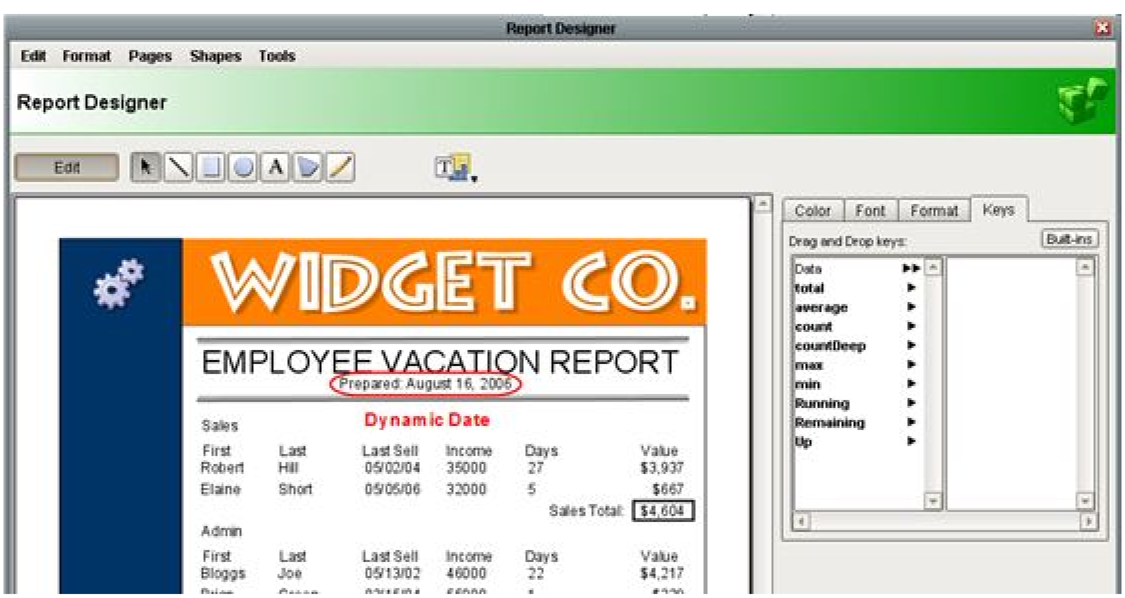
1. From the keys attribute panel, double click to drill down **Data**, then drag down **department** to the *grouping* field. This will automatically group our table by the **department** key. Each unique value of department will be represented by a separate table with the employees from that department.

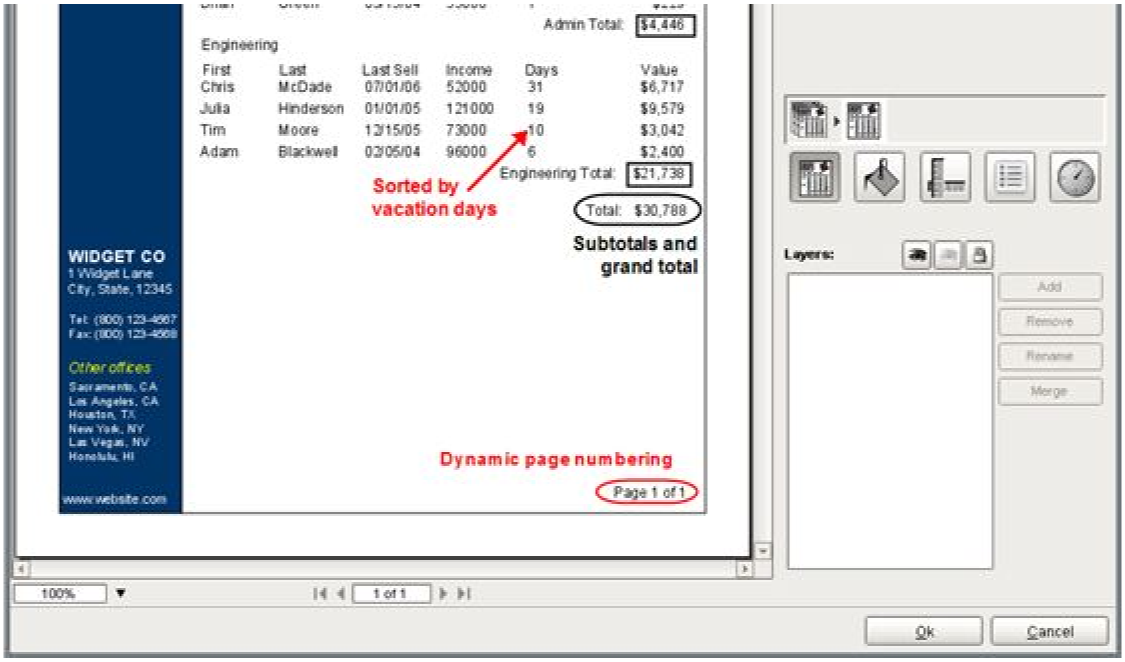
2. From the keys attribute panel, drag down **vacationdays** to the *sorting* field then click the sort icon from ascending ABC 123 to descending 321 ZYX. This sorts our employees by vacation days from most to least.



Preview

1. Click the preview button Preview to view your report.



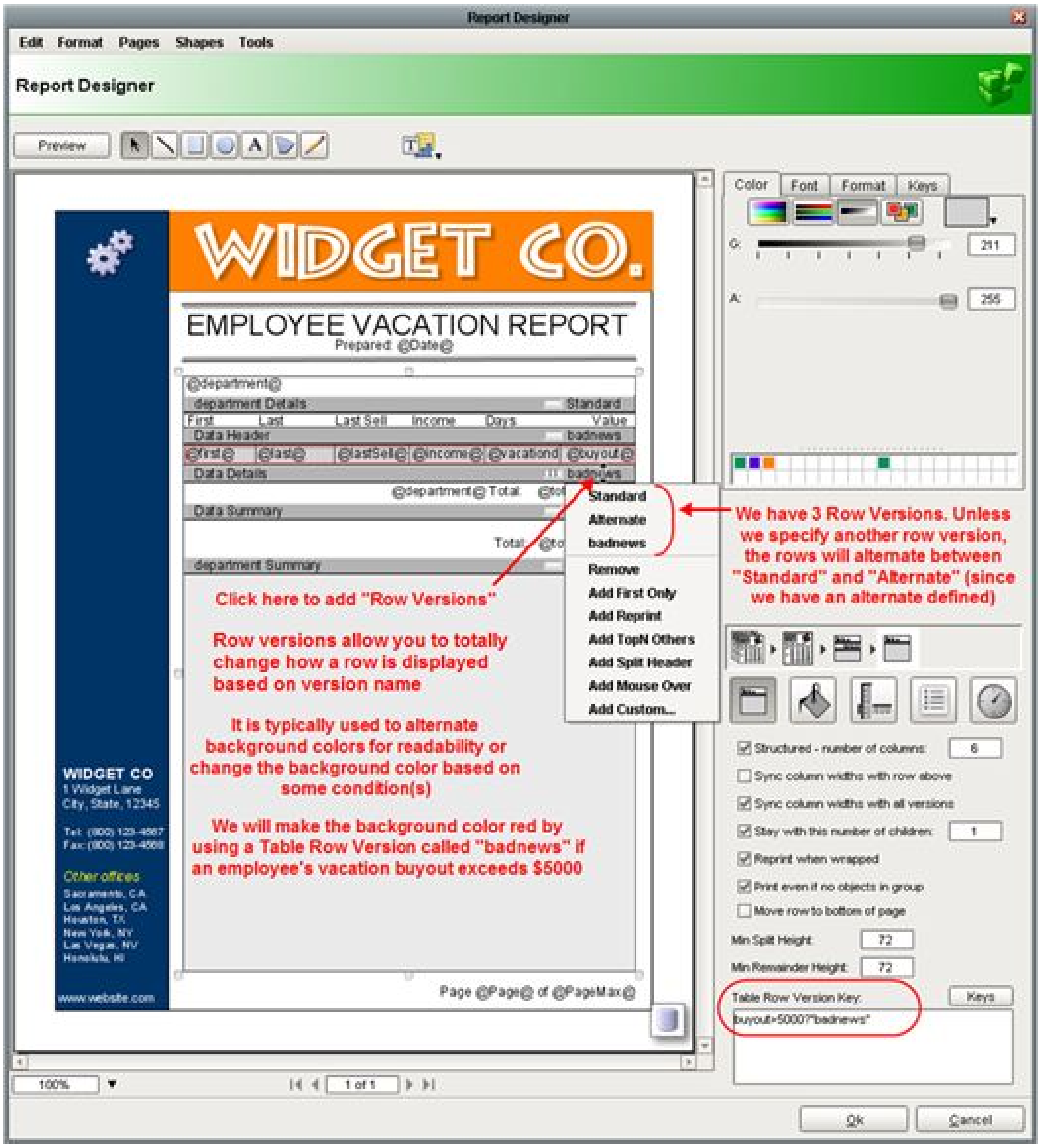


Index Previous Next (Row Versioning)

TIP	Don't be afraid to play with these options! It is much easier than it looks!

Tutorial 1 - Row Versioning

Now we want to color in the rows, and create different row versions for those employees that are entitled more than $5000.



Here were the steps for this report

1. We begin by customizing the *Standard* Row Version, created by default for header, details, and summary. Click **Data Details** to select the row and use the fill/stroke inspector to add a background color (fill) and border (stroke). Resize columns and try to make all adjustments now as duplicate rows will be based on this one.
2. Click on the Row Version Label (Where the image shows **Click here to add "Row Versions"**) and click "Add Alternate".
3. Customize *Alternate* rows. In this case our only change was to darken the background color.
4. Click on the Row Version Label (Where the image shows **Click here to add "Row Versions"**) and click "Add Custom". Add *badnews*
5. Customize *badnews* rows. To illustrate flexibility, added borders to the individual key labels, changed background colors, and modified font properties including the bold property and text centering.
6. Double click on the table then click on **Data Details** to select the **Data Details** row. Select the shape specific inspector property. Under **Table Row Version Key:** we enter:

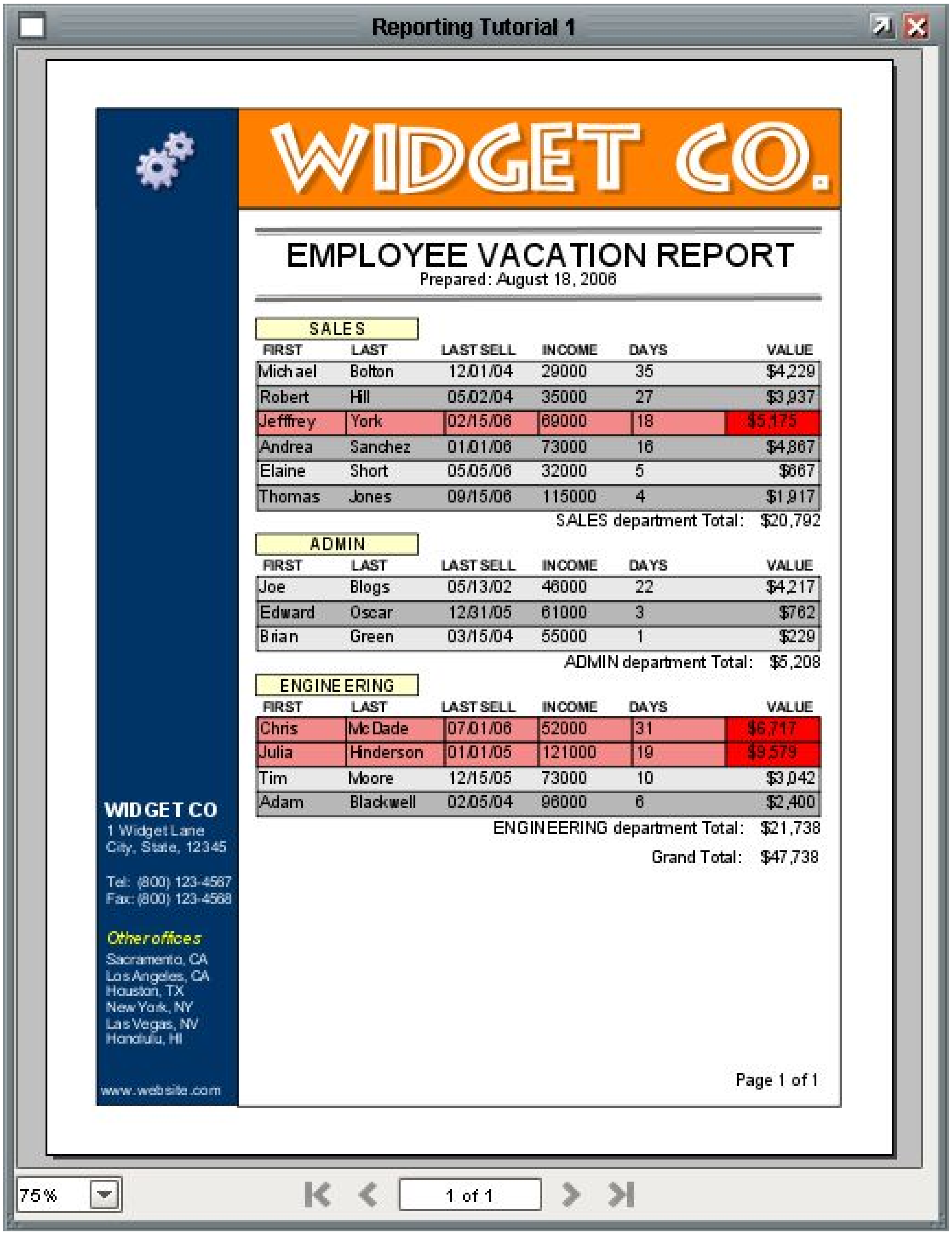
```
buyout >5000?"badnews"
```

How it works: This conditional statement will return the string "badnews" if **buyout** exceeds $5000 for a given employee, changing the row version to *badnews* for that person. We intentionally don't specify an ELSE condition. Since a valid string is not returned, the report will default to using *Standard*, *Alternate*, or whatever builtin row versions are defined.

```
buyout>5000?"badnews":"Alternate"
```

Would make employees show up as our *Alternate* dark gray or *badnews* red. *Standard* would never be displayed. Note: Versions are different for each row, and they each have their own defining *Table Row Version Key*

7. Make final minor cosmetic changes



FIRST	LAST	LAST SELL	INCOME	DAYS	VALUE
Michael	Bolton	12/01/04	29000	35	$4,229
Robert	Hill	05/02/04	35000	27	$3,937
Jeffrey	York	02/15/06	69000	18	$5,175
Andrea	Sanchez	01/01/06	73000	16	$4,867
Elaine	Short	05/05/06	32000	5	$667
Thomas	Jones	09/15/06	115000	4	$1,917
				SALES department Total:	$20,792
Joe	Blogs	05/13/02	46000	22	$4,217
Edward	Oscar	12/31/05	61000	3	$762
Brian	Green	03/15/04	55000	1	$229
				ADMIN department Total:	$5,208
Chris	McDade	07/01/06	52000	31	$6,717
Julia	Hinderson	01/01/05	121000	19	$9,579
Tim	Moore	12/15/05	73000	10	$3,042
Adam	Blackwell	02/05/04	96000	6	$2,400
				ENGINEERING department Total:	$21,738
				Grand Total:	$47,738

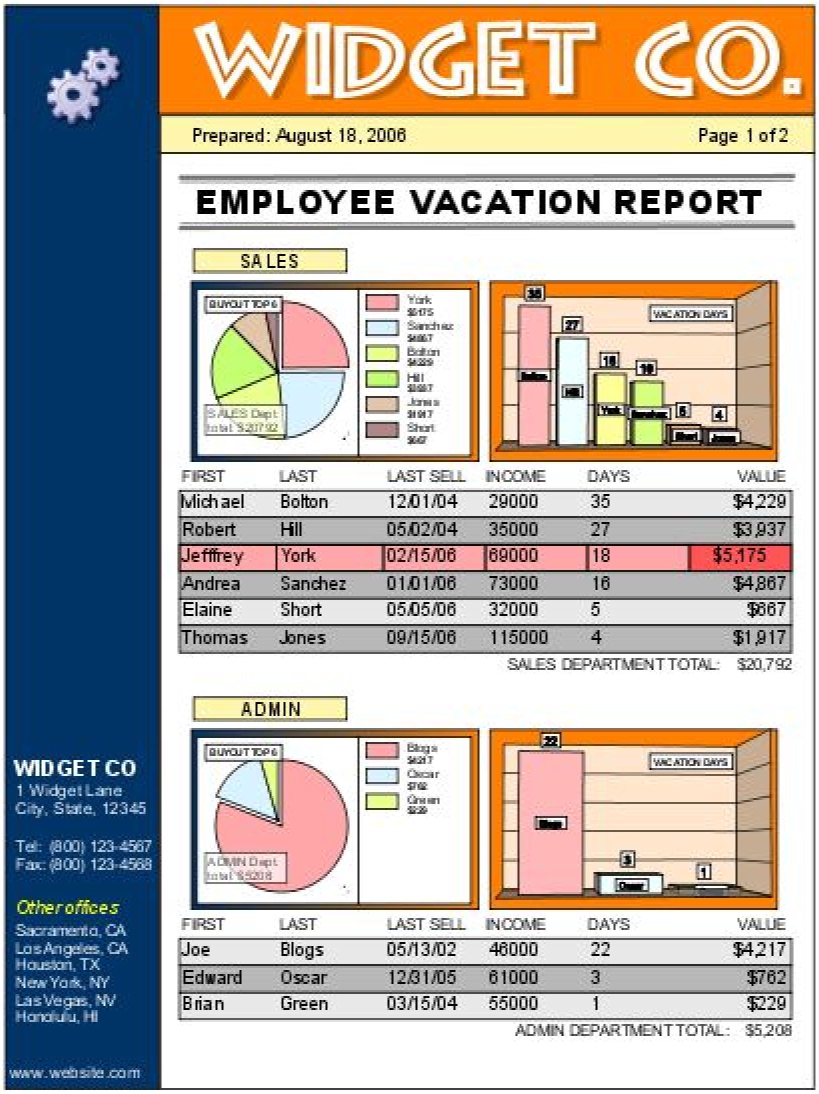
Done for now!

Index Previous (Substitution Keys) On to Tutorial 2

TIP	Borrow the look and feel of an existing report! This is much easier than it looks!

Tutorial 2 - Overview

Tutorial #2 adds dynamic graphs to the Widget Co. quarterly employee vacation report in Tutorial 1. We will make changes to the main table, have a unique header for the first page, and create a report summary for all employees. We will also add an extra dataset, polling data from two datasources.



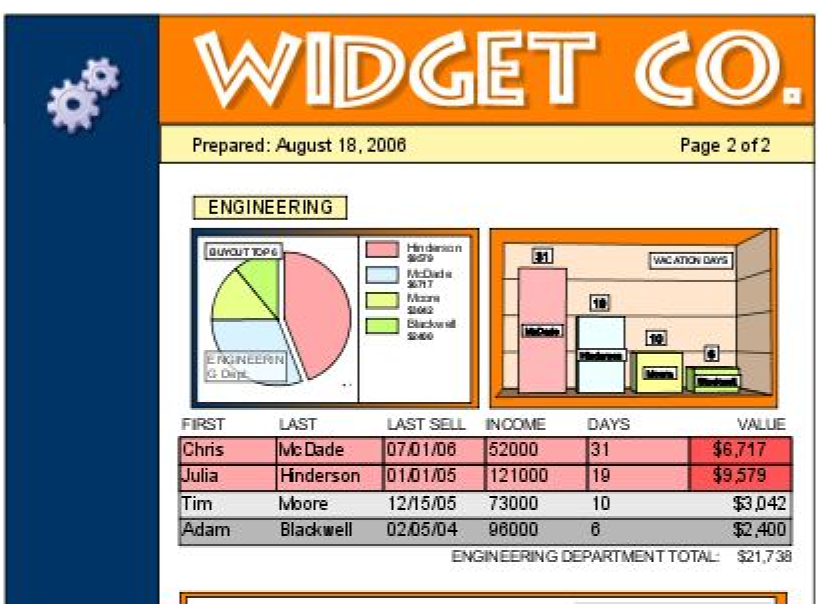
FIRST	LAST	LAST SELL	INCOME	DAYS	VALUE
Michael	Bolton	12/01/04	29000	35	$4,229
Robert	Hill	05/02/04	35000	27	$3,937
Jeffrey	York	02/15/06	69000	18	$5,175
Andrea	Sanchez	01/01/06	73000	16	$4,867
Elaine	Short	05/05/06	32000	5	$667
Thomas	Jones	09/15/06	115000	4	$1,917

SALES DEPARTMENT TOTAL: $20,792

FIRST	LAST	LAST SELL	INCOME	DAYS	VALUE
Joe	Blogs	05/13/02	46000	22	$4,217
Edward	Oscar	12/31/05	61000	3	$762
Brian	Green	03/15/04	55000	1	$229

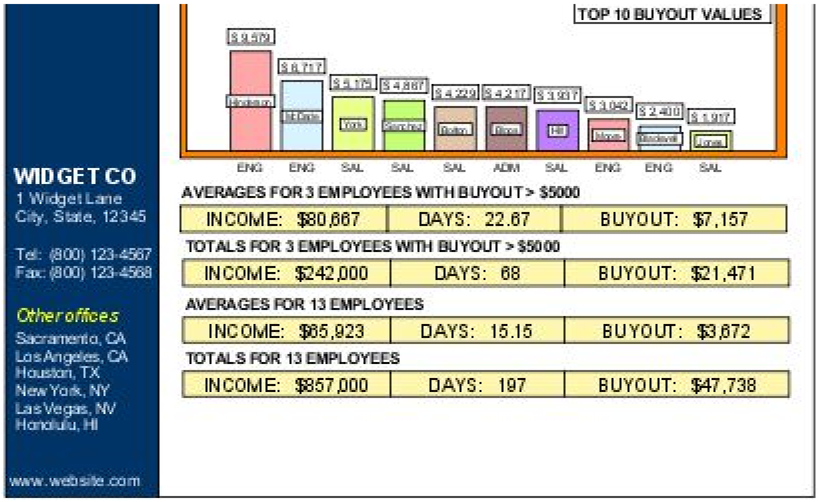
ADMIN DEPARTMENT TOTAL: $5,208

We add dynamic graphs to the report.



FIRST	LAST	LAST SELL	INCOME	DAYS	VALUE
Chris	McDade	07/01/06	52000	31	$6,717
Julia	Hinderson	01/01/05	121000	19	$9,579
Tim	Moore	12/15/05	73000	10	$3,042
Adam	Blackwell	02/05/04	96000	6	$2,400

ENGINEERING DEPARTMENT TOTAL: $21,738



Notice that the **EMPLOYEE VACATION REPORT** label only exists on the first page. Every page in Tutorial 1 would display that label.

- Background
- Getting Started
- Basic layout and summary
- More changes
- Graphs

Next (Background)

Tutorial 2 - Background

WIDGET CO.

The Vice President of Widget Co. is so happy with his **EMPLOYEE VACATION REPORT** that he insists you be the only one to modify it. After much thought he has come up with additional changes that will make his analysis easier and more effective.

1. Only display the **EMPLOYEE VACATION REPORT** header on the first page. Do what you can to maximize page usage. You are instructed not to remove the blue border.
2. Add pie graphs that illustrate vacation buyout value by department to indicate monetary entitlement.
3. Add bar graphs that show the number of vacation days per employee by department.
4. Add a summary bar chart that shows buyout values of employees with the greatest value, indicating the value and department.
5. Calculate average and total: income level, vacation days, and buyout value for all employees.
6. Calculate average and total: income level, vacation days, and buyout value for all employees with a buyout value exceeding $5000.

Previous (Index)   Next (Getting Started)

Tutorial 2 - Getting Started

After going through the documentation, you've come up with the following strategy:

1. Displaying a header on one page can be done with the reprint Table Row Version. Easy! We used the same technique to create alternate row colors in Tutorial 1!
2. Pie graphs should be simple enough. They need to be grouped by department. We will embed them within our department grouping.
3. Bar graphs will be exactly like pie graphs.
4. The summary bar chart needs to be outside department grouping. You choose to put it in the table summary.
5. Averages and totals should be no problem with aggregate keys. These will be placed with the above graph.
6. The last requirement strikes you as tricky to calculate within the report. You realize that you're dealing with a subset of the employees based on a definable condition, but maintaining totals and averages over that subset looks ugly. Can it be done with assignment expressions? Yes, but why not leverage our SQL database?

   You can come up with a single simple query that will return all employees with a buyout value > $5000. The report will see two different DataSets and can easily perform aggregate functions (total, min/max, average) on either. An additional benefit is that if you need to change the requirements you need only change one query.

   ```
   SELECT *, CAST(income/360 * 1.5 * vacationdays AS SIGNED) buyout FROM employees WHERE
   (income/360 * 1.5 * vacationdays) > 5000 ;
   ```

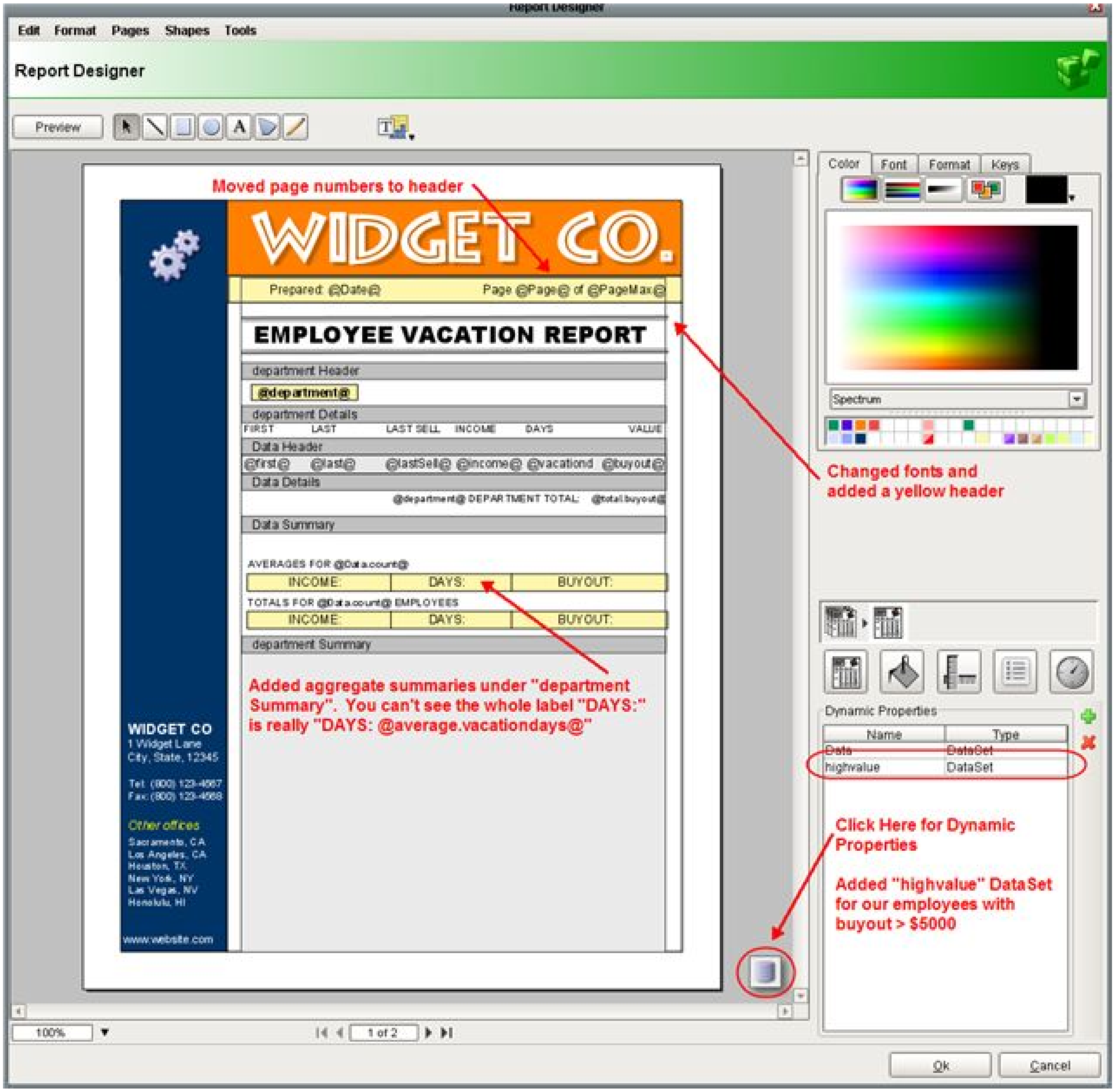
	first	last	department	lastSell	income	withholdings	vacationdays	buyout
2	Julia	Hinderson	ENGINEERING	2005-01-01 00:00:00	121000	10000	19	9579
8	Chris	McDade	ENGINEERING	2006-07-01 00:00:00	52000	4000	31	6717
11	Jefffrey	York	SALES	2006-02-15 00:00:00	69000	1000	18	5175

Index Previous (Background) Next (Basic Layout)

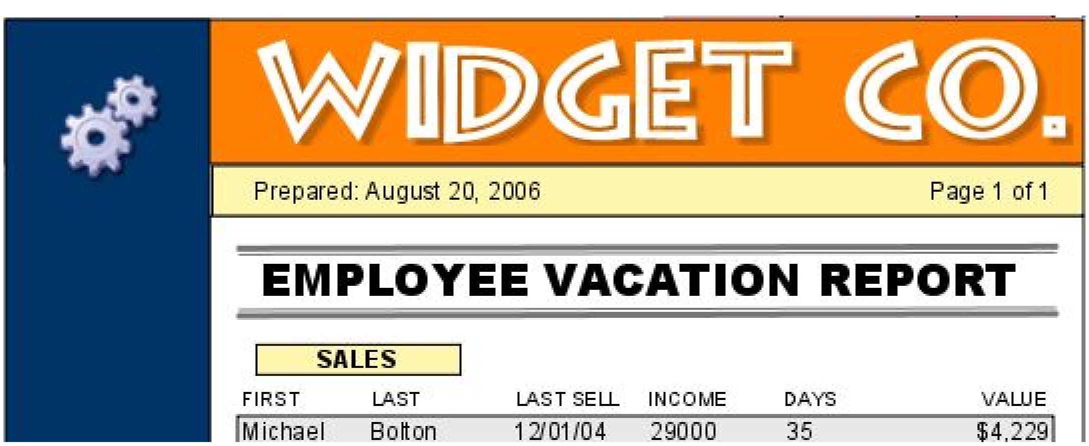
TIP	Get in the habit of utilizing the SQL database. It is easier to manipulate the data *before* the report gets it. This is especially true when you need to do joins or have other complex query requirements.

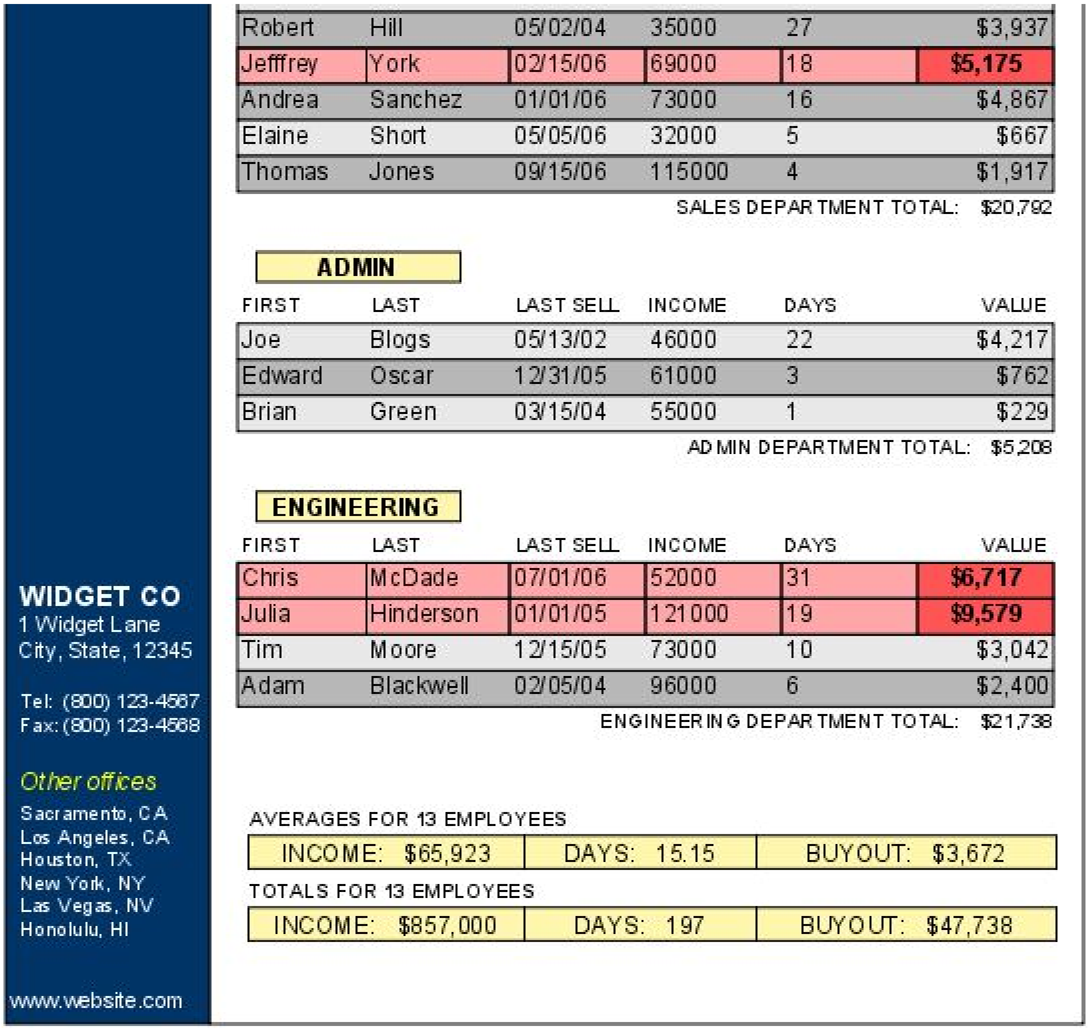
Tutorial 2 - Basic Layout

We're going to make a few minor aesthetic changes to give us room for graphs in the report. We will use both bar and pie graphs to indicate how many vacation days and how much vacation buyout money employees are entitled to. These graphs provide managers with an accurate idea on where they stand at a quick glance.



Almost everything here has been covered in Tutorial #1.





Robert	Hill	05/02/04	35000	27	$3,937
Jefffrey	York	02/15/06	69000	18	**$5,175**
Andrea	Sanchez	01/01/06	73000	16	$4,867
Elaine	Short	05/05/06	32000	5	$667
Thomas	Jones	09/15/06	115000	4	$1,917

SALES DEPARTMENT TOTAL: $20,792

ADMIN

FIRST	LAST	LAST SELL	INCOME	DAYS	VALUE
Joe	Blogs	05/13/02	46000	22	$4,217
Edward	Oscar	12/31/05	61000	3	$762
Brian	Green	03/15/04	55000	1	$229

ADMIN DEPARTMENT TOTAL: $5,208

ENGINEERING

FIRST	LAST	LAST SELL	INCOME	DAYS	VALUE
Chris	McDade	07/01/06	52000	31	**$6,717**
Julia	Hinderson	01/01/05	121000	19	**$9,579**
Tim	Moore	12/15/05	73000	10	$3,042
Adam	Blackwell	02/05/04	96000	6	$2,400

ENGINEERING DEPARTMENT TOTAL: $21,738

AVERAGES FOR 13 EMPLOYEES

INCOME: $65,923	DAYS: 15.15	BUYOUT: $3,672

TOTALS FOR 13 EMPLOYEES

INCOME: $857,000	DAYS: 197	BUYOUT: $47,738

WIDGET CO
1 Widget Lane
City, State, 12345

Tel: (800) 123-4567
Fax: (800) 123-4568

Other offices
Sacramento, CA
Los Angeles, CA
Houston, TX
New York, NY
Las Vegas, NV
Honolulu, HI

www.website.com

1. Change the font of our **EMPLOYEE VACATION REPORT** label and moved it from outside the table into the **department header**.
2. Add **highvalue** Dynamic Property. It needs to be a DataSet. Populate the data within the FactoryPMI designer, under the **highvalue** dynamic property of the *Report Viewer* component just like we did in Tutorial #1. Our new SQL query:

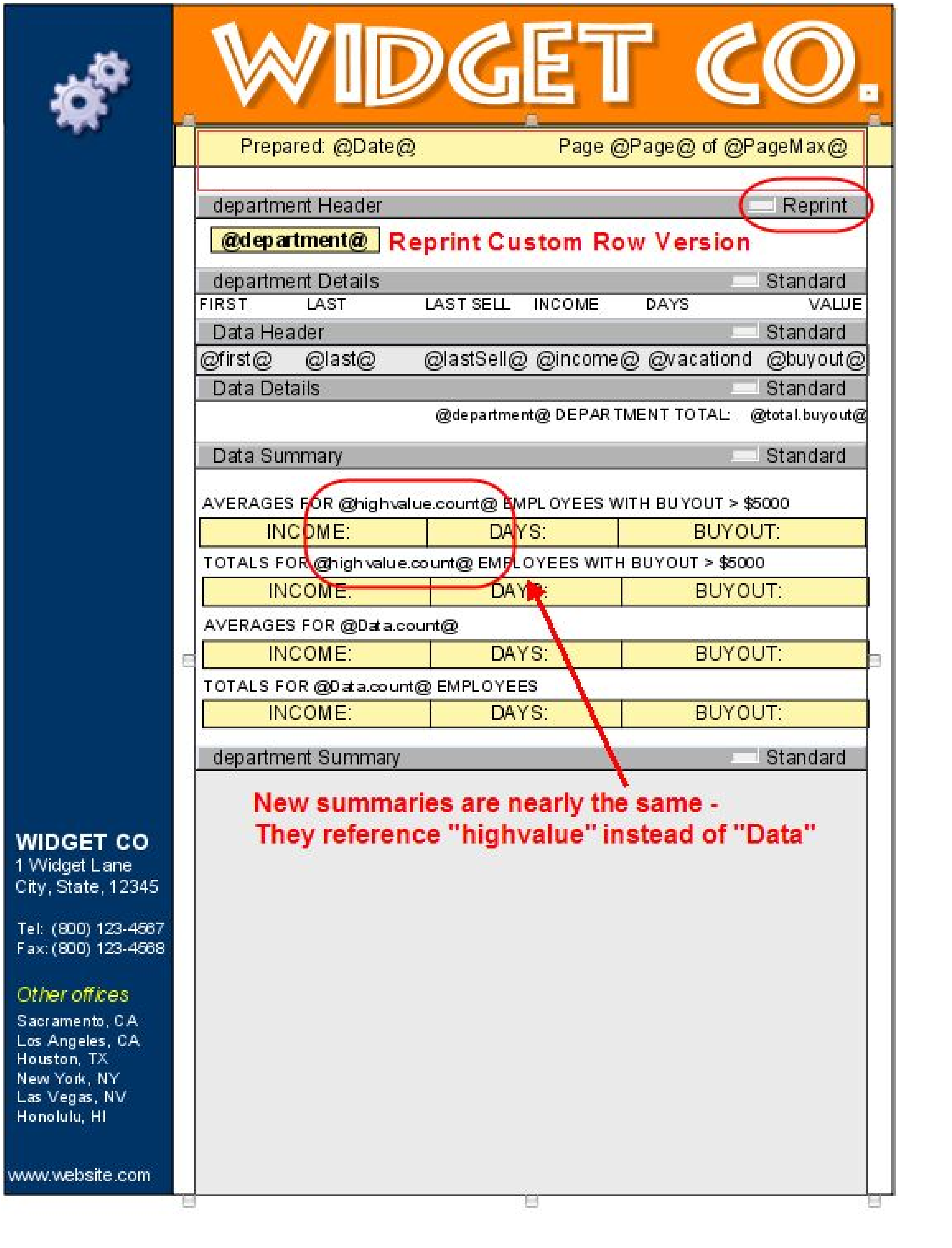
```
SELECT *, CAST(income/360 * 1.5 * vacationdays AS SIGNED) buyout FROM employees WHERE
(income/360 * 1.5 * vacationdays) > 5000;
```

   We are creating a second DataSet that contains the subset of employees whose buyout value exceeds $5000. This will simplify our conditional average and total calculations.

3. Add yellow header rectangle and move page numbers up.
4. Resize table downward to the bottom of the page for more room.
5. Add yellow labels for summary aggregates. We will be using: **count**, **total**, and **average** and placing them under **department summary**.

Tutorial 2 - More changes

We now add a reprint row version to only display **EMPLOYEE VACATION REPORT** on the first page. We will also add summaries for our buyout > $5000 employees.



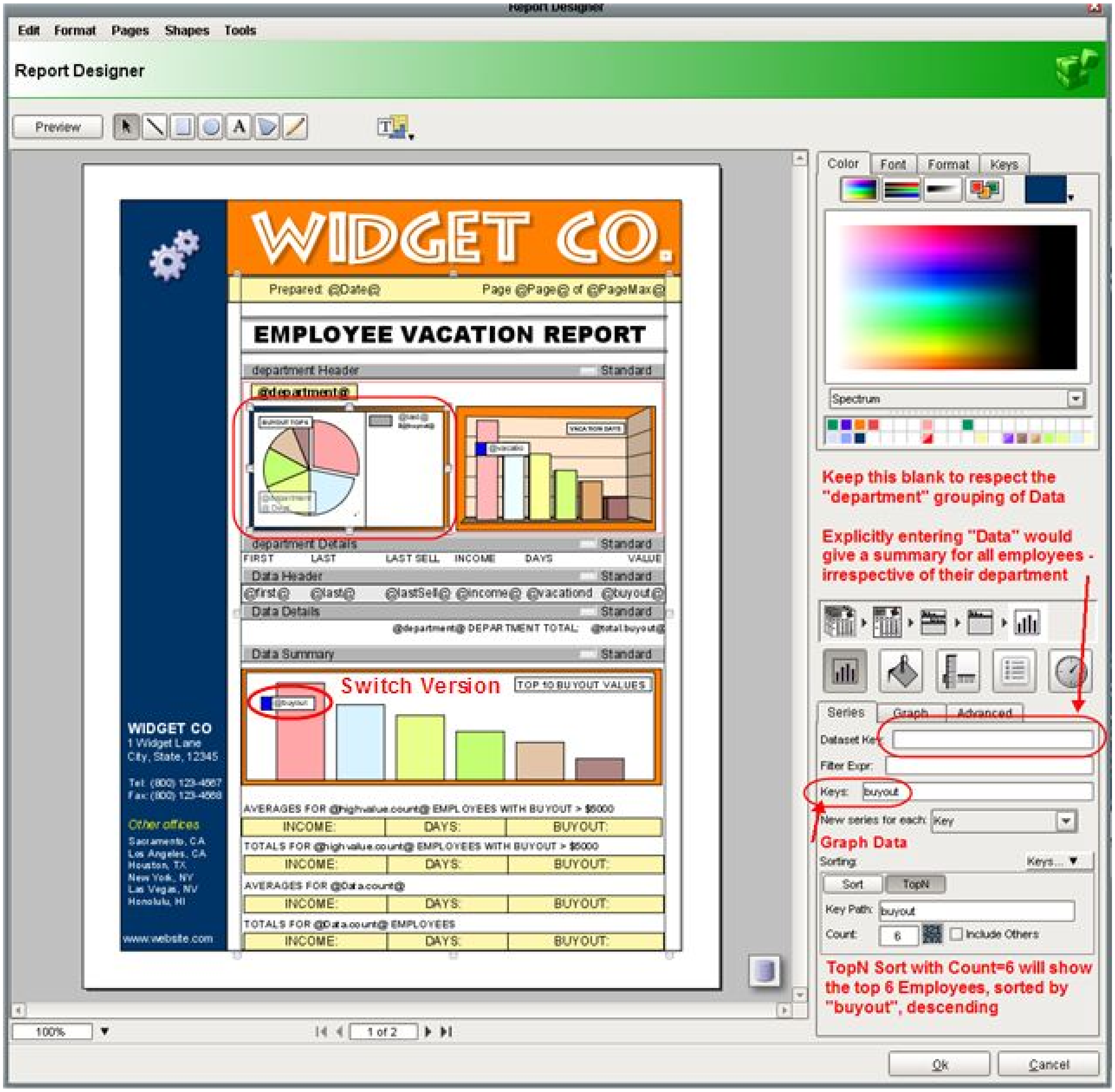
1. Add reprint row version for every other header besides the first page. Customize as necessary.
2. Add **highvalue** summaries. The process is identical to our existing summaries. Replace **Data** with **highvalue**.

Index Previous [Next (Graphs)](Next)

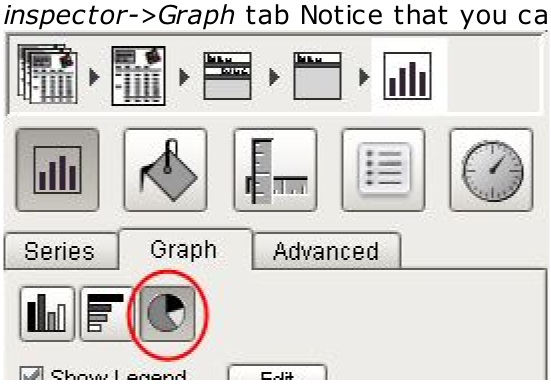
TIP	In this table, **Data** is implied and can be omitted since it is the table's primary DataSet. **highvalue** must be explicitly entered. For example, **@Data.count@** could have been entered **@count@**, while **@highvalue.count** could not have been simplified.

Tutorial 2 - Graphs

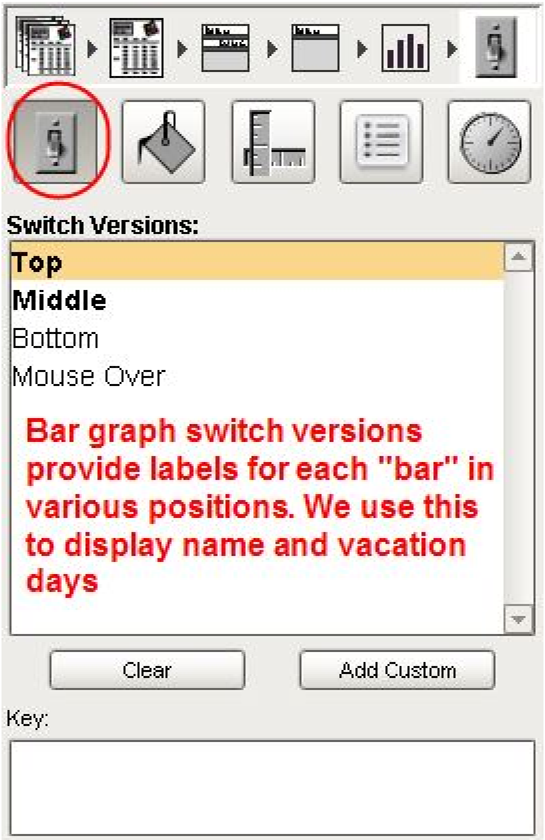
Next, add Graphs to the report!



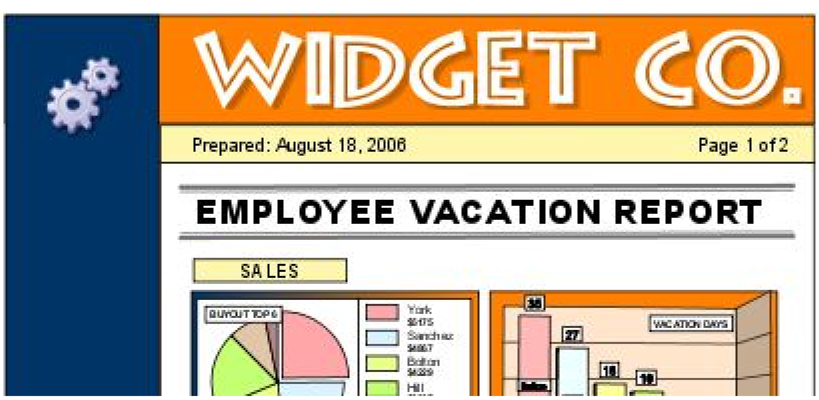
1. Drag 2 graphs down to the **department Header**. The left one will be a buyout pie graph, while the right will be a vacation days bar graph.
2. For both graphs, ensure the **Dataset Key** is blank. Found under *graph->shapespecific inspector->Series* tab. Set **Keys:** to **buyout** and **vacationdays**, respectively.
3. Make one graph a pie graph and the other a bar graph. Look for icons under *graph->shapespecific inspector->Graph* tab Notice that you can set series colors here under **Colors**.

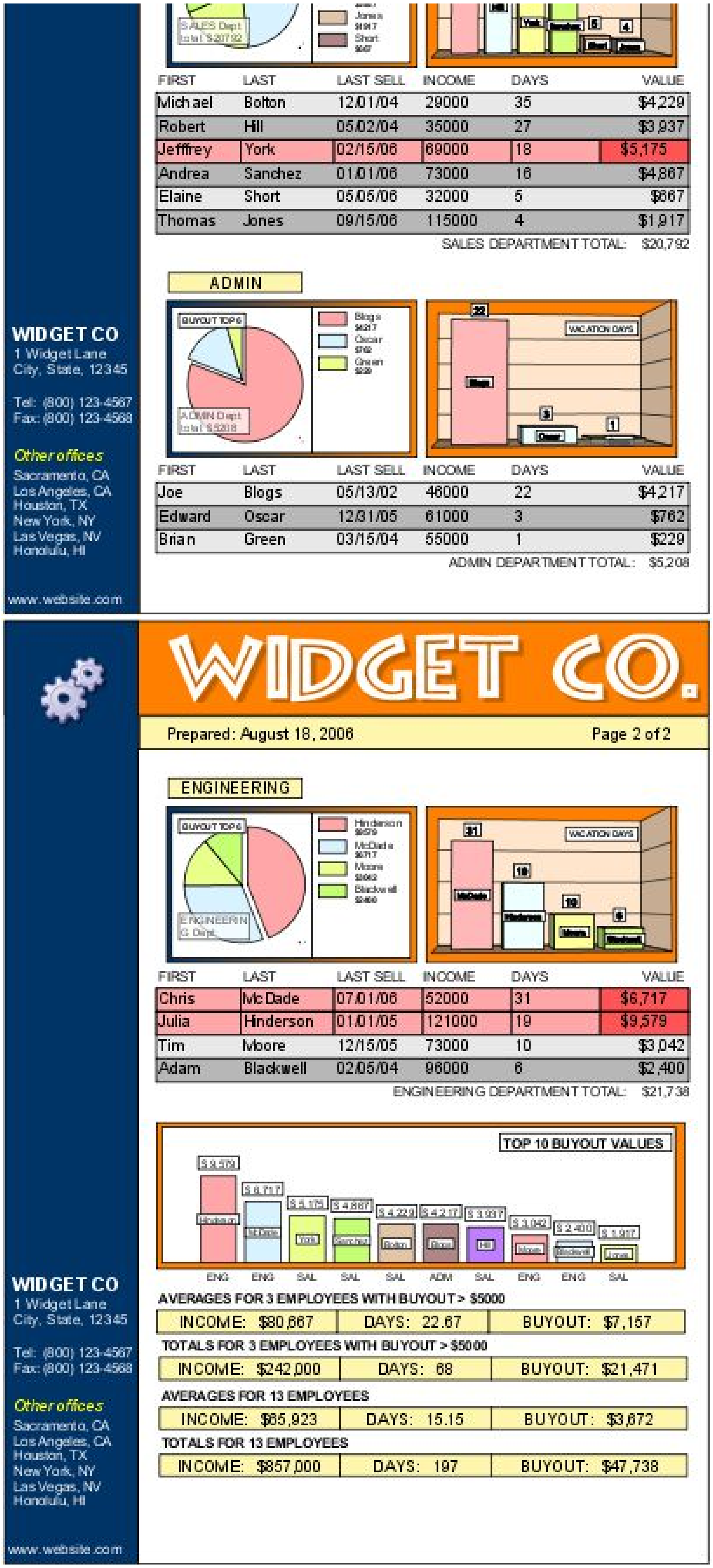


4. Enable switch versions by checking *Show Bar/Wedge Labels*. We will add one on the top and one in the middle. Look at bar chart labels on the final screenshot for an example.



5. Use lots of double clicking to drilling down to select basic shapes and text. Change colors and fonts as desired.
6. Added semi transparent label with department subtotal (**@total.buyout@**) to the pie graph.
7. Added department label for summary bar graph using bottom switch version. **@substring (department,0,3)@** used string functions to display a 3 letter abbreviation.





FIRST	LAST	LAST SELL	INCOME	DAYS	VALUE
Michael	Bolton	12/01/04	29000	35	$4,229
Robert	Hill	05/02/04	35000	27	$3,937
Jeffrey	York	02/15/06	69000	18	$5,175
Andrea	Sanchez	01/01/06	73000	16	$4,867
Elaine	Short	05/05/06	32000	5	$667
Thomas	Jones	09/15/06	115000	4	$1,917

SALES DEPARTMENT TOTAL: $20,792

ADMIN

FIRST	LAST	LAST SELL	INCOME	DAYS	VALUE
Joe	Blogs	05/13/02	46000	22	$4,217
Edward	Oscar	12/31/05	61000	3	$762
Brian	Green	03/15/04	55000	1	$229

ADMIN DEPARTMENT TOTAL: $5,208

WIDGET CO
1 Widget Lane
City, State, 12345

Tel: (800) 123-4567
Fax: (800) 123-4568

Other offices
Sacramento, CA
Los Angeles, CA
Houston, TX
New York, NY
Las Vegas, NV
Honolulu, HI

www.website.com

WIDGET CO.

Prepared: August 18, 2006 Page 2 of 2

ENGINEERING

FIRST	LAST	LAST SELL	INCOME	DAYS	VALUE
Chris	McDade	07/01/06	52000	31	$6,717
Julia	Hinderson	01/01/05	121000	19	$9,579
Tim	Moore	12/15/05	73000	10	$3,042
Adam	Blackwell	02/05/04	96000	6	$2,400

ENGINEERING DEPARTMENT TOTAL: $21,738

TOP 10 BUYOUT VALUES

AVERAGES FOR 3 EMPLOYEES WITH BUYOUT > $5000

INCOME: $80,667	DAYS: 22.67	BUYOUT: $7,157

TOTALS FOR 3 EMPLOYEES WITH BUYOUT > $5000

INCOME: $242,000	DAYS: 68	BUYOUT: $21,471

AVERAGES FOR 13 EMPLOYEES

INCOME: $65,923	DAYS: 15.15	BUYOUT: $3,672

TOTALS FOR 13 EMPLOYEES

INCOME: $857,000	DAYS: 197	BUYOUT: $47,738

WIDGET CO
1 Widget Lane
City, State, 12345

Tel: (800) 123-4567
Fax: (800) 123-4568

Other offices
Sacramento, CA
Los Angeles, CA
Houston, TX
New York, NY
Las Vegas, NV
Honolulu, HI

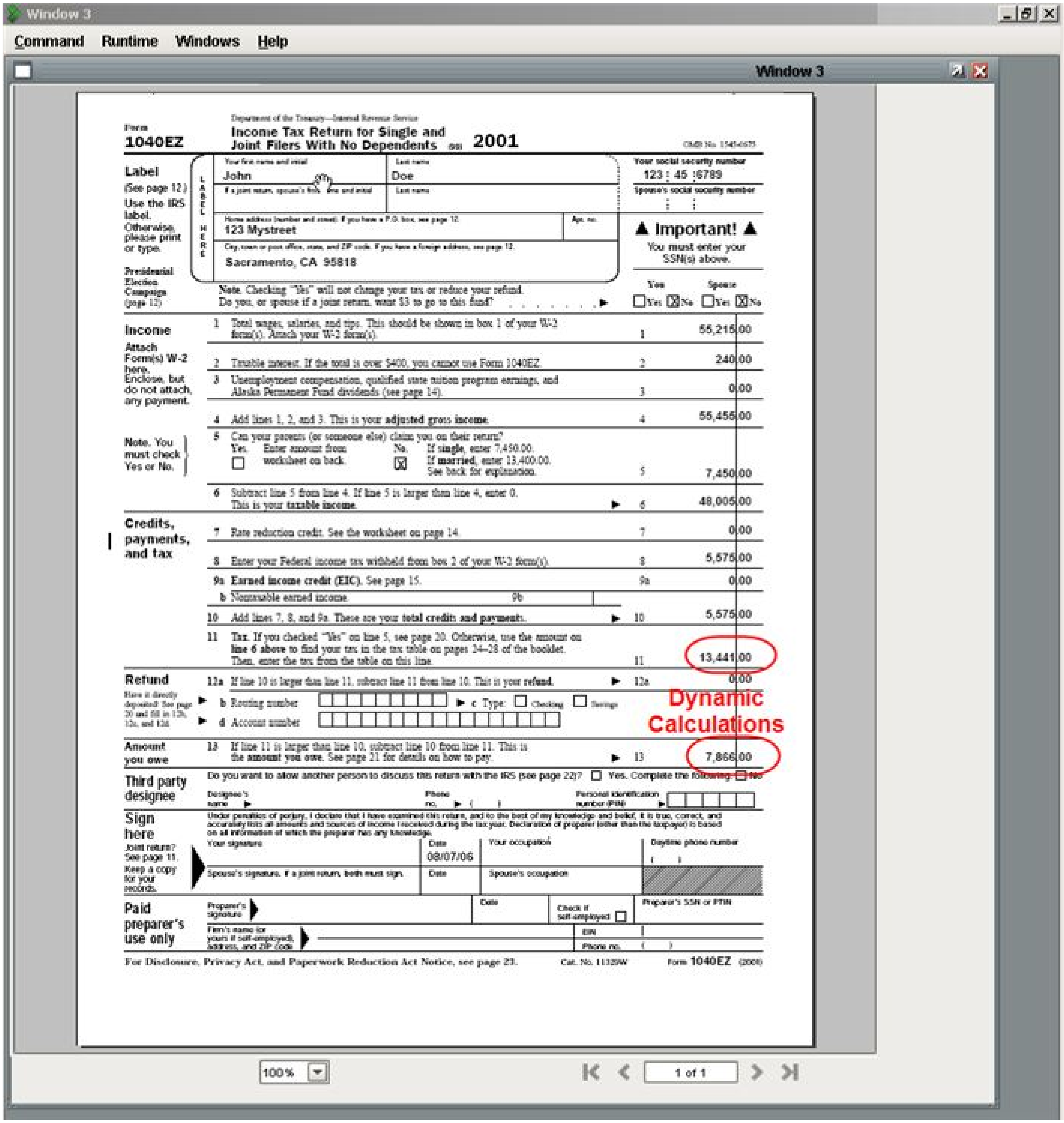
www.website.com

Index Previous (More Changes) On to Tutorial 3

TIP	The toughest part of creating small graphs is labeling the data legibly. This takes a little practice. Don't hesitate to mess up your report playing with options, then click cancel in the customizer window and start over again. You'll get the hang of it in no time!

Tutorial 3 - Overview

Tutorial #3 turns an existing PDF file into a dynamic report



- [Background]
- [Creating our report]

[Next (Background)]

Tutorial 3 - Background

WIDGET CO.

Widget Co. wants to automatically generate 1040EZ forms for its employees taxes.

Here are the requirements:

1. Start with an existing pdf report.
2. Dynamically fill in: name, income, withholdings, dependents, and other details.
3. Dynamically calculate taxes based on an expression.
4. Display a check mark (isVisible condition) based on an expression.
5. Allow users to print report or save as a pdf.

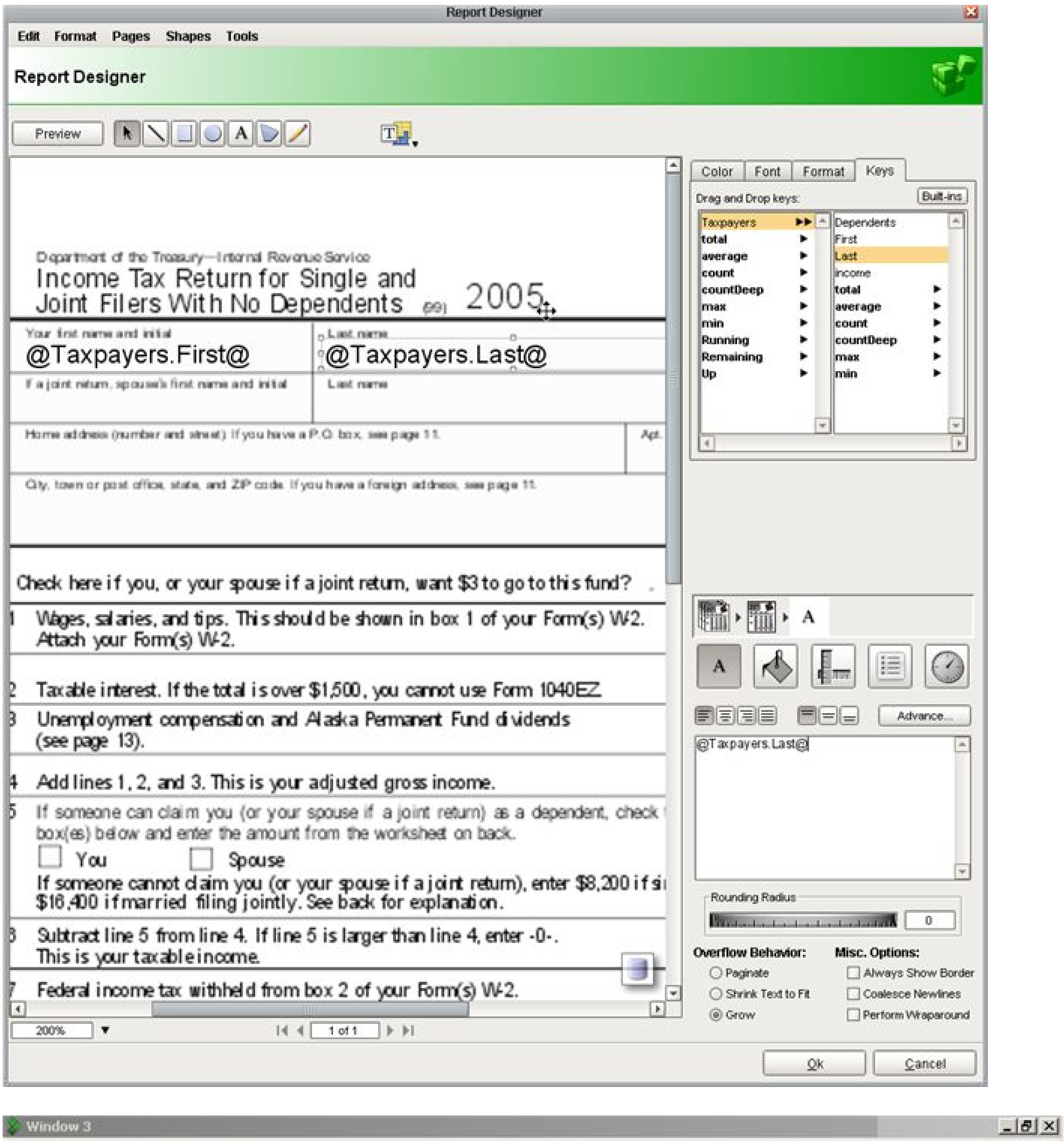
Previous (Index)   Next (Creating Our Report)

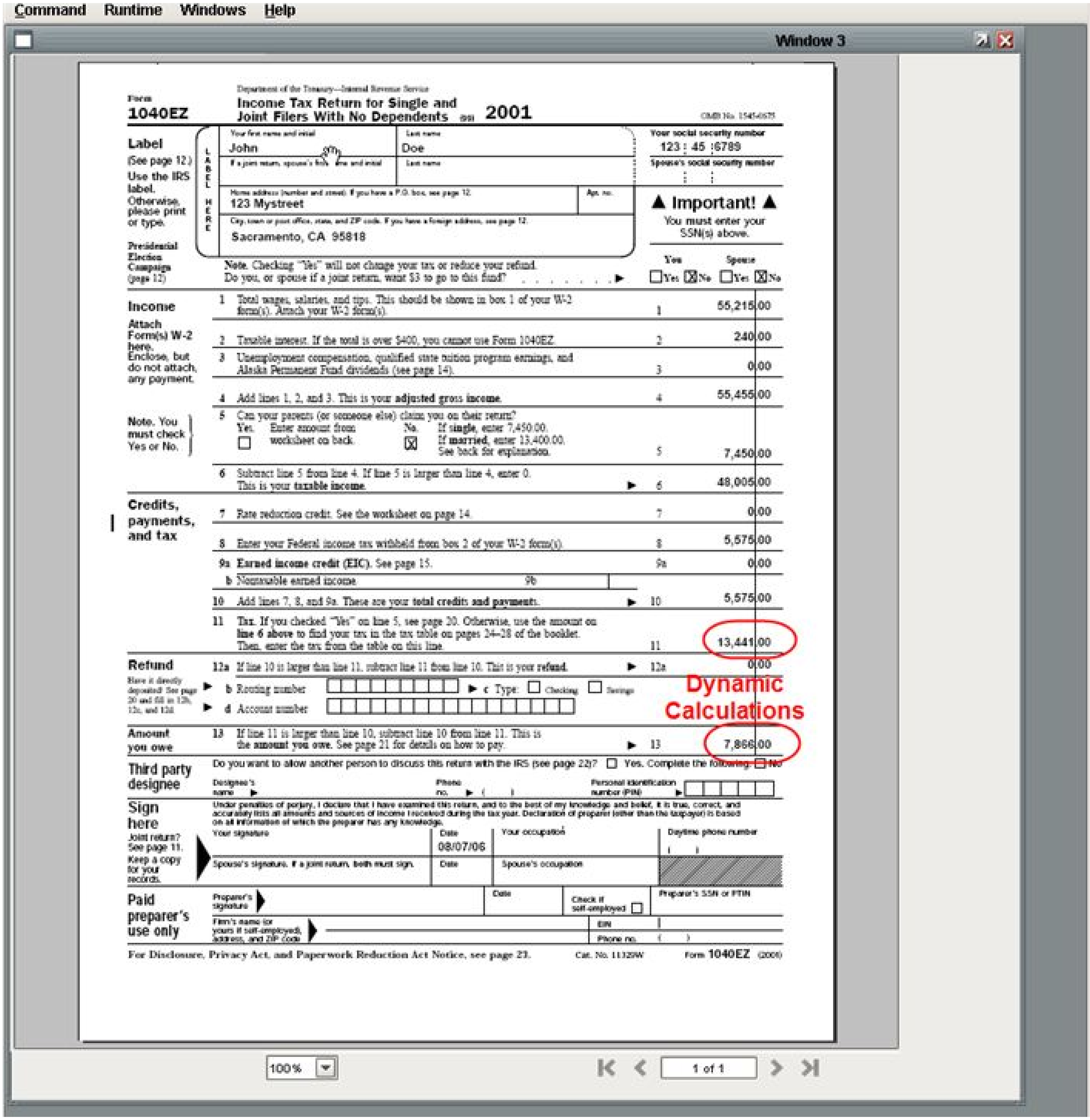
Tutorial 3 - Creating the report

Widget Co. wants to automatically generate 1040EZ forms for its employees taxes.

Here are the requirements:

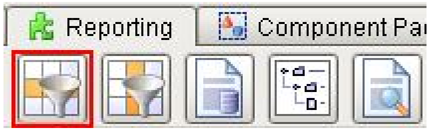
1. Start with an existing pdf report.
2. Drag in keys
3. Users can print report or save as a pdf by right clicking the report.



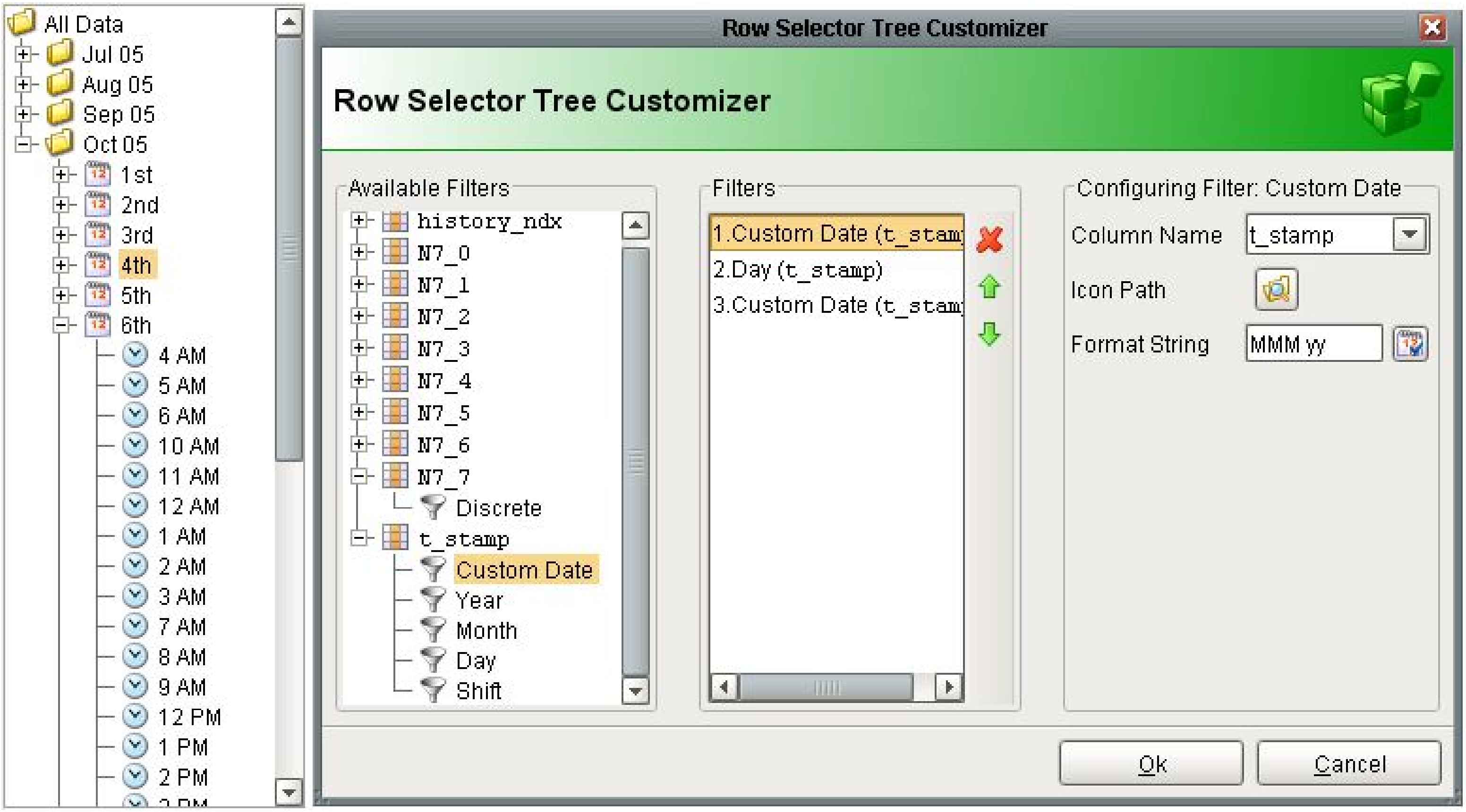


Previous (Index)

Row Selector

Icon in toolbar: 

Description



The selected data will output all data from Oct 4, 2005

The *Row Selector* is a component that allows users to filter a DataSet based on unique values of one or more columns. Each level in the sorting tree is based on these properties.

The user will see a dynamically generated expandable tree that groups their data by any number of choices. As they click down the tree, objects bound to the DataSet will indicate the filtered data. Here are a few examples.

- A line graph bound to a *Row Selector*. Set up grouping to be first by month and year, then day, then hour, like the top left illustration. Clicking on a month and year will dynamically update the graph for that time period. Further clicking to a specific day or hour will re-filter the graph for that period.
- A Report Viewer bound to a *Row Selector*. Grouping by department (String) would allow selection by department, automatically regenerating the Report on selection.
- An "alarm history" table bound to a *Row Selector*. This could first be broken down severity level (Integer), then broken into "Alarm Acknowledged" / "Not Acknowledged" (Boolean based). Clicking "Severity 3" would filter the table to all Severity 3 alarms. Selecting "Unacknowledged" would then filter the table to unacknowledged alarms of severity 3.

Properties

Show All Data Node	`showAllDataNode`	BOOLEAN

Displays or hides the 'All Data' (root) node.

Show Root Handles	`showRootHandles`	BOOLEAN

If true, root node(s) will have collapsible handles like child nodes.

Show Node Size	showNodeSize	BOOLEAN

If true, the number of nodes in each row will be shown.

Properties Loading	propertiesLoading	INTEGER

Indicates number of dataSets loading. This is strictly a bindable property. It can be used as status indication to the user that data is loading.

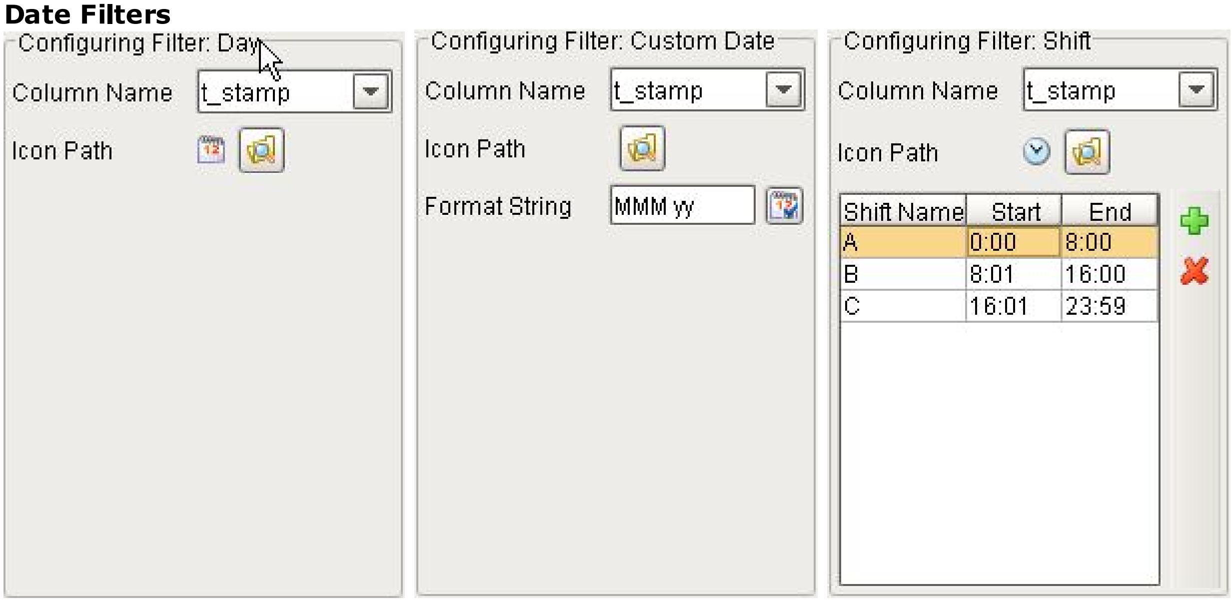
Customizer

The *Row Selector* customizer defines *Filters* that allow each level of user data filtering. Browse through the tree of **Available Filters**, then drag the desired filter to the *filter* pane. Different options will be available under **Configure Filter**: *FilterType* based on the filter type.

Common Filter Properties

Property	Function
Column Name	Allows selection of date column
Icon Path	Click to choose a graphic for each node.

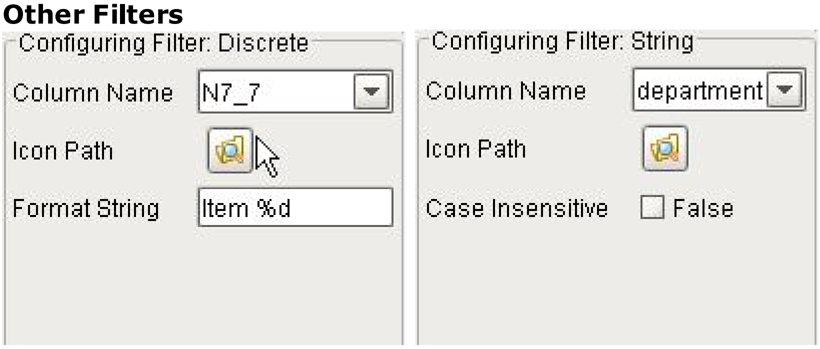
Date Filters



Different options for time columns

The **Day** (*Date*) filter separates rows by day.
The **Custom Date** (*Date*) filter uses pattern masks in the **Format String** for a flexible date criteria definition.
The **Shift** (*Date*) filter breaks up data into shifts, which are named defined time ranges.

Other Filters



The **Discreet** (*Integer*) filter breaks rows down by unique integer. **Format String** allows you to define the text string that the user sees.
The **String** (*String*) filter breaks rows down by unique string. **Case Insensitive** defines case sensitivity.

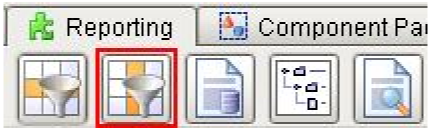
Events

- **mouse**
  - mouseClicked
  - mouseEntered
  - mouseExited
  - mousePressed
  - mouseReleased
- **mouseMotion**
  - mouseDragged
  - mouseMoved
- **propertyChange**
  - propertyChange

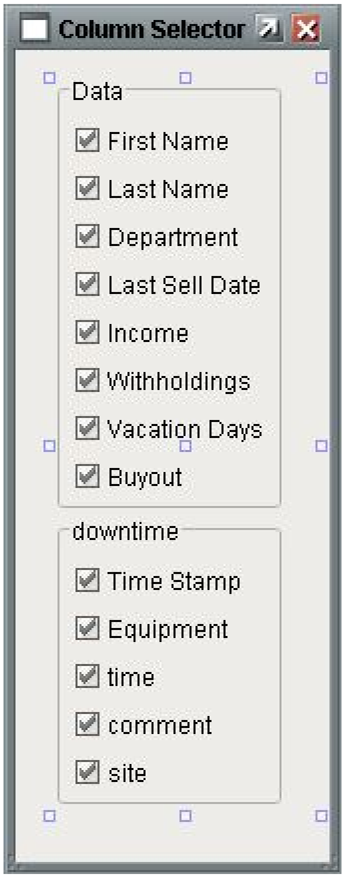
Scripting Functions

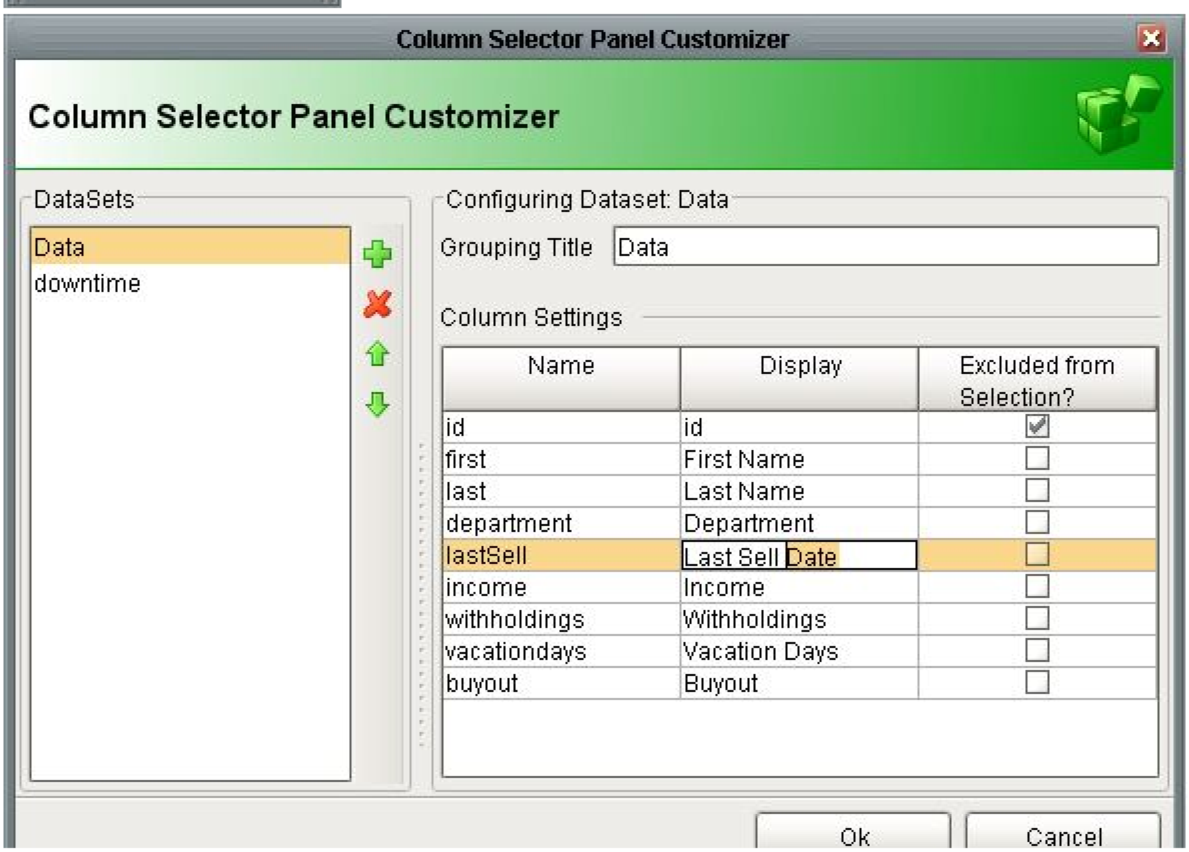
TIP	The *Row Selector* works well with the: Report Viewer, Graph, and Table components!

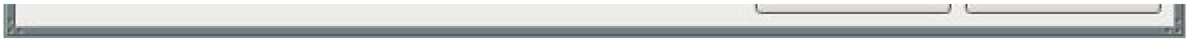
Column Selector

Icon in toolbar: 

Description







The *Column Selector* is a component that takes DataSets in, allows users to show or hide variables in the DataSets (Columns) via checkboxes, then outputs the resulting DataSet.

The *Column Selector* allows users to choose which columns in a DataSet that they wish to use. If an object is bound to the *Column Selector* it will update itself whenever a user checks or unchecks a column. This allows users to dynamically show/hide: Table columns, "pens" on a graph, data in a Report Viewer, or any other component set up to use a DataSet.

Properties

Group by DataSet	`grouping`	BOOLEAN

Displays each DataSet's columns in a separate bordered container. Applicable to multiple DataSets only.

Alphabetize	`alphabetize`	BOOLEAN

Orders columns alphabetically as opposed to their native order in the DataSet .

Normalize Widths	`normalizeWidths`	BOOLEAN

If true, all checkboxes will be assigned the same width, which causes them to line up in columns

Horizontal Gap	`hGap`	INTEGER

The horizontal gap, in pixels, between checkboxes and grouping panels.

Vertical Gap	`vGap`	INTEGER

The vertical gap, in pixels, between checkboxes and grouping panels.

Customizer

The *Column Selector* customizer is very straightforward. The left pane allows you to add and remove DataSets. Selecting a DataSet will display a list of columns in the table in the right pane. Under *Display* you may modify the name that users see. *Excluded from Selection* will remove the given column from the users list of choices.

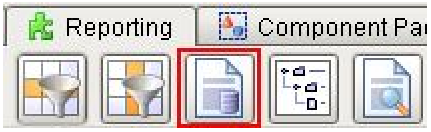
Events

- **mouse**
  - `mouseClicked`
  - `mouseEntered`
  - `mouseExited`
  - `mousePressed`
  - `mouseReleased`
- **mouseMotion**
  - `mouseDragged`
  - `mouseMoved`
- **propertyChange**
  - `propertyChange`

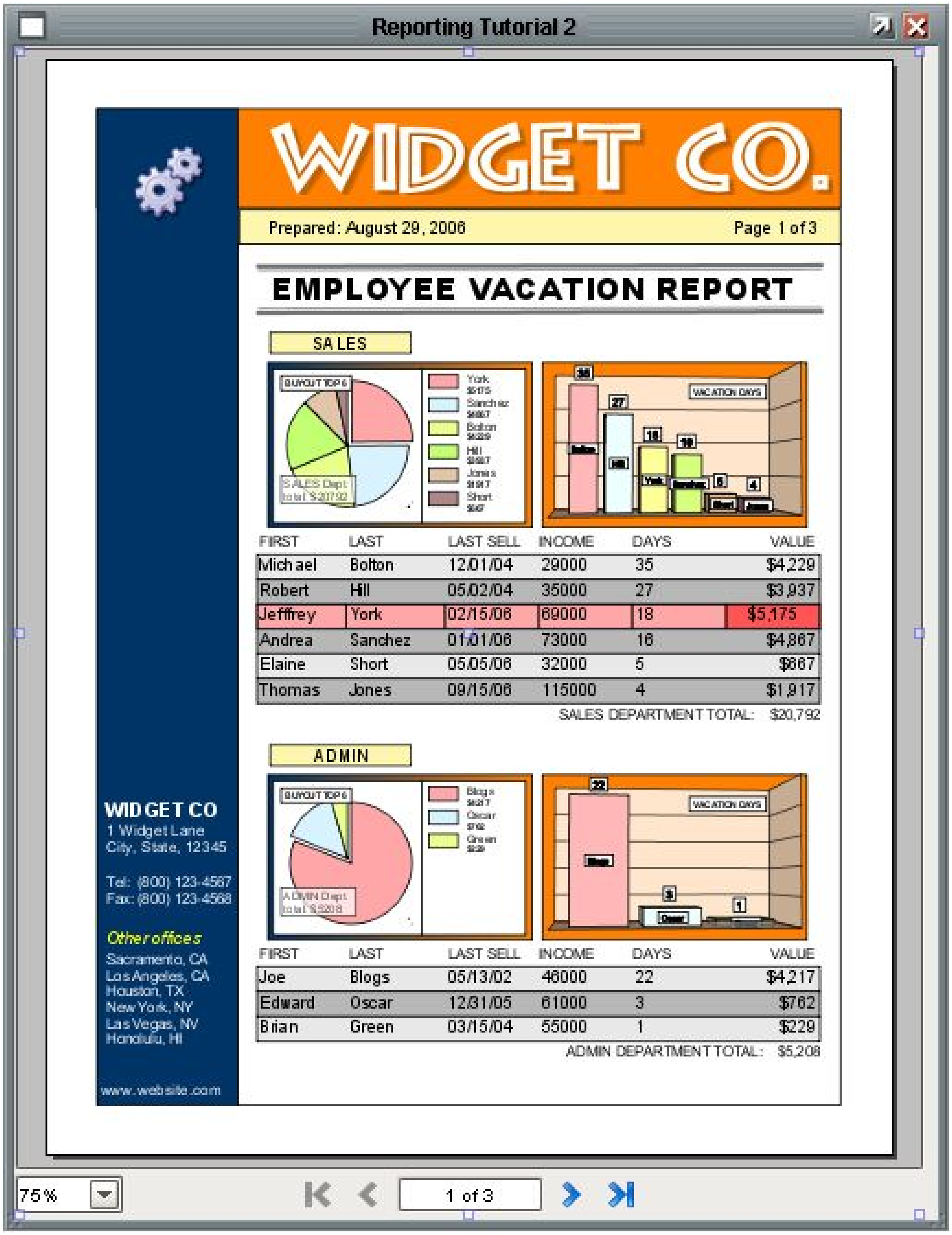
Scripting Functions

TIP	The *Column Selector* works well with the FactoryPMI Graph and Table components!

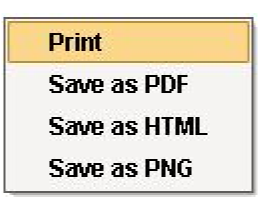
Report Viewer

Icon in toolbar: 

Description



The *Report Viewer* is the component that displays reports within FactoryPMI. Dynamic Properties bring data from FactoryPMI into the report. Any changes to the dynamic data automatically regenerates the report. Customization is done in the Report Designer via the customizer (Cntl+U)



Users can zoom in to the report and scroll between pages with the builtin controls located at the bottom. Right clicking anywhere on a report in the *Report Viewer* in the Runtime will allow you to print or save the report in several formats.

Properties

Zoom Factor	zoomFactor	INT

This variable sets and displays the current zoom level of the report.

Customizer

The customizer for this class is the Report Designer. It lets you add, remove, and edit properties for the Report's datasets as well as create entire reports.

Events

- **mouse**
  - mouseClicked
  - mouseEntered
  - mouseExited
  - mousePressed
  - mouseReleased
- **mouseMotion**
  - mouseDragged
  - mouseMoved
- **propertyChange**
  - propertyChange

Scripting Functions

print([printerName], [showDialog])

Prompts the report to print. The optional arguments can be used to specify the name of the printer to use, and whether or not to show the user the print options dialog box.

Parameters

[printerName]

The name of the printer to print to. Omit or use `None` to use the default printer.

[showDialog]

A boolean (0 or 1) indicating whether or not to show the user the print dialog options box.

Example:

```
# This would prompt the user to print, showing them the print dialog box and starting with the deafult printer selected
report = event.source.parent.getComponent("Report Viewer")
report.print()
```

Example:

```
# This would print to the "HP Laserjet" printer with no user interaction
report = event.source.parent.getComponent("Report Viewer")
report.print("HP Laserjet", 0)
```

Example:

```
# This would print to the default printer with no user interaction
report = event.source.parent.getComponent("Report Viewer")
report.print(None, 0)
```

getBytesHTML(continuous)

Creates an HTML byte array of the report generated.

Parameters

continuous

Create a paged HTML document or a continuous HTML document

Example:

```
# This code would prompt the user to save the HTML bytes to a file

path = fpmi.file.saveFile("myfile.html")
if path != None:
	fpmi.file.writeFile(path, report.getBytesHTML(1))
```

getBytesPDF()

Creates an HTML byte array of the report generated.
Example:

```
# This code would prompt the user to save the PDF bytes to a file

path = fpmi.file.saveFile("myfile.pdf")
if path != None:
	fpmi.file.writeFile(path, report.getBytesPDF())
```

saveAsHTML(filename, continuous)

Saves the generated report as HTML to the specified filename.

Parameters

filename

The filename, such as myfile.html

continuous

Create a paged HTML document or a continuous HTML document

saveAsPDF(filename)

Saves the generated report as a PDF to the specified filename.

Parameters

filename

The filename, such as myfile.pdf

saveAsPNG(filename)

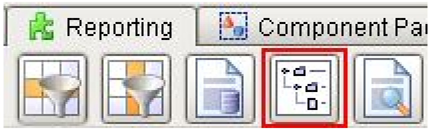
Saves the generated report as a PNG to the specified filename.

Parameters

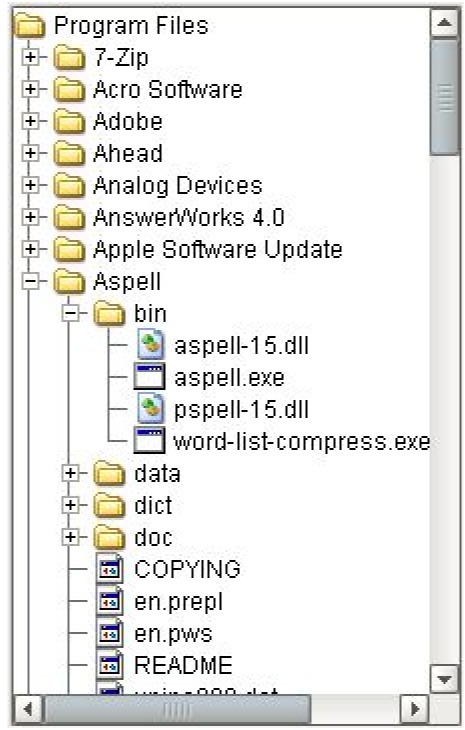
filename

The filename, such as myfile.png

File Explorer

Icon in toolbar: 

Description

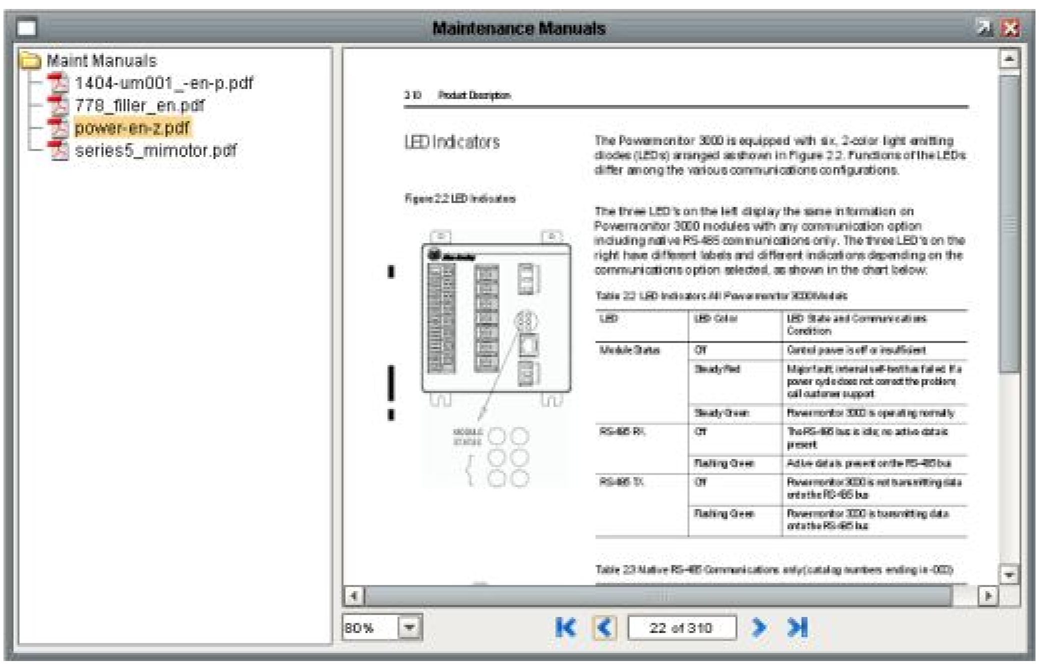


The File Explorer rooted at "C:\Program Files"

The File Explorer component displays a filesystem tree to the user. It can be rooted at any folder, even network folders. It can also filter the types of files that are displayed by their file extension (For example, "**pdf**"). The path to the file that the user selects in the tree is exposed in the bindable property **Selected Path**.

This component is typically used in conjuction with the PDF Viewer component, in order to create a PDF viewing window. This is very useful for viewing things like maintenance manuals from within your project. To create a window like the one shown below follow these steps:

1. Bind the PDF Viewer's **Filename** property to the File Explorer's **Selected Path** property
2. Set the File Explorer's **File extension filter** to "pdf"
3. Set the File Explorer's **Root Directory** to a network folder that has your maintenance manuals in it. (Use a network folder so that all clients will be able to access the manuals).



The File Explorer used with the PDF Viewer for manual viewing.

Properties

Selected Path	selectedPath	STRING

This Read-Only property provides the path to the selected file or folder.

Selected Path Is File	selectedPathIsFile	BOOLEAN

This Read-Only property is true when the selected path is a file, and false otherwise (i.e., the selected path is a folder).

File extension filter	`fileFilter`	STRING

A semicolon separated list of file extensions to display, such as "**pdf**" or "**html;htm;txt;rtf**". Leave blank to show all file types.

Root Directory	`rootDir`	STRING

The path to the root folder to display. Examples: "**C:\Program Files**" or "**\\fileserver\manuals\Maint Manuals**". If blank, the local system's filesystem root is used.

Customizer

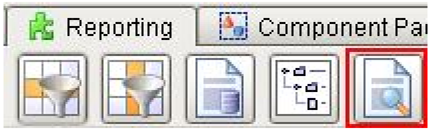
None.

Events

- **mouse**
  - `mouseClicked`
  - `mouseEntered`
  - `mouseExited`
  - `mousePressed`
  - `mouseReleased`
- **mouseMotion**
  - `mouseDragged`
  - `mouseMoved`
- **propertyChange**
  - `propertyChange`

Scripting Functions

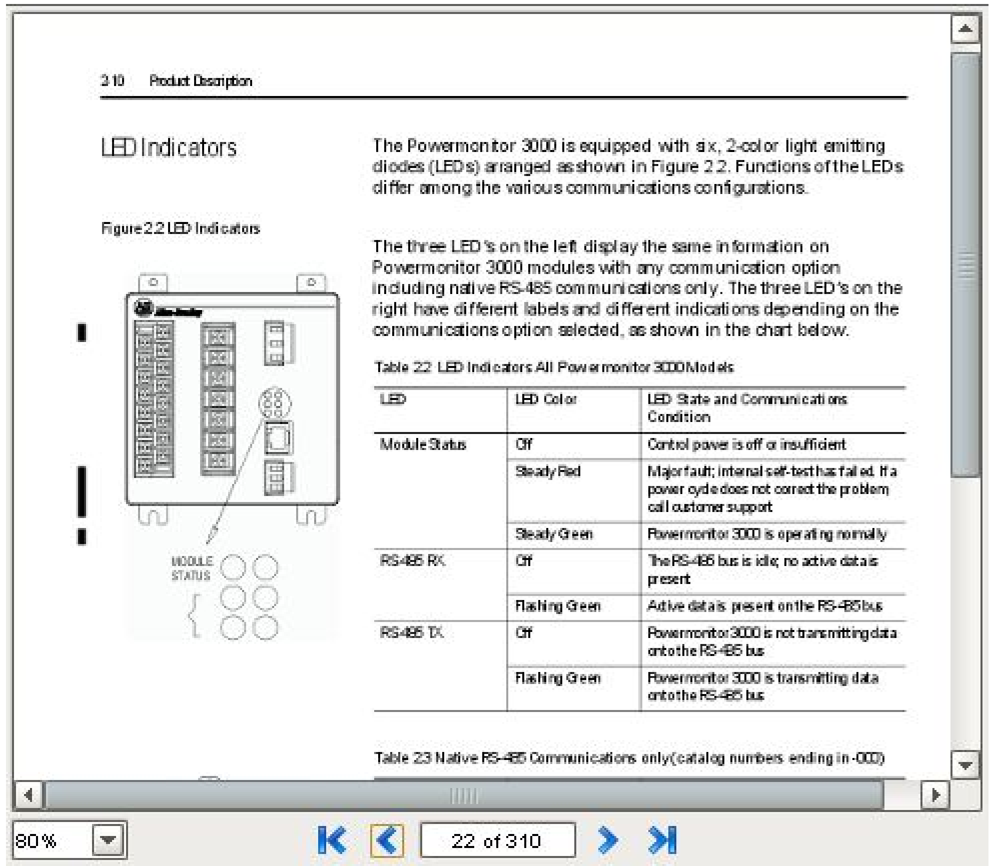
PDF Viewer

Icon in toolbar: 

Description

The PDF Viewer component displays a PDF that exists as a file in some accessable filesystem, or as a URL. Note that this component is simply for viewing existing PDFs - for creating dynamic reports, use the [Report Viewer](#) component.

This component is typically used in conjuction with the [File Explorer](#) component, in order to create a PDF viewing window. See the File Explorer's documentation for instructions on how to put these two components together.



The PDF Viewer viewing a maintenance manual.

Properties

Filename	filename	STRING

The path or URL to the PDF file to display. Examples: "**C:\PDFFiles\example.pdf**", "**\\fileserver\manuals\valve-2.pdf**", or "**http:\\www.example.com\test.pdf**".

Zoom Factor	zoomFactor	FLOAT

The zoom factor of the viewer. 1=100%.

Customizer

None.

Events

- [**mouse**](#)
  - mouseClicked
  - mouseEntered
  - mouseExited
  - mousePressed
  - mouseReleased

- **mouseMotion**
  - mouseDragged
  - mouseMoved
- **propertyChange**
  - propertyChange

Scripting Functions

setBytes(bytes)

Sets the PDF document to a byte array. Useful for loading a PDF document from a SQL database.

Parameters

bytes

The PDF byte array to display.

Example:

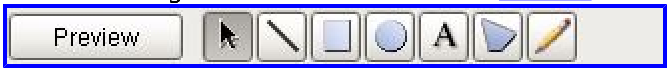
```
# This code would prompt the user to choose a pdf file. If the user chooses a file,
# it would then read that file into a byte array and call setBytes.

path = fpmi.file.openFile('pdf')

if path != None:
		bytes = fpmi.file.readFileAsBytes(path)
		pdfViewer.setBytes(bytes)
```

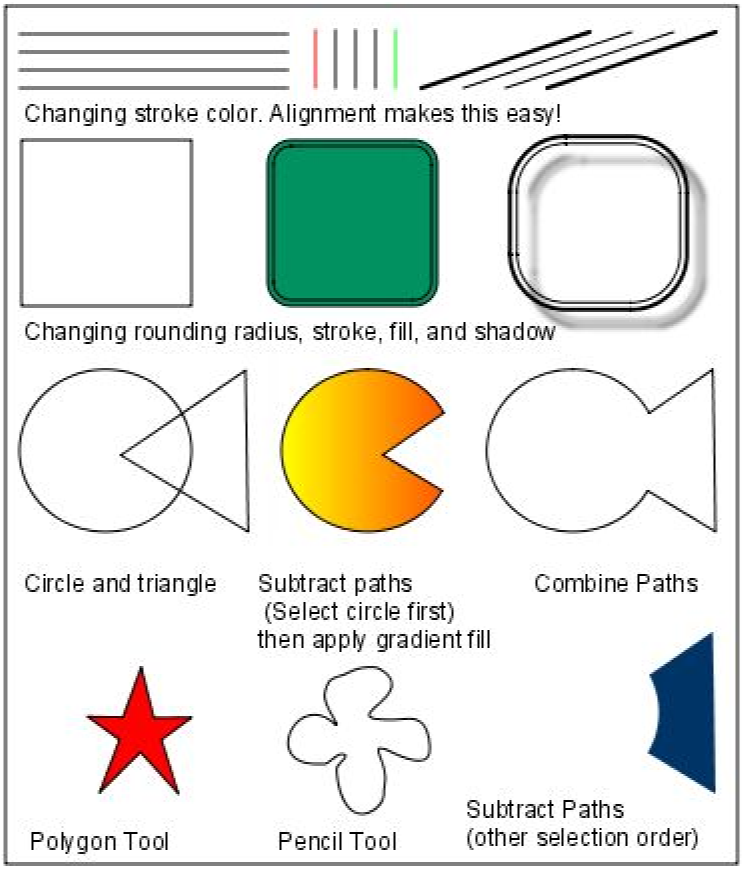
Example 2:

```
# This would get the PDF bytes from a SQL database
bytes = fpmi.db.runScalarQuery("SELECT PDFBlob FROM PDFTable WHERE ID = 1")
pdfViewer.setBytes(bytes)
```

Drawing Tools

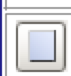
Basic drawing tools are found on the toolbar





Examples using the drawing tools

Drawing tools

Icon	Name	Description
Preview	**Toggle Preview/Edit Mode**	Toggles between Preview and Edit modes. This is equivalent to going between Preview and Design mode in the FactoryPMI designer. **Edit mode** will allow you to make changes to the layout of the report. **Preview mode** will populate the report with data and show you what it will look like in the runtime.
	**Selection Tool**	Default tool. Clicking on objects with the selection tool will select them for movement or modification.
	**Line Tool**	Click and drag to create a line.
	**Rect Tool**	Click and drag to create a rectangle. The Rect inspector will allow you to set rounding radius.
	**Oval Tool**	Click and drag to create an oval. The oval inspector will allow you to select sweep and start angle.
	**Text Tool**	Click and drag to create text. Click for more on text editing.
	**Polygon Tool**	The polygon tool lets you click points that will be joined with straight lines. Alternatively, you can click-drag-release to position line segments interactively. If you hold down the alt key while adding points the polygon tool will behave like pencil for added segments.

		Editing stops under the following conditions: clicking the same point twice, clicking close to the start point or clicking a new tool in the tool bar (like the selection tool) .
	**Pencil Tool**	The pencil tool lets you click and draw free-hand path segments, automatically smoothing the curve on mouse up. If you hold down the alt key, it will behave like polygon for added segments. Editing stops under the same conditions as polygon.

Shapes Menu

This shapes menu allows you to modify the layout of objects in a report

Menu Item	Function
Group/Ungroup	Allows you to merge the currently selected shapes into a single shape for convenient management. Contained shapes are still accessible, via double-click super-select. Ungroup separates grouped shapes.
Bring to Front/Send to Back	All shapes have an order on the page that determines what is drawn on top when two shapes overlap. These options allow you to alter that order.
Make Row Top/Center/Bottom	Quickly align several shapes in a row, either by their top, center, or bottom border. Useful when shapes are of different heights.
Make Column Left/Center/Right	Same as above, but for columns.
Make Same Size, Width, Height	Make several shapes the same width, height or both.
Equally Space Row/Column	Equalizes the distance between shapes horizontally or vertically.
Group in Switch/3D Shape	**This feature groups selected shapes in a Switch Shape, which has the same features as Table Row Versions. It's a powerful way to conditionally provide a different look for a specific element.**
Move to new layer	Creates a new page layer with the currently selected shapes.
Combine/Subtract Paths	Takes multiple overlapping shapes (such as a rectangle and an oval) and combines them into a single shape using the combined paths. A powerful tool to construct complex shapes.
Convert Into Image	Converts the selected shape into an image. Be sure to group shapes first if you want to convert multiple shapes into a single image.

TIP 	The Drawing Tools are really intuitive. Try them out. You'll be an expert in no time.

Crosstab

The *CrossTab* is a DataSet element like the table and graph. It shows a summaries of cross sections of data. To be useful, crosstab data should have the following:

- Lots of repetitious data. You should be looking for sums, averages, or other aggregate functions
- At least 2 (key) columns whose data are repetitious compared to the number of rows. Your data should look "rectangular". For example, If there is only one row for each combination of values of the 2 keys, you will get a trivial crosstab.
- You will typically have a third column that is a number to perform an operation on. Examples are: summing money, displaying average response times, counting occurrences, etc.

The CrossTab template is much simpler than the table template. By default it just shows one cell of a simple table. This is usually configured with an aggregate key, like "@total.getAmount@". After that, grouping keys are dragged to the horizontal and vertical axis.

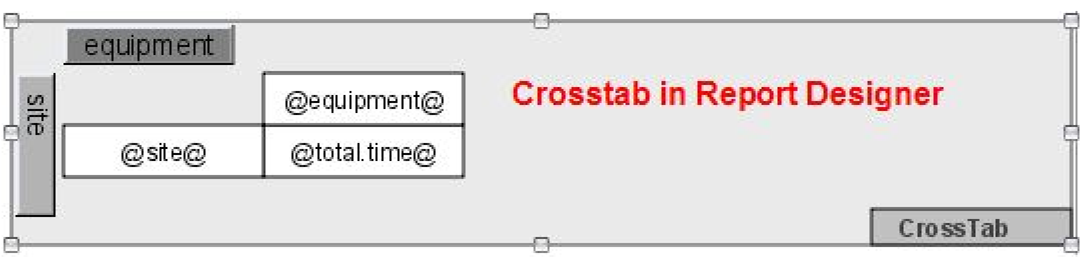
Example

We will use a crosstab to illustrate total downtime by equipment and location.

Employee data can be retrieved from the accounting database with the following SQL query:

```
SELECT * FROM downtime;
```

	t_stamp	equipment	time	comment	site
1	2005-12-20 17:55:00	labeler	50	Out of labels	North site
2	2005-12-22 11:55:00	filler	15	Scheduled maintenance	North site
3	2006-01-02 22:55:00	palletizer	10	Misalignment	East site
4	2006-01-03 02:55:00	conveyor line	25	backup	North site
5	2006-02-12 06:13:00	labeler	10	Scheduled maintenance	North site
6	2006-02-12 12:01:00	labeler	3	Out of labels	East site
7	2006-02-12 14:01:00	palletizer	17	Misalignment	North site
8	2006-02-12 16:23:00	conveyor line	23	Scheduled maintenance	North site
9	2006-02-12 20:04:00	filler	33	Overflow	East site
10	2006-02-12 20:13:00	labeler	21	Stuck labels	North site
11	2006-02-12 20:25:00	filler	20	Overflow	North site
12	2006-02-12 20:36:00	conveyor line	30	Scheduled maintenance	North site



Result	conveyor line	filler	labeler	palletizer
East site	<NA>	33	3	10
North site	78	35	81	17

Notice that the example only has 2 unique sites. This is because we only have 12 rows of data.

Graph

The *Graph* is a DataSet element like the [table](). It shows a 2D or 3D graphical representation of data in the form of bar graph or pie graph. Graphs are useful for illustrating relative amounts of summarized data.

Populating data including the concepts of data keys, sorting, and filtering are nearly identical to that of a table. The look of the graph has a much deeper [superselection]() model than a table.

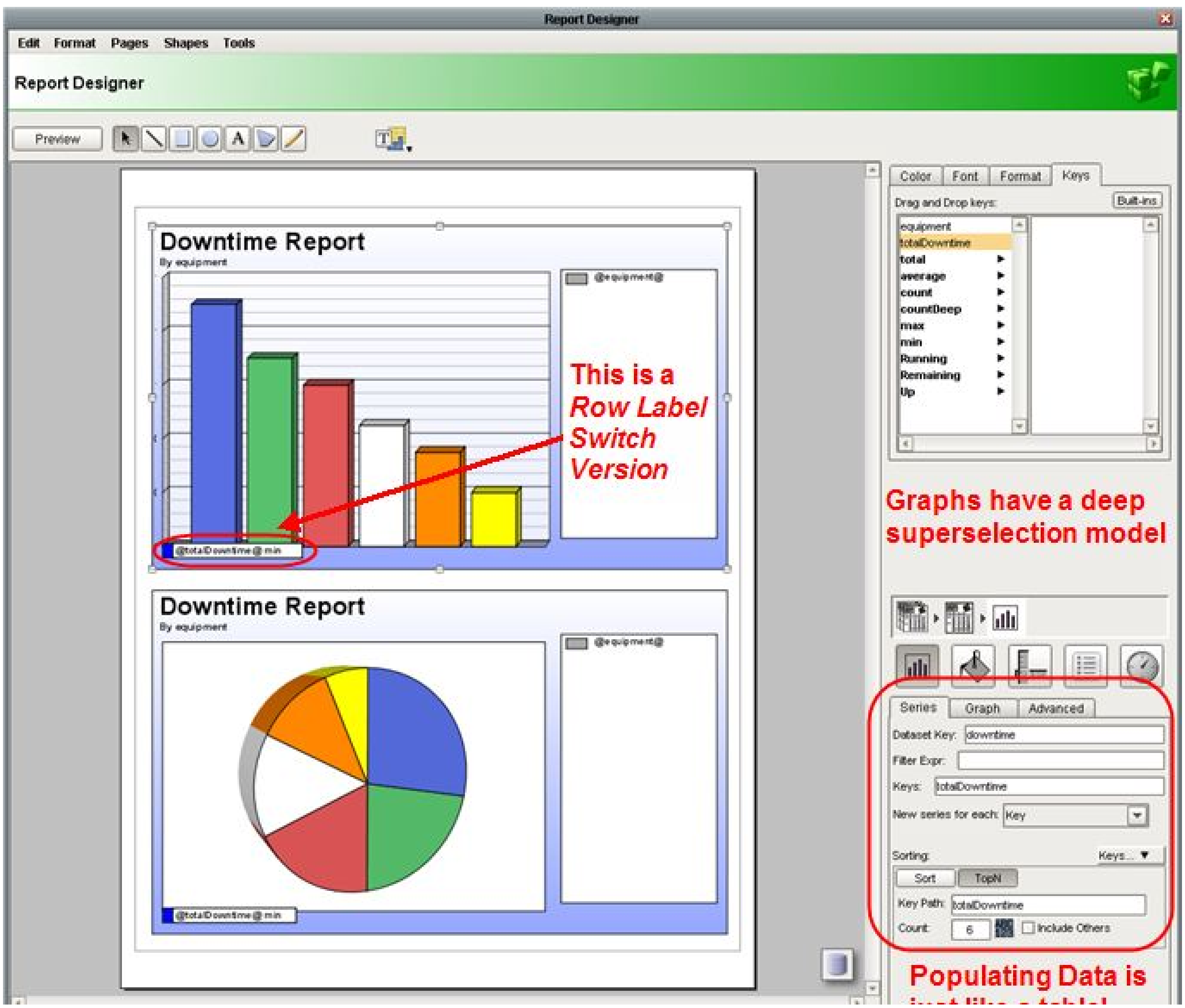
Example

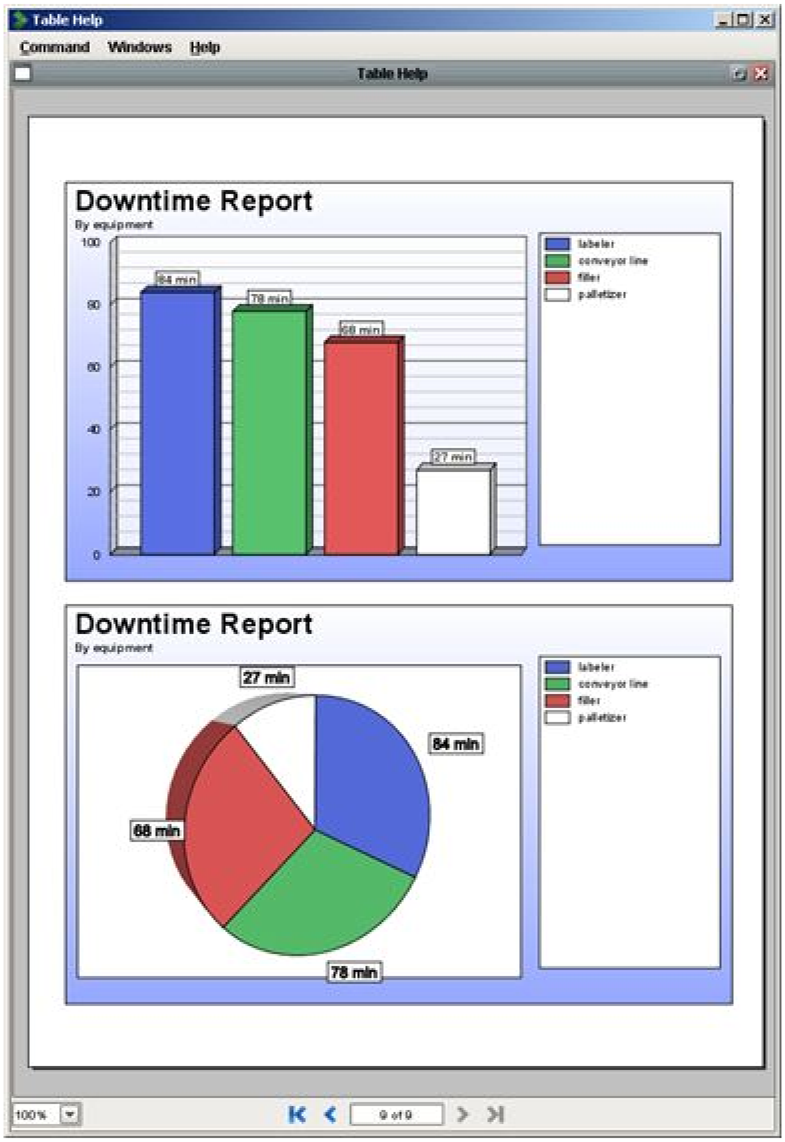
We will explore graph options with a total downtime by equipment example. The same [data]() is used as the [table]() example.

A downtime summary can be retrieved with the following SQL query:

```
SELECT equipment, sum(time) AS totalDowntime FROM downtime GROUP BY equipment;
```

equipment	downtime
conveyor line	78
filler	68
labeler	84
palletizer	27



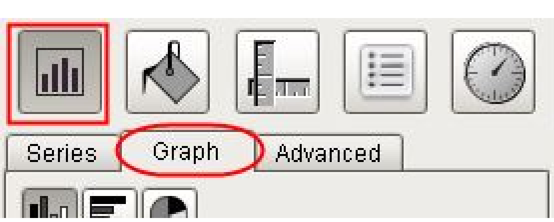


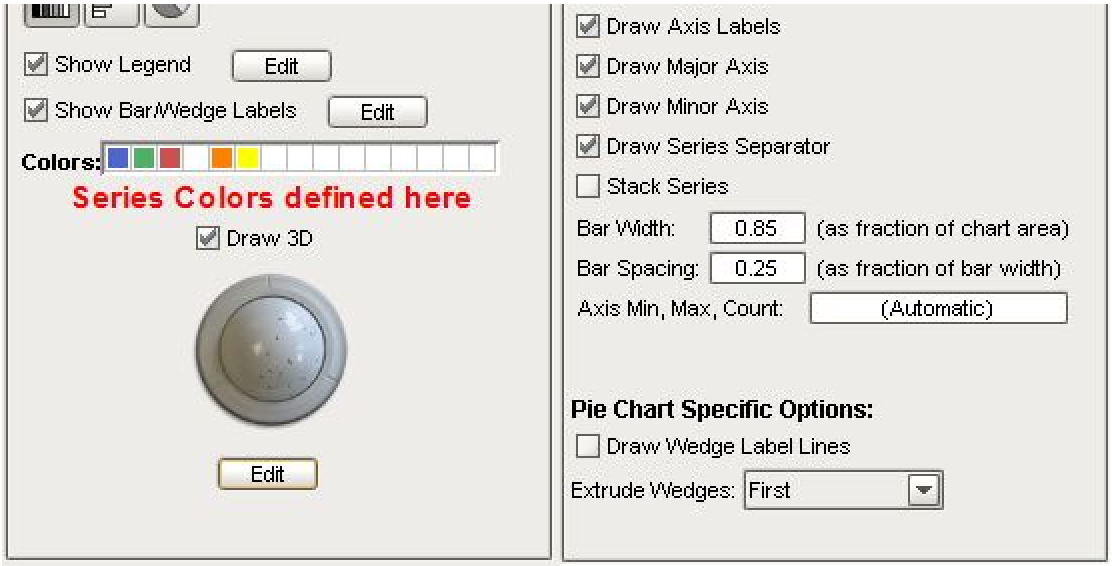
Report in the FactoryPMI runtime

Graph Settings

Basic graph settings can be found on the *Graph* Tab of the graph shape specific inspector.

Graph Menu Item	Function
Graph Type	Choose Bar, Horizontal Bar, or Pie type graph.
Show Legend	Displays a legend object to label series
Show Bar/Wedge Labels	Builtin graph labels. You may want to rotate them to create space.
Colors:	Drag colors to define graph series colors. Colors will repeat if rightmost color is white.
Draw 3D	Render your graph in a three dimensional format.





Embedding Graphs in a table row

Graphs can be embedded in table rows. Leave the Dataset Key blank to have access to the data provided at that level of grouping! This technique is demonstrated in Tutorial #2.

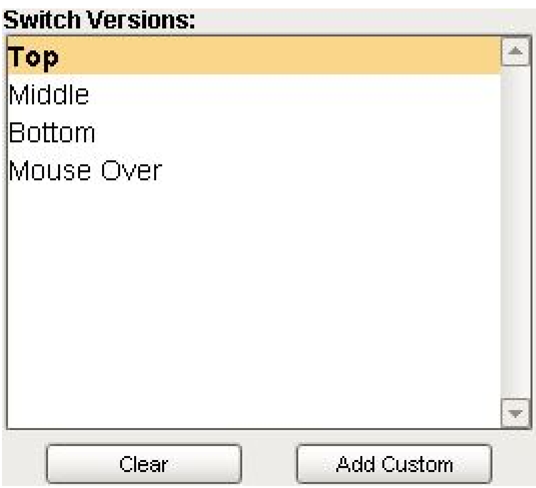
Since a graph is generally a large shape, you usually want to define an explicit page break for the row that contains the graph, so that the graph won't get chopped off on a page boundary. Select the light gray region to the left of the Group in the Table inspector to do this.

Row Label Switch Versions

Row Label Switch Versions are a way to have the graph position labels on each row (Bar in a bar graph, slice in a pie graph). Both examples above use builtin graph labels. The "**Top**" version label on a bar graph will place the label just above the top of the bar on the Y plane for each line. **Middle** and **bottom** work similarly.

You can get to the **switch versions** customizer two ways:

- Click on an existing label on the graph. This is illustrated on an image above.
- From the graph shape specific inspector, Select the *Graph* tab. Click on **Show Bar/Wedge Labels**.



Custom Children

The Graph shape supports additional custom children. Add axis labels or arbitrary text by superselecting the graph and using standard tools such as Text, Rect, Polygon, etc. You can reference keys in added text children which will be evaluated against the group of objects provided for the graph.

TIP	The best way to get the hang of graphs is to create a huge one and experiment with it.

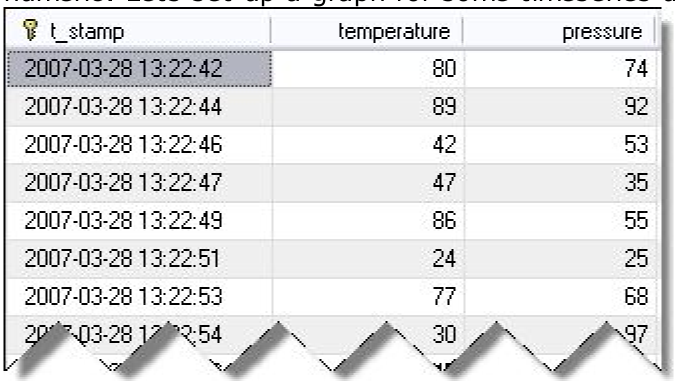
Line Graph

The *Line Graph* is a DataSet element like the table. It shows a graphical representation of data in the form of a line, area or scatter garph.

Populating data including the concepts of data keys, sorting, and filtering are nearly identical to that of a table.

Example

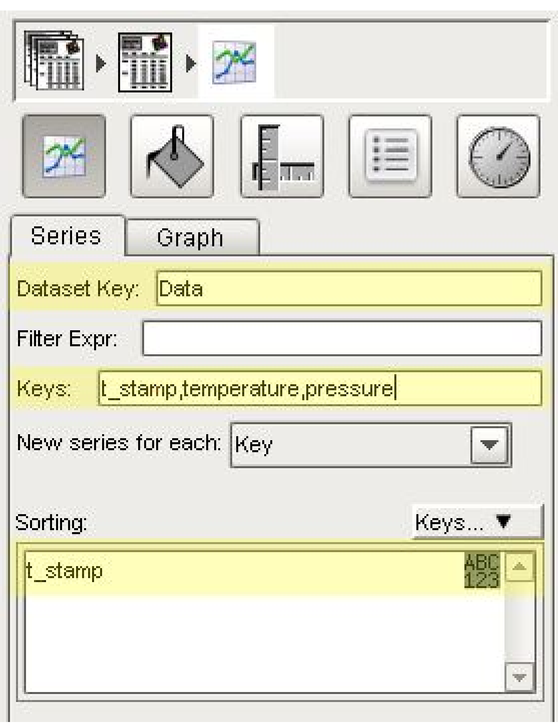
The Line Graph component is used to display data where the X value is time or numeric, and the Y value(s) are numeric. Lets set up a graph for some timeseries data. Suppose you have a table with data like this:



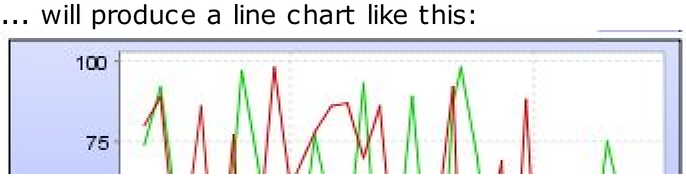
t_stamp	temperature	pressure
2007-03-28 13:22:42	80	74
2007-03-28 13:22:44	89	92
2007-03-28 13:22:46	42	53
2007-03-28 13:22:47	47	35
2007-03-28 13:22:49	86	55
2007-03-28 13:22:51	24	25
2007-03-28 13:22:53	77	68
20[illegible]-03-28 1[illegible]:54	30	97

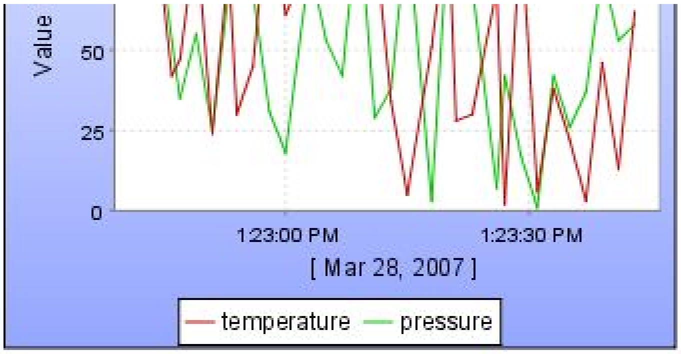
The **t_stamp** column is your X value, and the other columns are your "pens" or series of Y values. You get this data into a report by binding a DataSet property of the report viewer (see [Concepts > Basic > Dynamic Properties]) to a SQL query, such as `SELECT t_stamp, temperature, pressure FROM graph_data`. Lets say that you had this data in the default **Data** property.

You set up the Line Graph's data the same way you would a Graph or Table. The only trick is that the **keys** needs to be a comma separated list of keys, with the first one being your X value. Lastly, make sure that the data is sorted ascending by the X value. The following setup:



... will produce a line chart like this:





Line Graph Settings

Basic graph settings can be found on the *Graph* Tab of the line graph shape specific inspector.

***Graph* Menu Item**	**Function**
Graph Type	Choose Line, Area, or Scatter type graph.
Timeseries	If true, the X axis (first Key) should be a date/time. If false, the X axis should be a number.
Show Legend	Displays a legend with the name of each series (each Key besides the first one.
Show Domain Axis	If true, the domain axis (X axis) will be shown.
Domain Axis Label	The label for the domain axis. Date axes may automatically display additional label information to disambiguate certain ranges.
Show Range Axis	If true, the range axis (Y axis) will be shown.
Range Axis Label	The label for the range axis.
Range Axis Min, Max	Leave blank for automatic, or specify a range like **0,100**
Colors:	Drag colors to define graph series colors.

Embedding Graphs in a table row

Graphs can be embedded in table rows. Leave the Dataset Key blank to have access to the data provided at that level of grouping! This technique is demonstrated in Tutorial #2.

Since a graph is generally a large shape, you usually want to define an explicit page break for the row that contains the graph, so that the graph won't get chopped off on a page boundary. Select the light gray region to the left of the Group in the Table inspector to do this.

Images and Image Placeholders

Create images by clicking on the image button  on the add shapes button of the toolbar. Double click on an image in the **Image Browser** window.

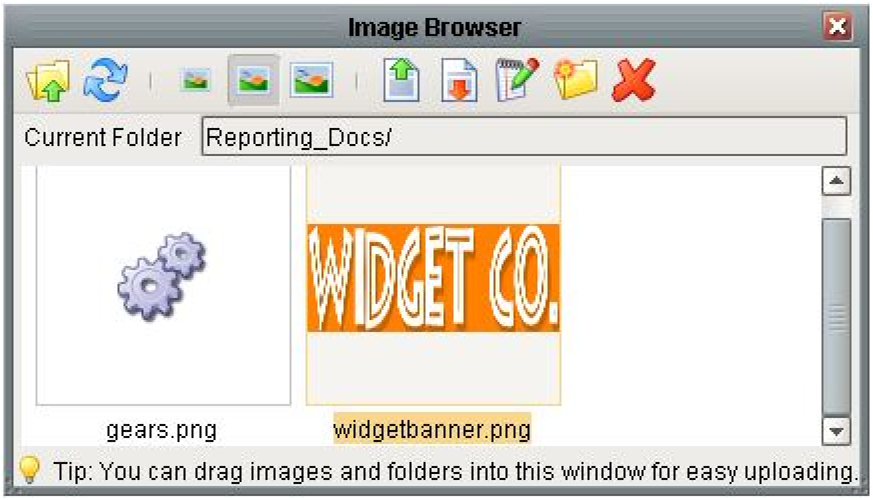


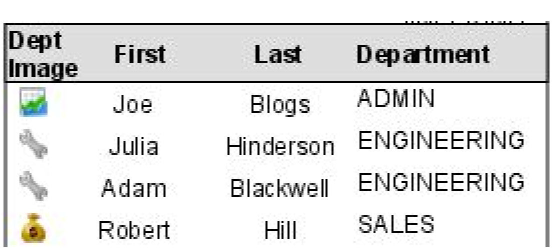
Image Options

Image options are specified in the shape specific inspector for images.

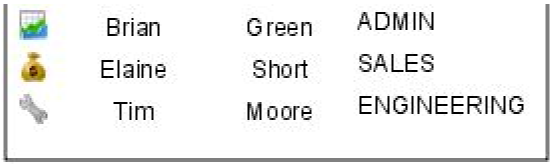
Option	Function
Key	Specify a string expression that returns an image path to change the image. Useful for a multistate image within a table.
Page	Applicable to pdfs only. Selects page number of multipage pdf to display.
Margins	Specifies how many pixels you want of margins around the image.
Style - stretch	Stretches the picture to the image object's size, regardless of aspect ratio.
Style - tile	Tiles the original sized picture within the image object, cutting off sides as necessary.
Style - fit	Resizes picture to image object maintaining aspect ratio.
Style - fit if needed	Resizes picture to image object maintaining aspect ratio, shrinking if necessary, but never enlarging.
Size borders to image	Applicable to **fit** and **fit if needed**.
Rounding Radius	Turns stroke (border) from rectangle, to rounded rectangle, to circle as the number is increased.

Image Placeholders

Images can be populated with BLOB data from an SQL database. They are referred to as *Image Placeholders* when used in this fashon. Simply define the **Key** to the Blob image.



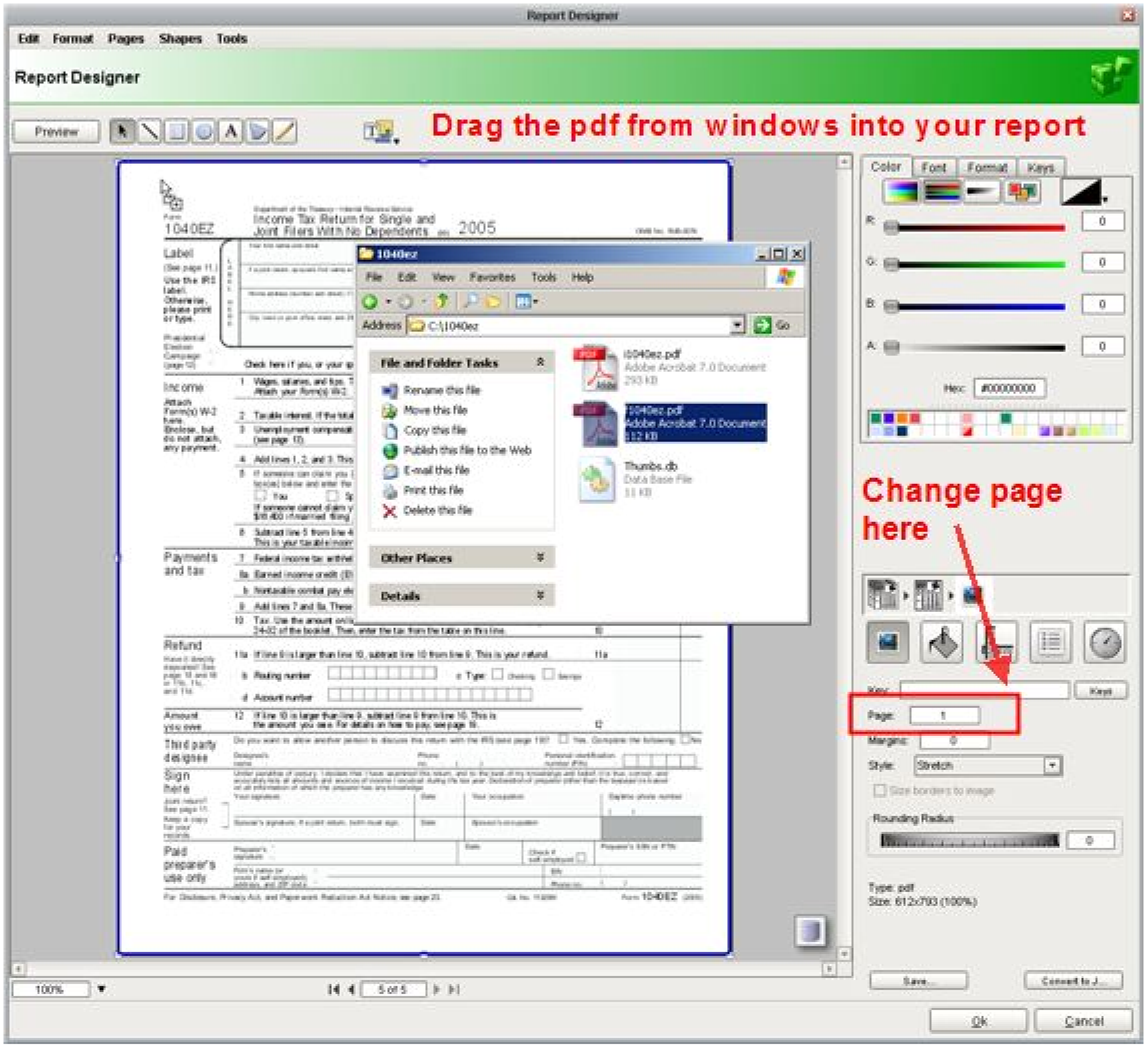
Dept Image	First	Last	Department
	Joe	Blogs	ADMIN
	Julia	Hinderson	ENGINEERING
	Adam	Blackwell	ENGINEERING
	Robert	Hill	SALES

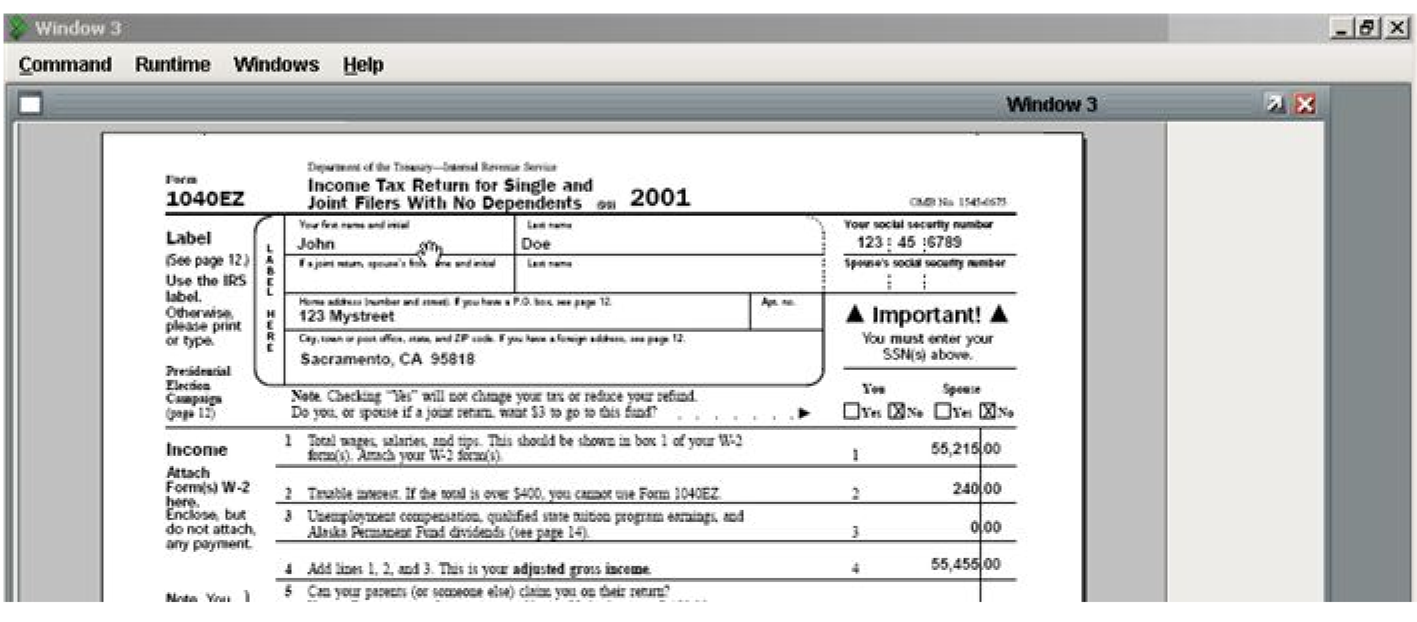


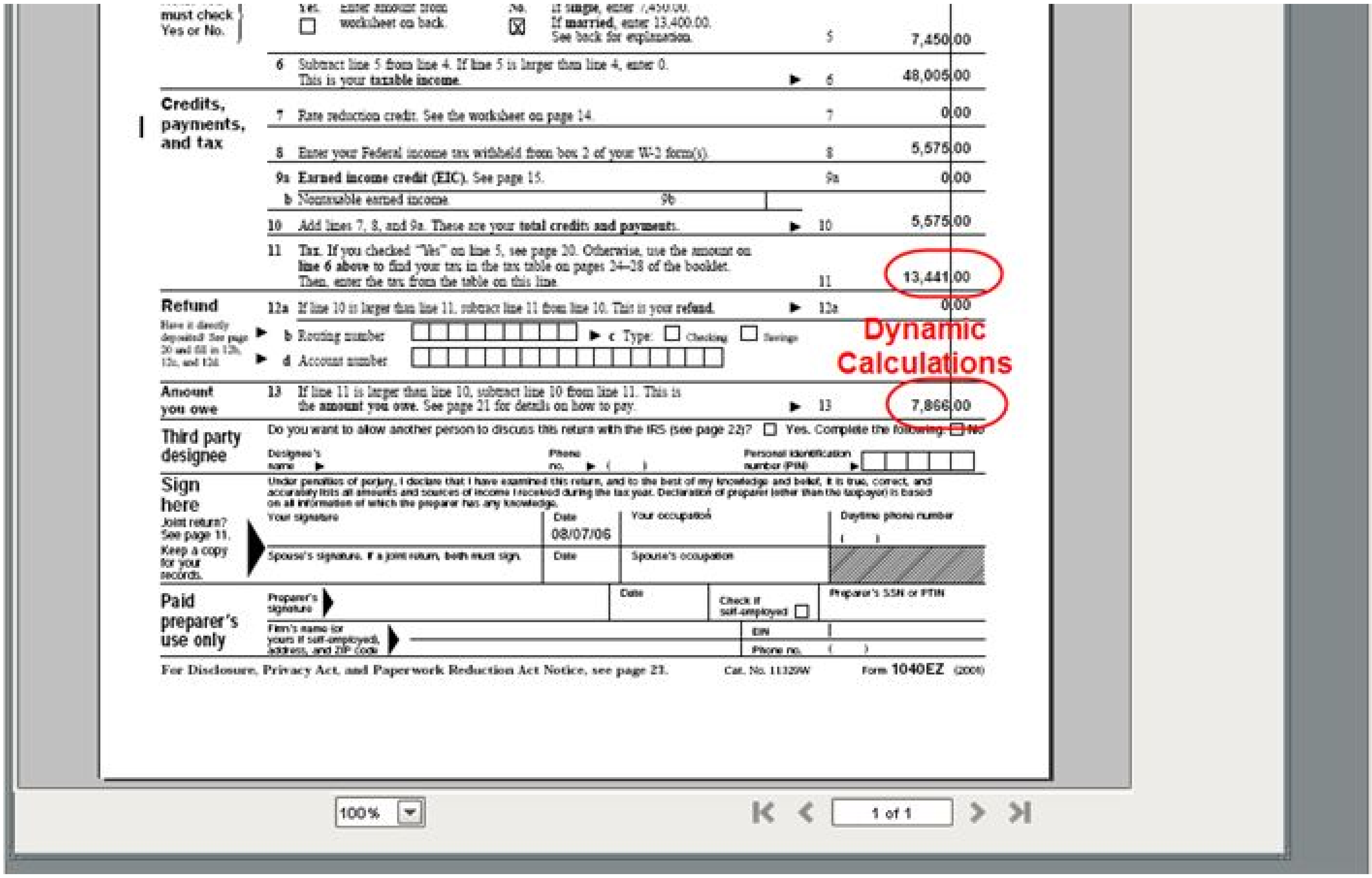
Using an *Image Placeholder* and blobs to dynamically illustrate table row based on **department**.

Using Adobe Acrobat (PDF) files

Pdf files are typically used when you have an existing report that you wish to create dynamically. Simply drag text labels or even tables on the pdf to generate a "filled in" report.







A filled out pdf report

TIP	Extract only the pdf pages that you are going to use before putting them into your report.

Labels

Labels can be used to print out mailing labels, create name tags, or any other generic labels. You can use standard Avery label sheets or specify your own dimensions.

Shape Specific Inspector Item	Function
List Key	Name of DataSet that will populate the labels
Avery Product Number	Choose from a list of Avery Label Formats
Rows/Columns	Defines the number of rows and columns on the page
Label Width/Height	Width and height of labels in pixels
Spacing Width/Height	Distance between labels on the page in pixels
Sorting	Specifies printing order. Works the same as table sorting.
Paginate	Two setting (Off ☐ or On ☑) option that determines whether or not to use page breaks. Broken are useful for pdf files, continuous are good for Flash or CSV. Typically leave this alone.

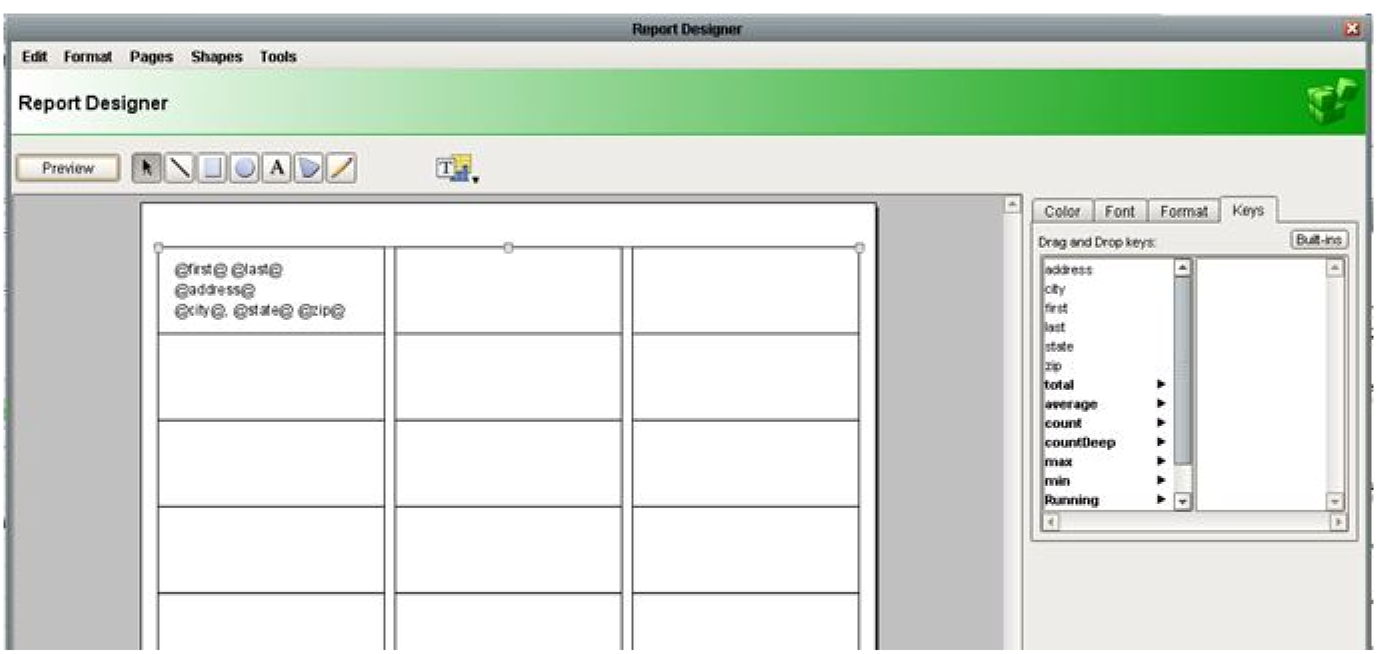
Example

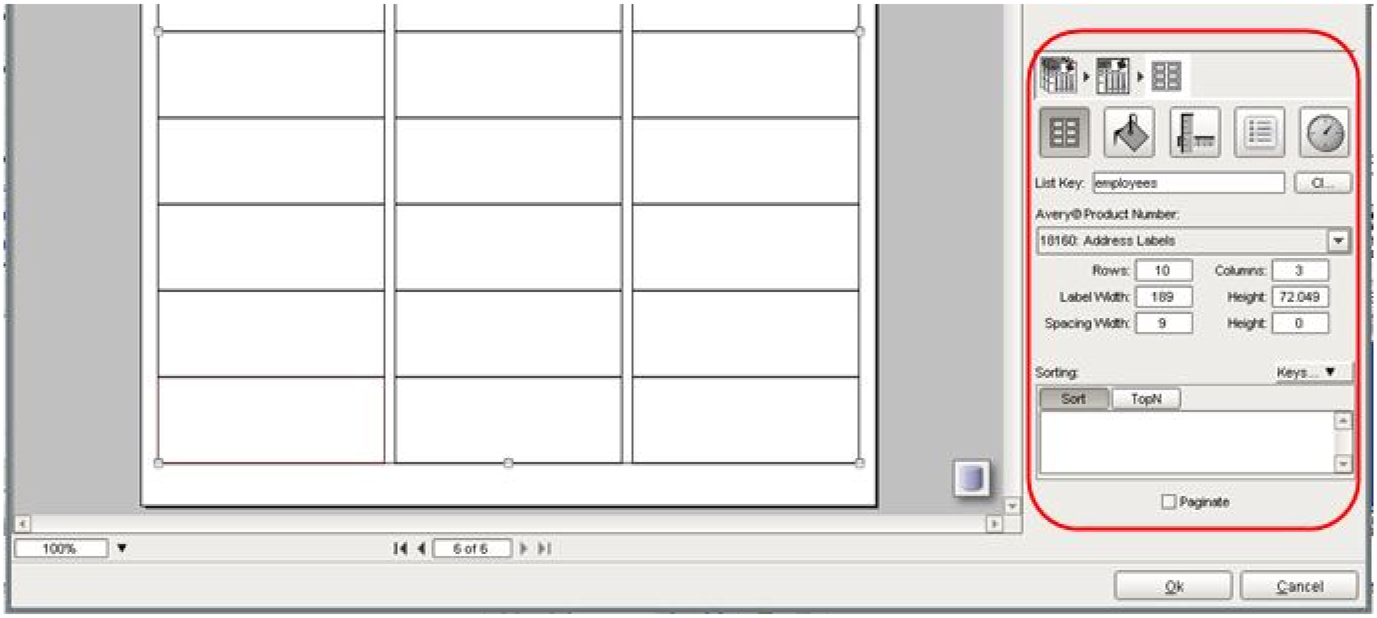
1. Create labels from tool bar or by dragging a DataSet to the report.
2. Specify the DataSet name (*employees*) as the List Key in the shape specific inspector
3. Choose the appropriate Avery Product number or manually specify dimensions
4. Create text labels. Set up your labels with substitution keys

Employee addresses can be retrieved from the database with the following SQL query:

```
SELECT e.first, e.last, a.address, a.city, a.state, a.zip FROM employees e, address a WHERE
e.id=a.emp_id;
```

first	last	address	city	state	zip
Joe	Blogs	1234 Mystreet	Sacramento	CA	95818
Julia	Hinderson	237 Winding Way	Carmichael	CA	95715
Adam	Blackwell	779 Elm Street	Sacramento	CA	95833





Joe Blogs
1234 Mystreet
Sacramento, CA 95818

Julia Hinderson
237 Winding Way
Carmichael, CA 95715

Adam Blackwell
779 Elm Street
Sacramento, CA 95833

Resulting Output

Barcode

Description

The reporting barcode component is identical to FactoryPMI's normal barcode component. It displays some text encoded as a barcode, and also displays the text verbatim below the barcode. In the report designer, you can drag a data key into the text box, and the barcode will be dynamic just like any other reporting component.

Properties

Value

The value (code) that will be encoded as a barcode. Acceptable values vary depending on the encoding type. Drag a data key (Like `@SerialNum@`) into the value box to make the barcode dynamic.

Type

The encoding type of the barcode. Types are<

- Code 39
- Code 39 Narrow
- Extended Code 39
- Extended Code 39 Narrow
- Code 128
- Codabar
- Codabar Narrow
- Interleaved Code 25
- Interleaved Code 25 Narrow
- MSI
- EAN 13
- EAN 8

Bar Width

The width of a single bar.

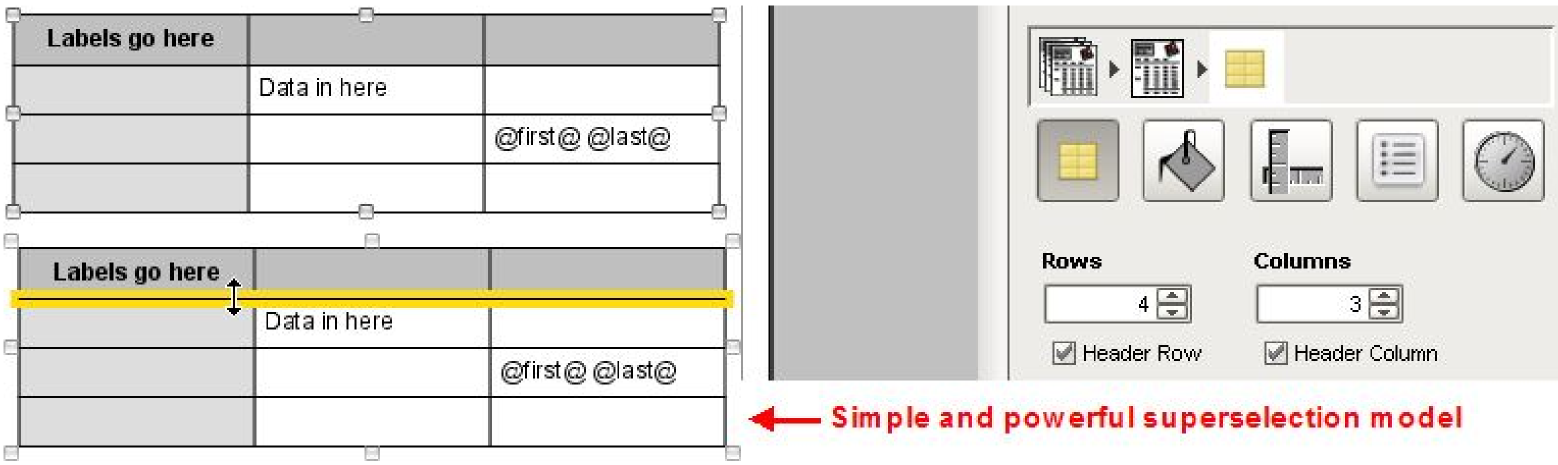
Bar Height

The height of all bars.

Simple Table

The *Simple Table* is a table of a fixed size that does not have a dataset key. It has an intuitive superselection model.

Property	Function
Rows	Specify number of rows
Columns	Specify number of columns
Header Row	Optional Header Row
Header Column	Optional Header Column



Note: **@first@** will not resolve to anything because there is not an implied dataset key. You would need a full path such as **@employees[0].first@** (unless you have the dynamic non-DataSet property, *first*).

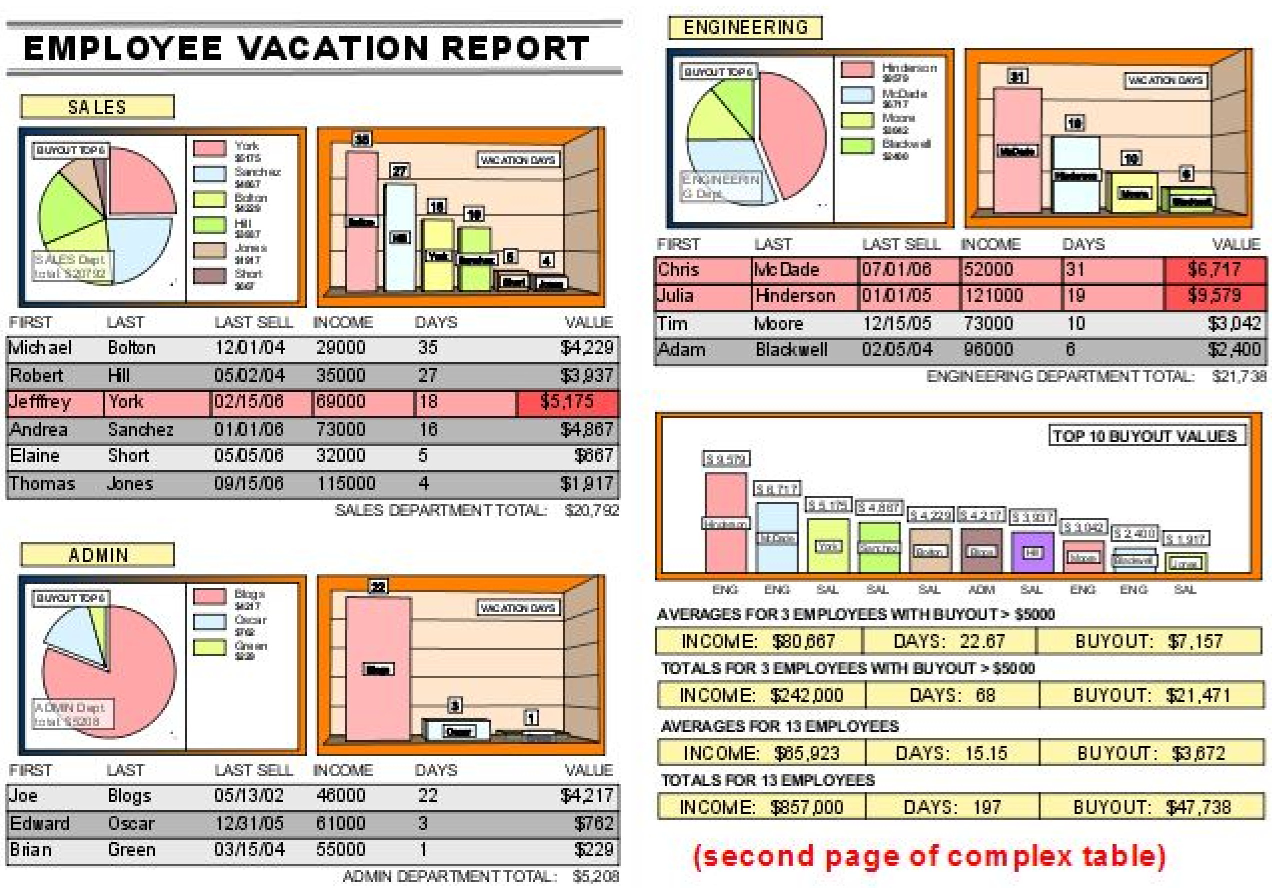
Tables - Overview

[Tables](#) are objects that display data in a structured, repetitious format. Their complexity can range from trivially simple to complicated. The Reporting engine will automatically create new pages to fit all data within the table's boundaries. Combine that feature with powerful data manipulation and layout tools, and you get an object that often forms the basis of your reports.

A Simple Table

Dec 20, 2005 17:55	labeler	50 minutes	Out of labels
Dec 22, 2005 11:55	filler	15 minutes	Scheduled maintenance
Jan 02, 2006 22:55	palletizer	10 minutes	Misalignment
Jan 03, 2006 02:55	conveyor line	25 minutes	backup
Feb 12, 2006 06:13	labeler	10 minutes	<NA>
Feb 12, 2006 12:01	labeler	3 minutes	Out of labels
Feb 12, 2006 14:01	palletizer	17 minutes	Misalignment
Feb 12, 2006 16:23	conveyor line	23 minutes	Scheduled maintenance
Feb 12, 2006 20:04	filler	33 minutes	Overflow
Feb 12, 2006 20:13	labeler	21 minutes	Stuck labels
Feb 12, 2006 20:25	filler	20 minutes	Overflow
Feb 12, 2006 20:36	conveyor line	30 minutes	Scheduled maintenance

A Complex Table



The above table was created in [Tutorial #2](#). It uses the following features:

- [Header](#), [detail](#), and [summary](#) rows
- [Grouping](#)
- [Sorting](#)
- [Row Versioning](#)

Next (Table Basics)

TIP	Table related help sections can be referenced independently, but will be written so that examples follow sequentially.

Tables - Basics

Let's go through the process of creating a simple table! We will cover: getting data into the report, creating a table, defining data, and explore basic parts. Make sure you understand how the Dataset key defines the table's DataSet.

Date	Equipment	Downtime	User entered comment
Dec 20, 2005 17:55	labeler	50 minutes	Out of labels
Dec 22, 2005 11:55	filler	15 minutes	Scheduled maintenance
Jan 02, 2006 22:55	palletizer	10 minutes	Misalignment
Jan 03, 2006 02:55	conveyor line	25 minutes	backup
Feb 12, 2006 06:13	labeler	10 minutes	<NA>
Feb 12, 2006 12:01	labeler	3 minutes	Out of labels
Feb 12, 2006 14:01	palletizer	17 minutes	Misalignment
Feb 12, 2006 16:23	conveyor line	23 minutes	Scheduled maintenance
Feb 12, 2006 20:04	filler	33 minutes	Overflow
Feb 12, 2006 20:13	labeler	21 minutes	Stuck labels
Feb 12, 2006 20:25	filler	20 minutes	Overflow
Feb 12, 2006 20:36	conveyor line	30 minutes	Scheduled maintenance

Summary goes here

Resulting Basic Table

Getting data into the report

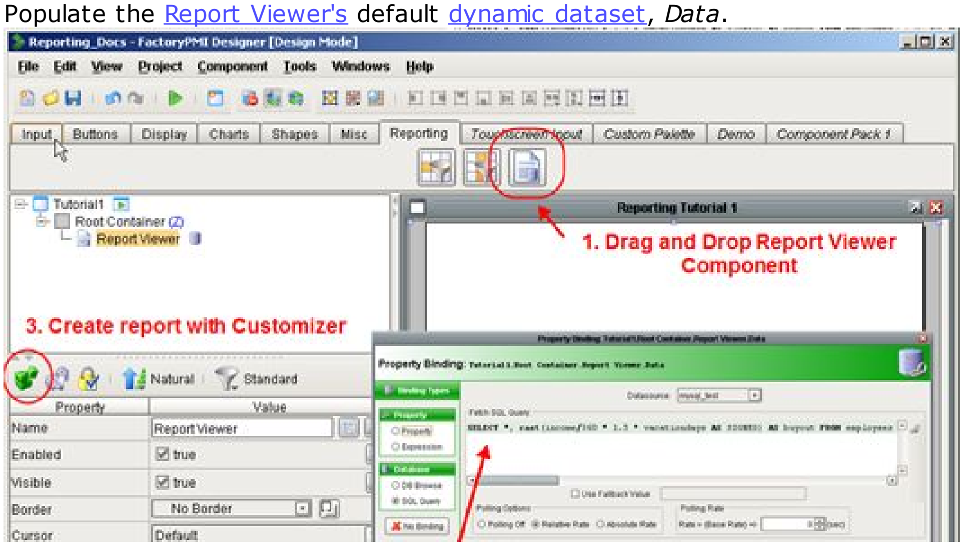
Before creating a useful table, you must get the data from the SQL database into the Report Viewer.

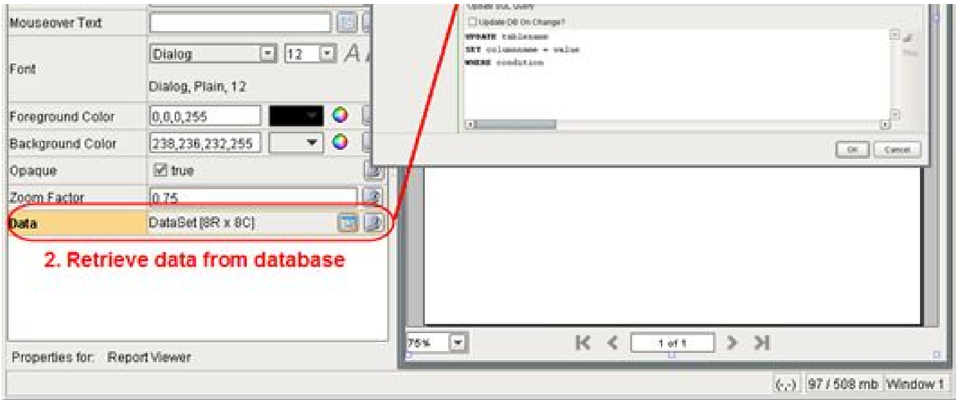
Example downtime data can be retrieved with the following SQL query:

```
SELECT * FROM downtime;
```

downtime_ndx	t_stamp	equipment	time	comment
1	2005-12-20 17:55:00	labeler	50	Out of labels
2	2005-12-22 11:55:00	filler	15	Scheduled maintenance
3	2006-01-02 22:55:00	palletizer	10	Misalignment
4	2006-01-03 02:55:00	conveyor line	25	backup
5	2006-02-12 06:13:00	labeler	10	
6	2006-02-12 12:01:00	labeler	3	Out of labels
7	2006-02-12 14:01:00	palletizer	17	Misalignment
8	2006-02-12 16:23:00	conveyor line	23	Scheduled maintenance
9	2006-02-12 20:04:00	filler	33	Overflow
10	2006-02-12 20:13:00	labeler	21	Stuck labels
11	2006-02-12 20:25:00	filler	20	Overflow
12	2006-02-12 20:36:00	conveyor line	30	Scheduled maintenance

Populate the Report Viewer's default dynamic dataset, *Data*.



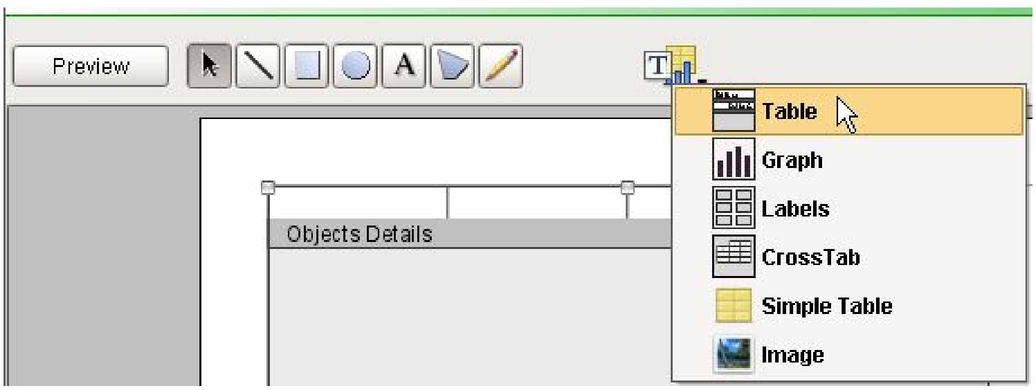


(Illustration from [Tutorial #1](#)).

Your report now has data. You're ready to create a table!

Creating a Table

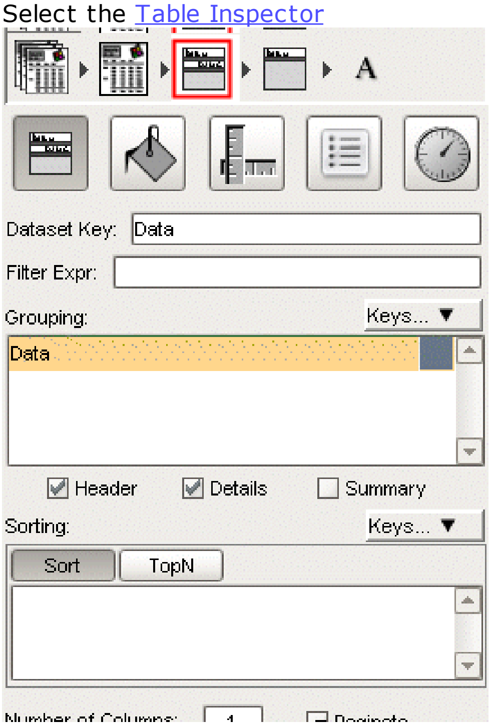
1. Open the [Report Designer](#) by selecting the [Report Viewer](#) in the FactoryPMI Designer and applying the customizer (Cntl+U).
2. Click the table icon on the [add shapes button](#) of the [toolbar](#).
3. Size and position table as desired.

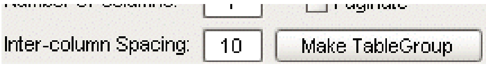


Defining Data

The **Dataset Key** is the name of the DataSet that a table or graph is getting its input from. *@yourSubstitutionKey@* in the table with a defined Dataset key will work as if it were *@DataSet_Key.yourSubstitutionKey@*

1. Click the table to select it
2. Select the [Table Inspector](#)

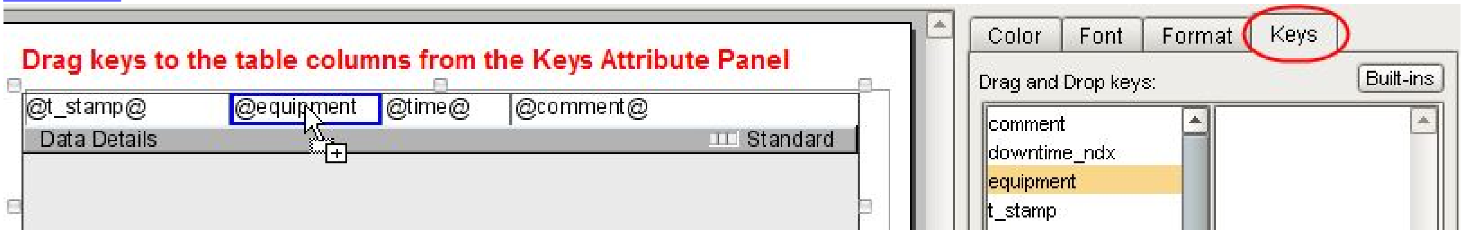




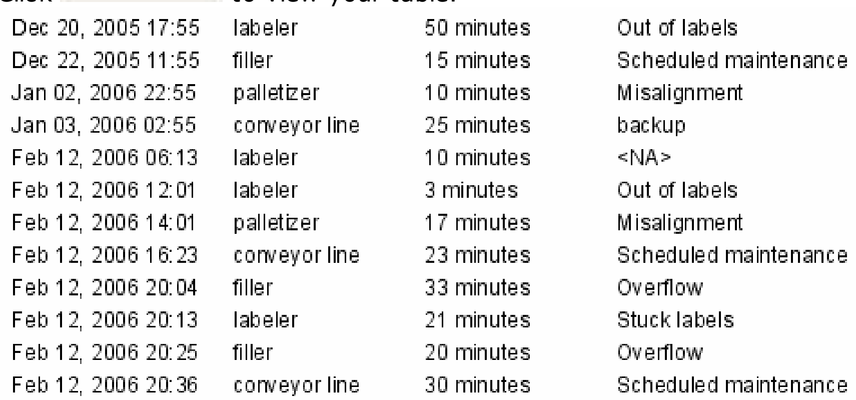
3. Under **Dataset Key:**, type *Data* or drag the *Data* DataSet from the Keys Attribute Panel, choose **table** and click ok. This is the step that defines which DataSet this table will use. You may only define one per table. If you created the table by dragging the DataSet, you will not need to fill in the **DataSet Key** in the next section.

   With a defined dataset key, your table can reference that data without explicitly defining the path. For example, in this table, **@Data.comment@** is the same as **@comment@**.

4. Drag Keys to the table columns from the Keys Attribute Panel. We appended the string "minutes" to the time label and formatted the date.

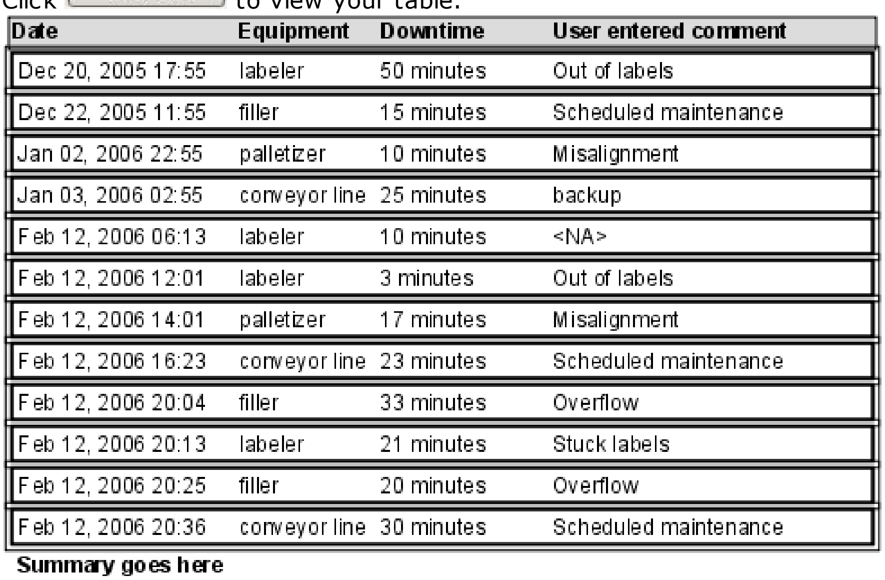


5. Click  to view your table.



Dec 20, 2005 17:55	labeler	50 minutes	Out of labels
Dec 22, 2005 11:55	filler	15 minutes	Scheduled maintenance
Jan 02, 2006 22:55	palletizer	10 minutes	Misalignment
Jan 03, 2006 02:55	conveyor line	25 minutes	backup
Feb 12, 2006 06:13	labeler	10 minutes	<NA>
Feb 12, 2006 12:01	labeler	3 minutes	Out of labels
Feb 12, 2006 14:01	palletizer	17 minutes	Misalignment
Feb 12, 2006 16:23	conveyor line	23 minutes	Scheduled maintenance
Feb 12, 2006 20:04	filler	33 minutes	Overflow
Feb 12, 2006 20:13	labeler	21 minutes	Stuck labels
Feb 12, 2006 20:25	filler	20 minutes	Overflow
Feb 12, 2006 20:36	conveyor line	30 minutes	Scheduled maintenance

6. Click "Header" and "Summary" check boxes in the Table Inspector. Add text labels to Header and Summary.
7. Select **Data Header**, add a Fill Color (Background) in the Fill & Stroke Inspector.
8. Select **Data Details**, add a Stroke Color (Outline) in the Fill & Stroke Inspector.
9. Adjust text, fill, and stroke as desired.
10. Click  to view your table.



Date	Equipment	Downtime	User entered comment
Dec 20, 2005 17:55	labeler	50 minutes	Out of labels
Dec 22, 2005 11:55	filler	15 minutes	Scheduled maintenance
Jan 02, 2006 22:55	palletizer	10 minutes	Misalignment
Jan 03, 2006 02:55	conveyor line	25 minutes	backup
Feb 12, 2006 06:13	labeler	10 minutes	<NA>
Feb 12, 2006 12:01	labeler	3 minutes	Out of labels
Feb 12, 2006 14:01	palletizer	17 minutes	Misalignment
Feb 12, 2006 16:23	conveyor line	23 minutes	Scheduled maintenance
Feb 12, 2006 20:04	filler	33 minutes	Overflow
Feb 12, 2006 20:13	labeler	21 minutes	Stuck labels
Feb 12, 2006 20:25	filler	20 minutes	Overflow
Feb 12, 2006 20:36	conveyor line	30 minutes	Scheduled maintenance

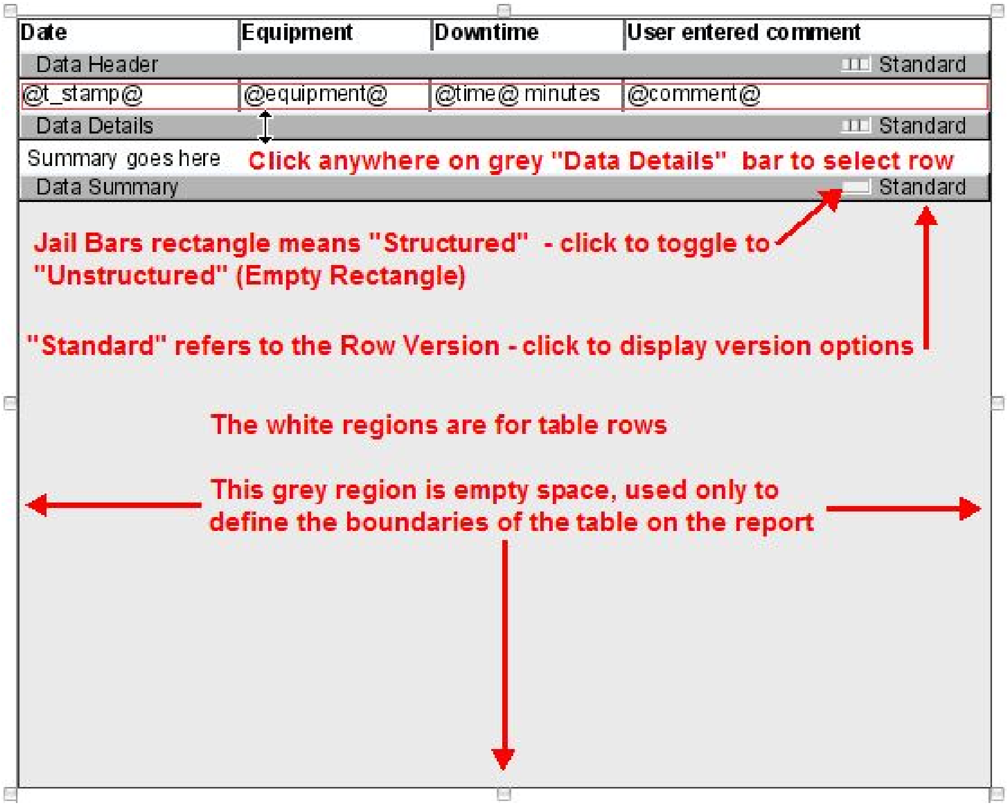
Summary goes here

Anatomy of a Table

There aren't many parts to a table.

- You have the entire table to define the region on the report that the table occupies. Much area of simple tables often end up as a placeholder.

- Header, detail, and summary rows are optional for each level of Grouping.
- The Table Inspector and Table Row Inspector are where table configuration occur.



Previous (Table Overview)   Next (Table Rows)

TIP	Tables can also be created by dragging a DataSet to your report. This will automatically set the **Dataset Key**.

Tables - Rows: Header, Detail, and Summary

Rows are an important fundamental aspect of tables. The different types of rows can be independently enabled for each level of Grouping. Table Row Versioning gives you the option of conditionally displaying rows with a different format.

Header Row

The header row is used to add such report features as column labels. An interesting feature of the header is reprint versioning, which allows a different header on every page after the first. The main data in a table has one header row. Each subgroup of data can have its own row header.

With grouping, the "top" level **Header** is the first row for the entire report. Lower level **Headers** fall immediately below higher level **Details**. In many cases where one is used, the other could be used equivalently in its case.

Detail Rows

The detail rows typically represent the majority of the data on a table or the "middle" rows. You might disable detail rows in unusual situations such as only displaying aggregate summaries in a grouped report.

With grouping, the **Detail** rows appear below the same level **Header** and above the **Header** of the next level.

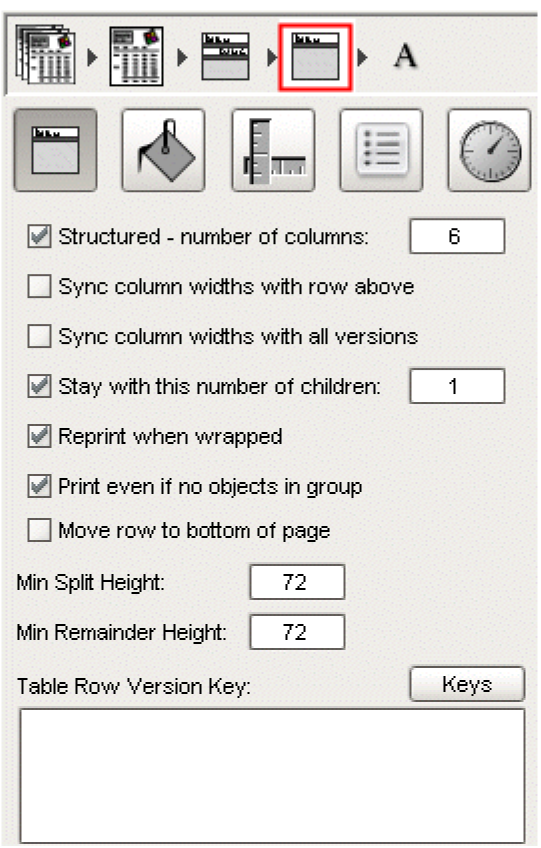
Summary Row

The summary row works like the header row. It prints at the bottom of the table.

With grouping, **Summary** rows are always last, always in the opposite order of the **Headers**.

Row Properties

Row properties are defined in the Table Row Inspector.



Row Precedence Example

Suppose you have a table with the following levels of grouping: First, Middle, Data. *Data* is your main DataSet, first and middle are strings (or numbers). The following is the order of grouping:

Group	Section
First	**Header**
First	**Details**
Middle	**Header**
Middle	**Details**
Data	**Header**
Data	**Details**
Data	**Summary**
Middle	**Summary**
First	**Summary**

Previous (Table Basics)   Next (Table Row Versioning)

TIP	Table Row Versioning allows any given row to use different constructions based on an expression. This gives you options like: alternating row background color, emphasizing alarm states, and conditionally displaying different information in general.

Tables - Sorting and Filtering

Sorting orders your data by a key or list of keys. Filtering excludes data based on some condition. Both are done in the table inspector.

Sorting

There are two similar methods of sorting. They can be ascending (ABC 123) or descending (321 ZYX) and can use aggregate keys.

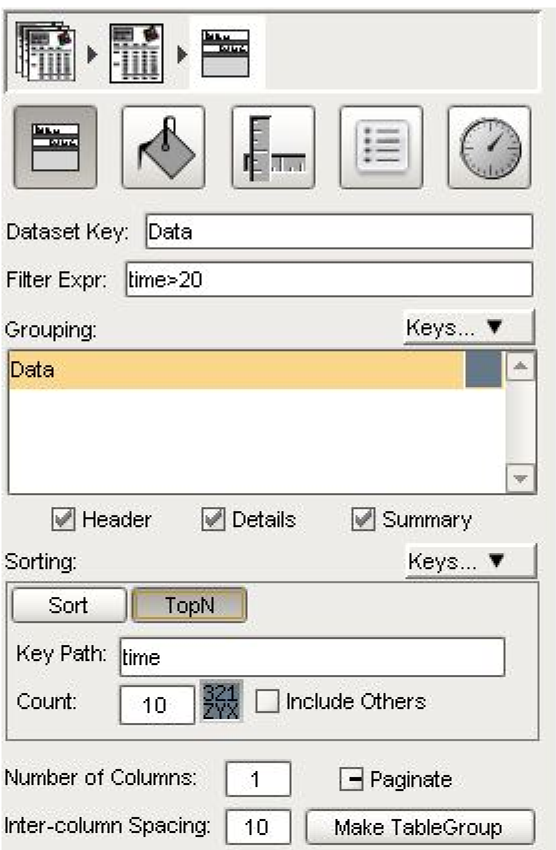
Sort takes a list of keys and sorts by the first one. In the event of a tie it goes down the list.

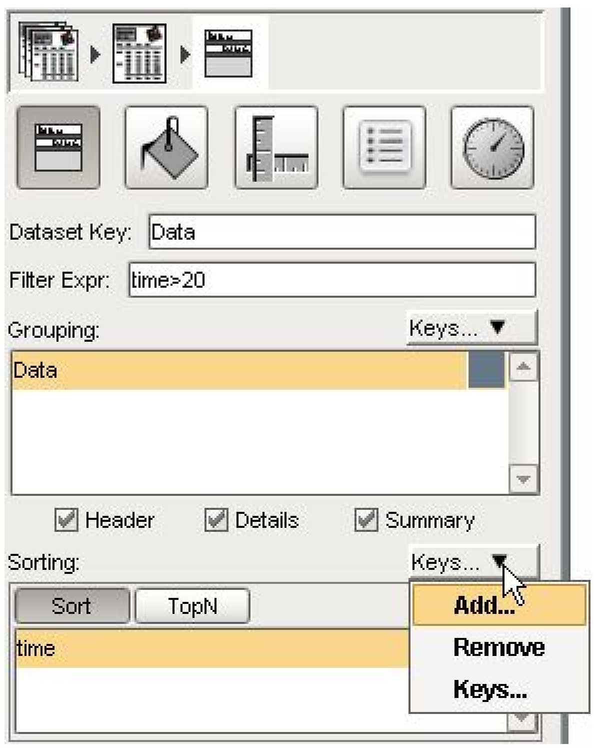
TopN uses a single key path. The **Count** option allows a limit to the number of rows processed.

Filtering

Filtering gives the option of processing data based on an expression. If the expression resolves false, the row will be skipped. Note: you do not need @ symbols to reference keys.

Example





Date	Equipment	Downtime	User entered comment
Dec 20, 2005 17:55	labeler	50 minutes	Out of labels
Feb 12, 2006 20:04	filler	33 minutes	Overflow
Feb 12, 2006 20:36	conveyor line	30 minutes	Scheduled maintenance
Jan 03, 2006 02:55	conveyor line	25 minutes	backup
Feb 12, 2006 16:23	conveyor line	23 minutes	Scheduled maintenance

This example is sorted descending by downtime and filtered by downtime greater than 20 minutes.

Previous (Table rows) Next (Rows Versioning)

TIP	The term *processed* is used instead of *displayed* because **TopN** and **Filtering** work with aggregate functions. Filtered data is treated as if it didn't exist.

Tables - Row Versioning

Row versions allow you to conditionally display rows of data in different format. They are used to make certain data stand out or to make your report more legible.

Row versions are either builtin or user defined and may be specified with a version key expression. They are applicable to header, detail, and summary rows

Builtin Row Versions

By default reports use the **Standard** row version. Here are the builtin row versions:

Built in Version	Description
Standard	Default row version
Alternate	Applies every other row. Good for grey-bar reports by changing the background color.
Reprint	Applies every page after the first. Good for one time headers or (continued) indications to save space.
First Only	Applies only to the first instance of the row. Good for showing header information without using an upper level detail row.
TopN	Applies to **count** number of rows in a **TopN** sort. Using "include others" will then distinguish between **TopN** and non-**TopN** rows.
Split Header	Applies to headers that has been split due to excessive height. Good for providing "Continued" type indicators.
Running (Footer)	Provides a different footer row for a table whose data extends to the next page.
Mouse Over (N/A)	Used for interactive highlighting in flash based reports. (Not applicable at this time).

User Defined Row Versions

User defined row versions are identified by a string based name. They will be used when the Row Version Key expression is a string that matches the row version name.

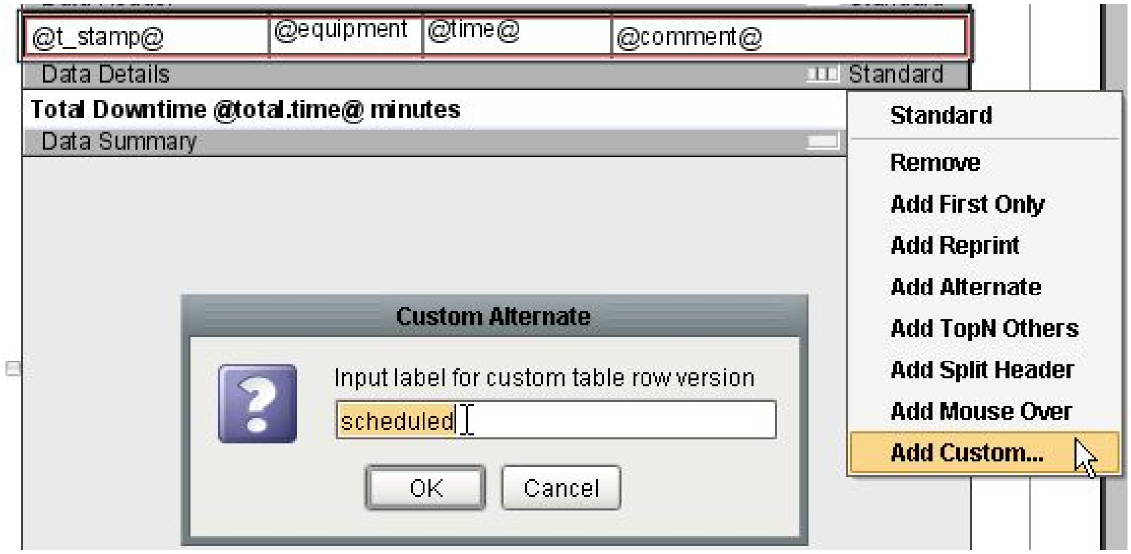
Row Version Key

The *Row Version Key* is an expression that must return a string. If that string equals the name of a row version, either builtin or user defined, that version will be used. An invalid string will default back to normal builtin row version behavior.

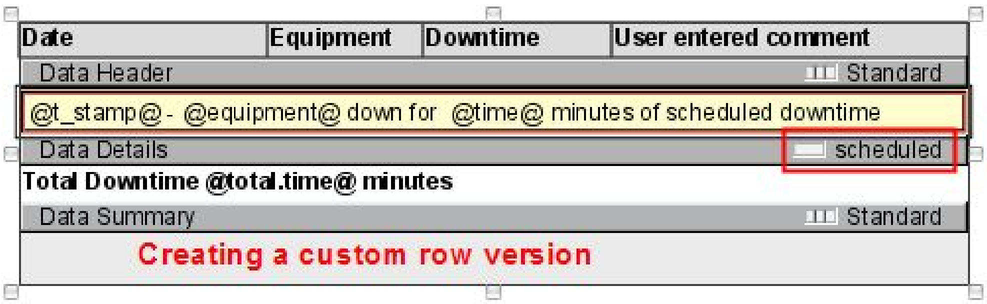
Example

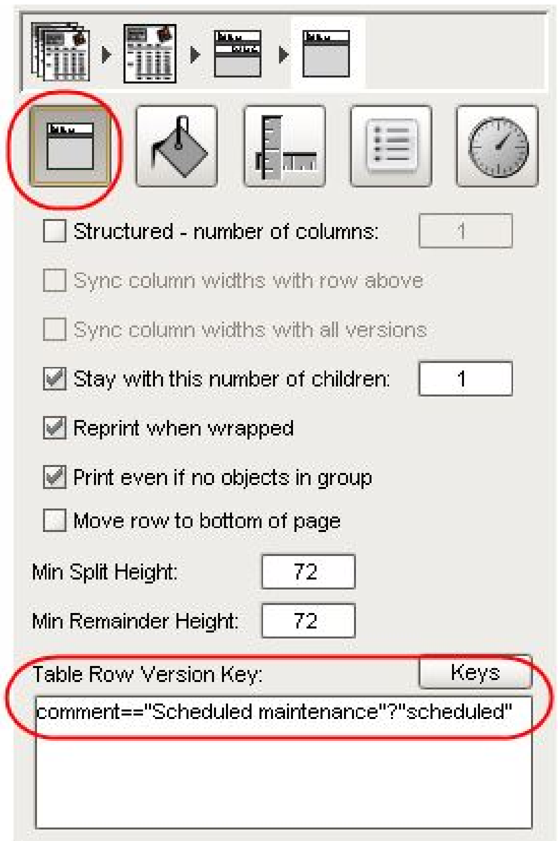
1. Add a custom row version. *scheduled*, in this case.
2. Select your row and customize it
3. Specify **Table Row Version Key**. Tip: start with the expression "scheduled" to try out your custom row version before using more complex expressions. In this case we use: IF comment = "Scheduled maintenance" THEN use our custom row version.

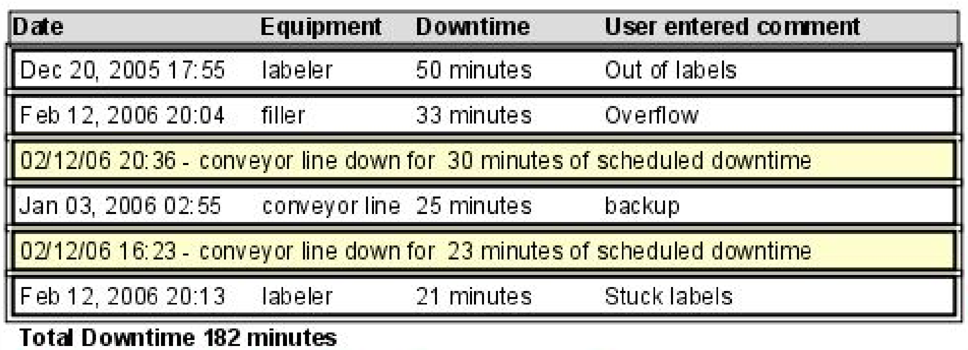
TIP	When using an IF condition for row versioning leave out the ELSE. Your table will then still respect builtin row versions. If you defaulted the ELSE to "Standard", none of the builtin versions such as *Alternate* would ever appear.



Add a custom row version







Date	Equipment	Downtime	User entered comment
Dec 20, 2005 17:55	labeler	50 minutes	Out of labels
Feb 12, 2006 20:04	filler	33 minutes	Overflow
02/12/06 20:36 - conveyor line down for 30 minutes of scheduled downtime			
Jan 03, 2006 02:55	conveyor line	25 minutes	backup
02/12/06 16:23 - conveyor line down for 23 minutes of scheduled downtime			
Feb 12, 2006 20:13	labeler	21 minutes	Stuck labels

Total Downtime 182 minutes

Preview of table with custom row version

[Previous (Table sorting)](#) [Next (Table grouping)](#)

TIP	Make sure that you're happy with the *Standard* row version before you create other row versions. This will save you time as other versions begin as a copy of *Standard*.

Tables - Grouping

Grouping breaks tables down by keys that share a common value. Tables support an arbitrary level of groups. Each can have its own header, detail, and summary rows. Additionally, totals and other aggregate functions are supported for any level of grouping.

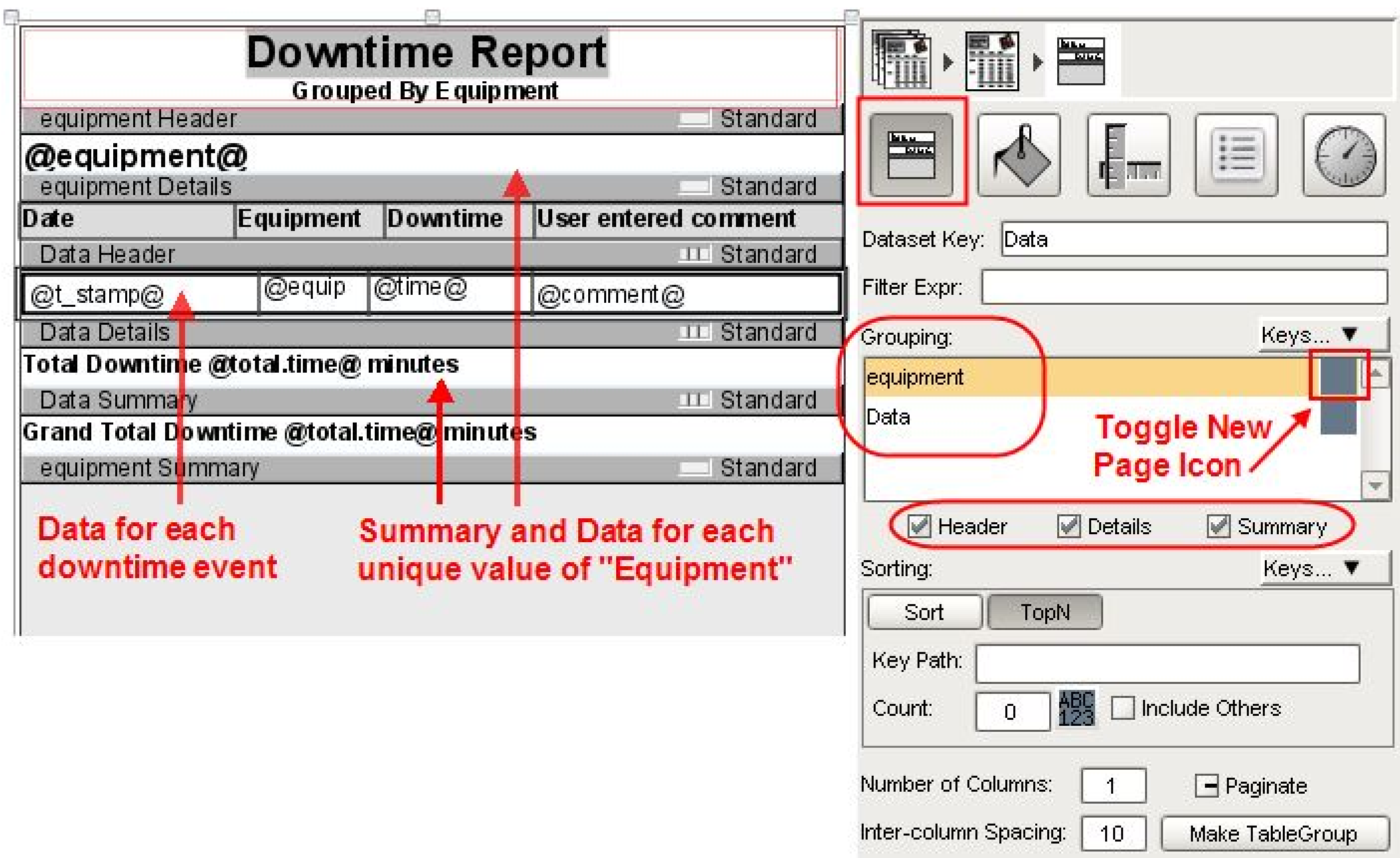
See Table Rows for specifics on row precedence with grouped tables.

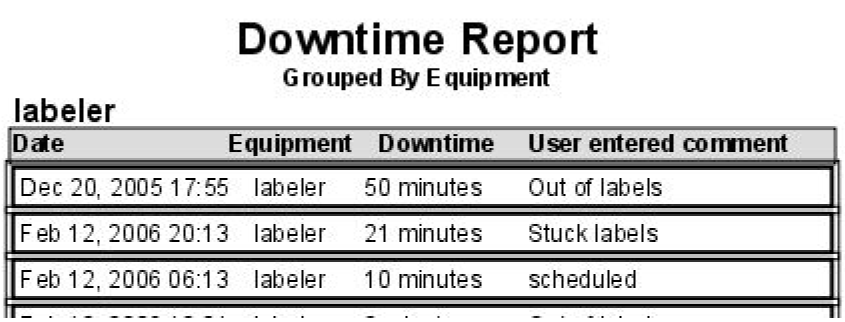
Example

This example begins with the Table Basics example. We'll group our existing downtime report table by equipment.

1. Drag the *equipment* key into the grouping table inspector.
2. Check header, detail, and summary to enable all.
3. Add headers and details.
4. Use **@total.time@** for both summary rows. Notice that the **total** respects grouping.

   In the **equipment Summary** row *total.time* is a sum of all time at that level of grouping, which includes **all** downtime events. In the **Data Summary** row *total.time* is a sum of all downtime at that level of grouping, total time that has already been grouped by equipment, equivalently, total downtime by equipment.





Feb 12, 2006 12:01	labeler	3 minutes	Out of labels

Total Downtime 84 minutes

filler

Date	Equipment	Downtime	User entered comment
Feb 12, 2006 20:04	filler	33 minutes	Overflow
Feb 12, 2006 20:25	filler	20 minutes	Overflow
Dec 22, 2005 11:55	filler	15 minutes	Scheduled maintenance

Total Downtime 68 minutes

palletizer

Date	Equipment	Downtime	User entered comment
Feb 12, 2006 14:01	palletizer	17 minutes	Misalignment
Jan 02, 2006 22:55	palletizer	10 minutes	Misalignment

Total Downtime 27 minutes

conveyor line

Date	Equipment	Downtime	User entered comment
Feb 12, 2006 20:36	conveyor line	30 minutes	Scheduled maintenance
Jan 03, 2006 02:55	conveyor line	25 minutes	backup
Feb 12, 2006 16:23	conveyor line	23 minutes	Scheduled maintenance

Total Downtime 78 minutes

Grand Total Downtime 257 minutes

Separating Groups with new pages

Clicking on the gray box of a particular level of grouping on the **grouping** panel of the table inspector will change the icon from the *default icon* to the *New Page* icon. Each new instance of that level of grouping will create a new page in the report.

In the example above, separating the *equipment* level of grouping by page would create separate report pages for the following: labeler, filler, palletizer, and converyor line.

[Previous (Table Row Versioning)]

Tip	Double Clicking a key while a table is selected will add that key to the grouping list and add it as a table row.

Table Groups

Table groups allow you to specify child tables for each object in the master list (using a list key found in each of those objects). It also allows you to specify additional "peer" tables that pick-up exactly where the first table ends (note: multiple tables can also be configured as multiple- page templates, providing a page break between tables).

Use

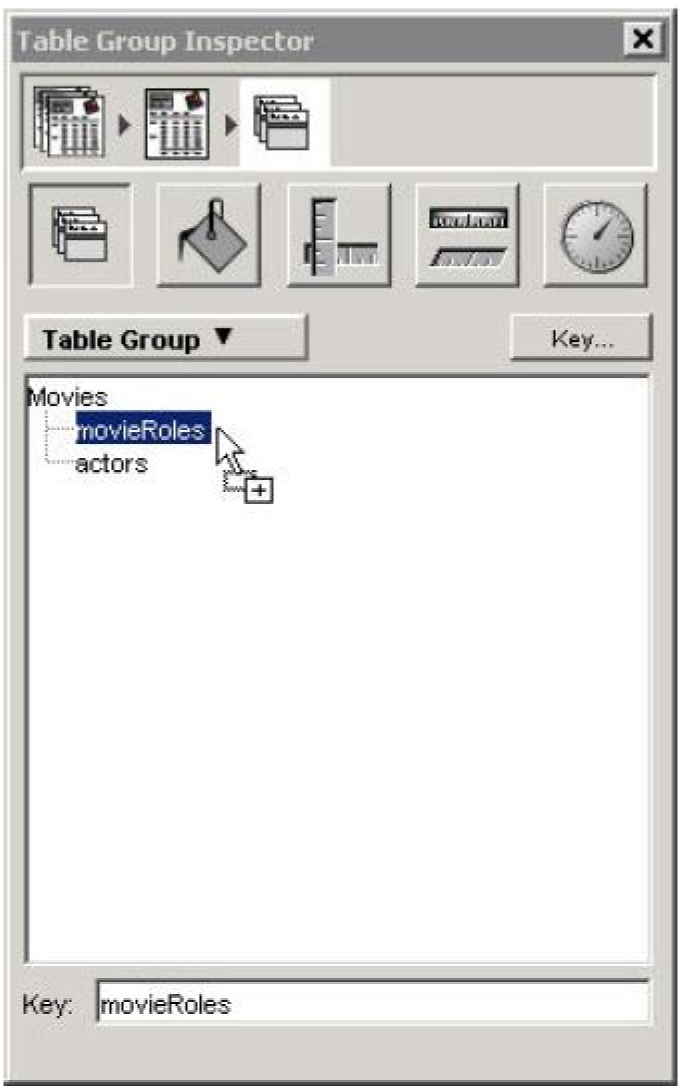
To turn a table into a table group, simply select the table and click the "Group in Table Group" button in the table inspector. The table is actually a child of a "Table Group" element, which has it's own inspector.

Now you can drag any list key of the master table into the table group's table tree to add a child table (the Table Group pull-down menu also provides a way to add child or peer tables). This will add a whole new table for this "child" list key. You can edit each of the different tables in the table hierarchy by clicking their node in the table tree. Double-click a node to get its table inspector (or double click on the table template in the open document).

You can get back to the table group inspector by clicking on the "Table Group" button at the bottom left corner of the table template, or by selecting the table group icon in the "Selection Path" area of the inspector.

Parent Reference

To reference the parent row object from a child table, you can simply use the key prefix "Parent". So if a row in a movie role child table wanted to display the movie title, it could use the key "@Parent. getTitle@".



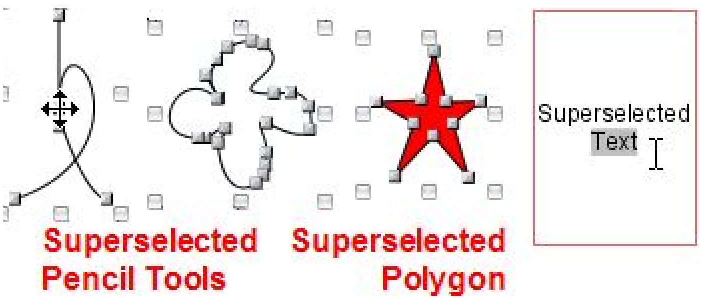
Selection and Alignment

Selection is done with the selection tool  on the tool bar.

Reporting has a "deeper" selection model than the FactoryPMI designer. Simple object selection is done by single clicking an object. "Selecting deep" is done by double-clicking to get into the report hierarchy. For instance, if you group two rectangles together, you can select the individual rectangles by double clicking "into" the group.

Superselection

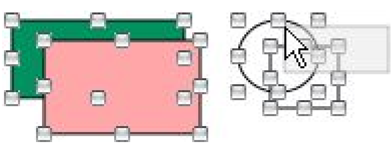
Superselection refers to an editing state that some shapes go into when double clicked. Text is the most common of these. When a text box is selected you can move and resize it. When it's super-selected, you can place the text cursor or select a range of characters and insert or delete text. The polygon and pencil are two other basic tools that support superselection.



Multiple Selection

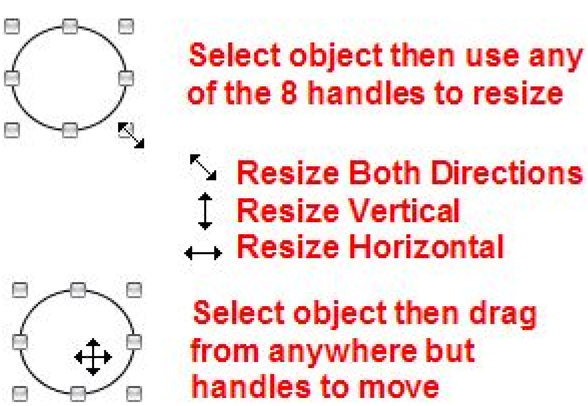
Multiple Selection can be done two ways:

- Clicking and dragging the mouse over a range of the report. Everything the selection rectangle touches becomes selected.
- Hold the shift key while making a selection or dragging a selection rect. Shapes hit by that action will be added or removed from the currently selected shapes.



Resizing and Moving objects

To resize or move an object first select it with a single click. To *resize* left click and drag one of the 8 resizing handles. To *move* the object, left click and drag anywhere on the object when it is selected. Both operations support shift dragging.



Alignment

Alignment is accomplished by selecting multiple objects, then choosing "**Make ...**" from the shapes menu or right click menu.

***Shapes* Menu Item**	**Function**
Make Row Top/Center/Bottom	Quickly align several shapes in a row, either by their top, center, or bottom border. Useful when shapes are of different heights.
Make Column Left/Center/Right	Same as above, but for columns, aligning their sides or center.
Make Same Size, Width, Height	Make several shapes the same width, height or both.
Equally Space Row/Column	Equalizes the distance between shapes horizontally or vertically.

Shift Drag

Holding the shift key while you drag shapes will constrain movement to: horizontal, vertical, or 45 degrees.

TIP	Getting used to selection and superselection is one of the most important concepts to master to become proficient with FactoryPMI Reporting.

Object Layout

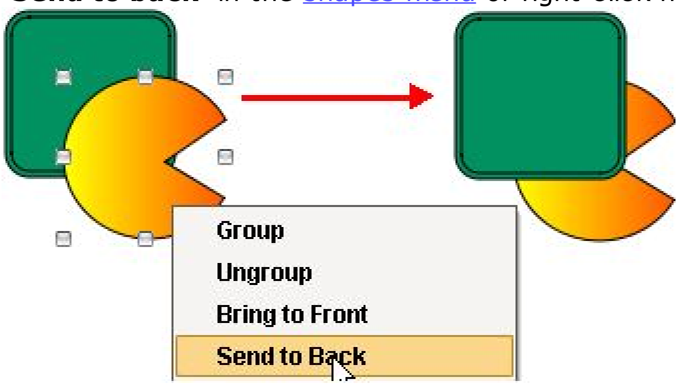
Object layout is an important aspect in creating a professional report. FactoryPMI Reporting uses a WYSIWYG (what you see is what you get) approach.

Headers and Footers

Creating headers and footers is just like creating any other set of objects on your report. There is no explicit header or footer section. The key is sizing and positioning your table around your header or footer. Each new page that the table creates will have that same header and footer. The idea extends to pdf based reports.This is illustrated in tutorial #1

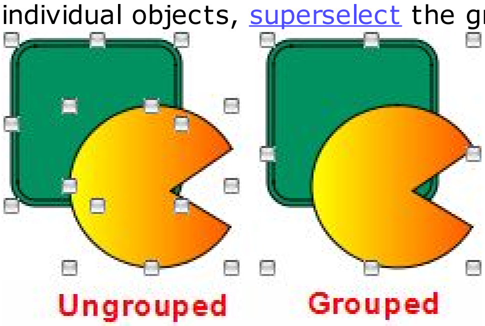
Z Order

Z order defines relative order of objects when they overlap. Simply select the object and click "**Bring to front**" or "**Send to back**" in the shapes menu or right click menu.



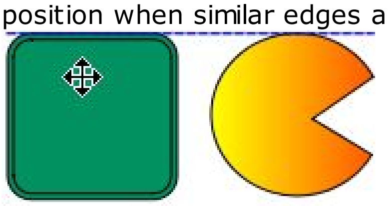
Object Grouping

Grouping makes a set of object behave as one with respect to: selection, moving, and resizing. To "drill down" to individual objects, superselect the grouped object.



Alignment

Alignment is simple. As you move an object around, the Report Designer will draw in a blue dashed line and snap to position when similar edges align. Below the top edge aligns.

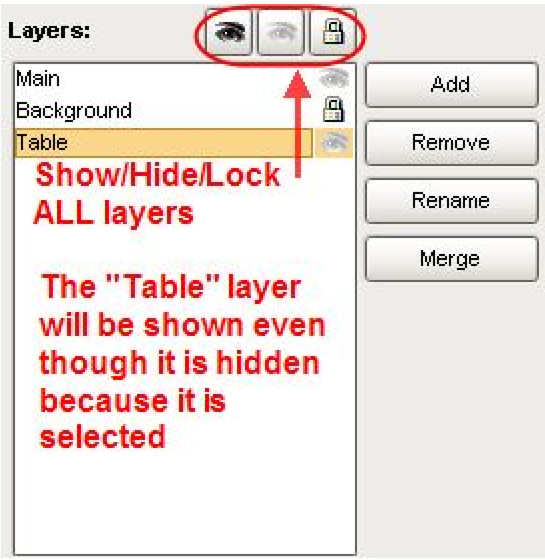


Layers

Layers are logical "layers" that take up the space of the entire screen, but contain a subset of the objects on it. They allow you to work on certain parts of your report independently of the rest.

- Selecting a layer, even a hidden one, will show it

- Show displays a layer and allows you to work on it
- Hide hides a layer and doesn't allow you to work on it
- Lock displays a layer, but doesn't let you select any objects on it



TIP	Another important aspect of layout is selection and alignment.

Text Editing

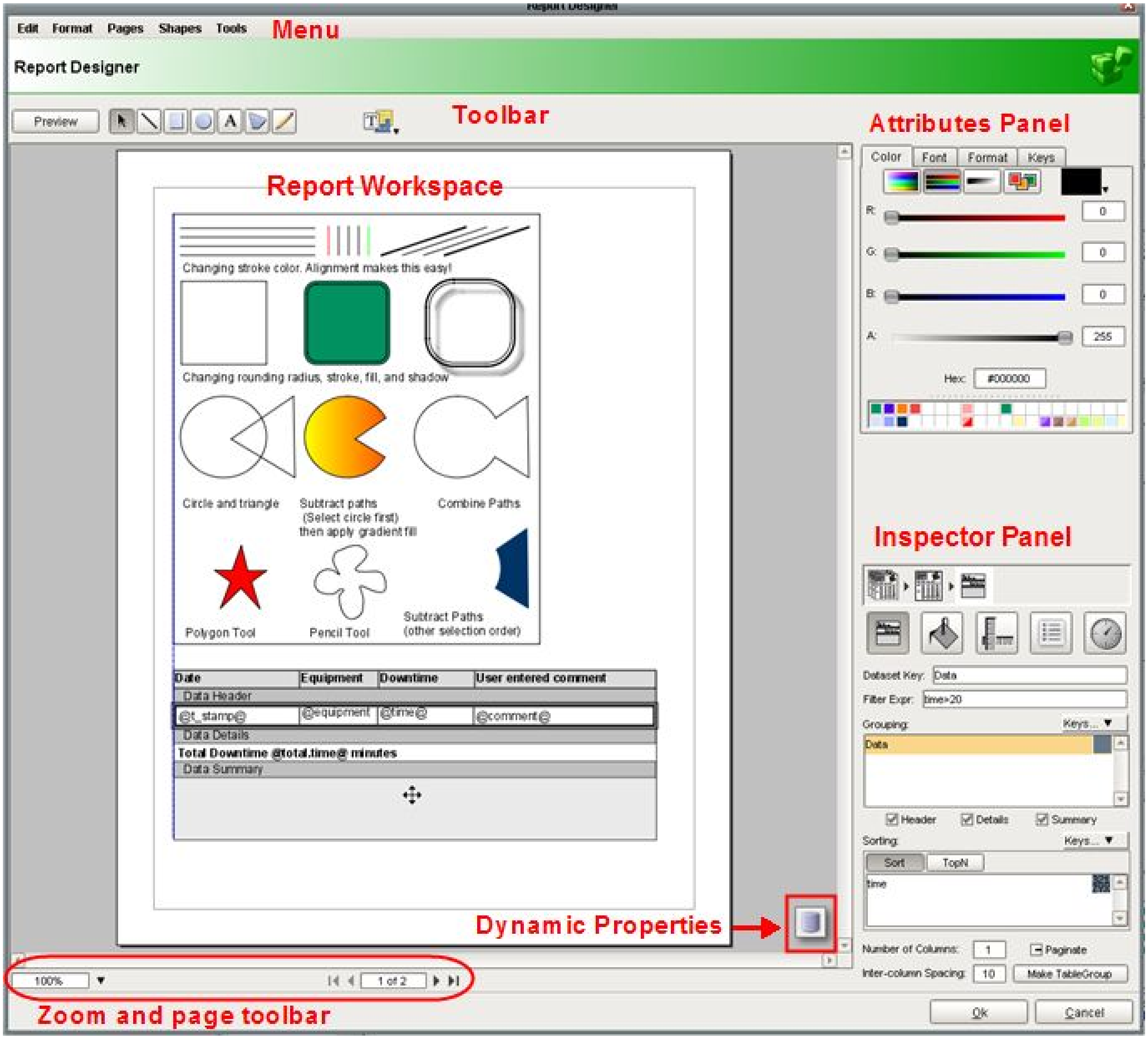
Text editing is pretty straightforward. A few things to know:

- Superselection is key here. Distinguish between selecting a text label versus superselecting the text itself.
- Text properties that are modified on the font attribute panel (color, **bold** (Cntl+B), *italics* (Cntl+I), font, size) apply to selected (highlighted) text. If you have an entire object selected prior to making a text property change, all text in that object will be modified.
- Properties that are modified on the text inspector panel such as: text alignment, shadows, fill and stroke, and transparency are object properties. Changes will usually affect all text in that object regardless of specific text selection.

TIP	Most text properties can be set in the Font Attribute Panel or the Text Inspector Panel. The notable exception is font color, which is set by highlighting text and using the Color Attribute Panel.

Report Designer

The Report Designer is the Customizer (Cntl+U) for the Report Viewer. It is the window where you create your reports.



Major Sections

- The menu provides various options, most for selected objects.
- The toolbar allows you to create shapes objects.
- The Report workspace is where you create your report.
- The Attribute Panel is where you modify common properties.
- The Inspector Panel gives you access to more specific object properties.
- Dynamic Properties bring FactoryPMI data into your report

Menu

Edit Format Pages Shapes Tools

The menu provides quick access to many common functions. It is divided into five sections:

- Edit
- Format
- Pages
- Shapes
- Tools

Edit

The *edit* menu provides functions like cut, copy and paste.

Menu Item	Function
Undo	Undoes the last action.
Redo	Re-does the last undo (assuming nothing was changed after the last undo).
Cut/Copy/Paste	Allows you to easily duplicate or import document elements using the system clipboard.
Select All	Selects all elements at the current level of selection (or all text, if editing a text field).

Format

The *format* menu is used for text formatting.

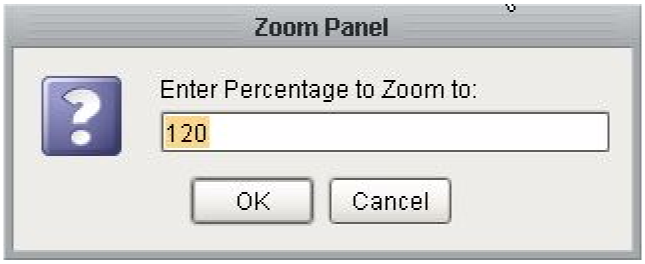
Menu Item	Function
Font Panel...	This selects up the Font Panel tab of the Attributes panel.
Bold, Italic, Underline, Outline	Modifies or unmodifies currently select text or text fields. This functionality is also available in the font panel.
Align Left, Center, Right	Aligns currently selected text or text fields to the left, center or right. This functionality is also available in the Text Inspector.
Subscript, Superscript	Modifies or unmodifies currently select text or text fields.

Pages

The *pages* menu allows you to add or remove pages to the report and change the zoom level

Menu Item	Function
Add Page	Adds a page to the current open document, after the currently selected page.
Add Page Previous	Adds a page to the current open document, before the currently selected page.
Remove Page	Removes the currently selected page in the current open document.
Zoom In/Out	Increases/decreases document zoom by 10%.
Zoom 100%/200%	Zooms to the specified percent of actual document size.

Zoom Toggle Last	Zooms to the last specified Zoom.
Zoom...	Brings up a zoom panel that allows you to exactly specify a zoom as a percentage of actual document size.



Zoom Panel Dialog

Shapes

This *shapes* menu allows you to modify the layout of objects in a report

Menu Item	Function
Group/Ungroup	Allows you to merge the currently selected shapes into a single shape for convenient management. Contained shapes are still accessible, via double-click super-select. Ungroup separates grouped shapes.
Bring to Front/Send to Back	All shapes have an order on the page that determines what is drawn on top when two shapes overlap. These options allow you to alter that order.
Make Row Top/Center/Bottom	Quickly align several shapes in a row, either by their top, center, or bottom border. Useful when shapes are of different heights.
Make Column Left/Center/Right	Same as above, but for columns, aligning their sides or center.
Make Same Size, Width, Height	Make several shapes the same width, height or both.
Equally Space Row/Column	Equalizes the distance between shapes horizontally or vertically.
Group in Switch/3D Shape	This feature groups selected shapes in a Switch Shape, which has the same features as Table Row Versions. It's a powerful way to conditionally provide a different look for a specific element.
Move to new layer	Creates a new page layer with the currently selected shapes.
Combine/Subtract Paths	Takes multiple overlapping shapes (such as a rectangle and an oval) and combines them into a single shape using the combined paths. A powerful tool to construct complex shapes.
Convert Into Image	Converts the selected shape into an image. Be sure to group shapes first if you want to convert multiple shapes into a single image.

Tools

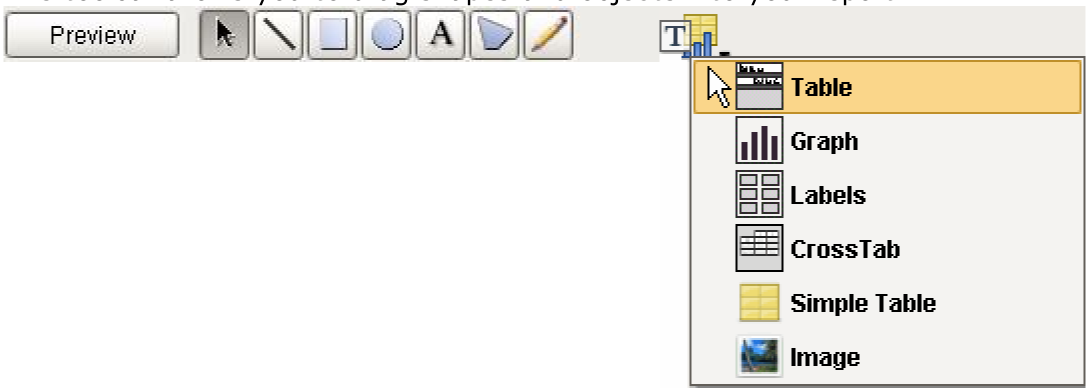
The *tools* menu contains layout tools

Menu Item	Function
Color Panel	Selects the color tab in the Attribute Panel.
Font Panel	Selects the font tab in the Attribute Panel.
Formatter Panel	Selects the format tab in the Attribute Panel.
Keys Panel	Selects the keys tab in the Attribute Panel.
Toggle Rulers	Adds rulers to the page borders to assist in precise layout.

Image Placeholder	Adds an empty image placeholder object to the document, which can be positioned, sized and configured with a substitution key.

Reporting - Toolbar

The toolbar allows you to drag shapes and objects into your report



Toolbar Icons

Icon	Name	Description
Preview	**Toggle Preview/Edit Mode**	Toggles between Preview and Edit modes. This is equivalent to going between Preview and Design mode in the FactoryPMI designer. **Edit mode** will allow you to make changes to the layout of the report. **Preview mode** will populate the report with data and show you what it will look like in the runtime.
	Selection Tool	Default tool. Clicking on objects with the selection tool will select them for movement or modification.
	Line Tool	Click and drag to create a line.
	Rect Tool	Click and drag to create a rectangle. The Rect inspector will allow you to set rounding radius.
	Oval Tool	Click and drag to create an oval. The oval inspector will allow you to select sweep and start angle.
	Text Tool	Click and drag to create text. Click for more on text editing.
	Polygon Tool	The polygon tool lets you click points that will be joined with straight lines. Alternatively, you can click-drag-release to position line segments interactively. If you hold down the alt key while adding points the polygon tool will behave like pencil for added segments. Editing stops under the following conditions: clicking the same point twice, clicking close to the start point or clicking a new tool in the tool bar (like the selection tool).
	Pencil Tool	The pencil tool lets you click and draw free-hand path segments, automatically smoothing the curve on mouse up. If you hold down the alt key, it will behave like polygon for added segments. Editing stops under the same conditions as polygon.

Add Shapes Button

Icon	Name	Description
	Table	Arguably the most powerful Reporting feature. Tables will occupy a fixed size on the screen but create as many pages in the report as the dataset requires. Useful for a downtime report that may cover one day or six months, for example.
	Graph	The graph is a dynamic bar or line graph. It is simple, yet conveys much information.

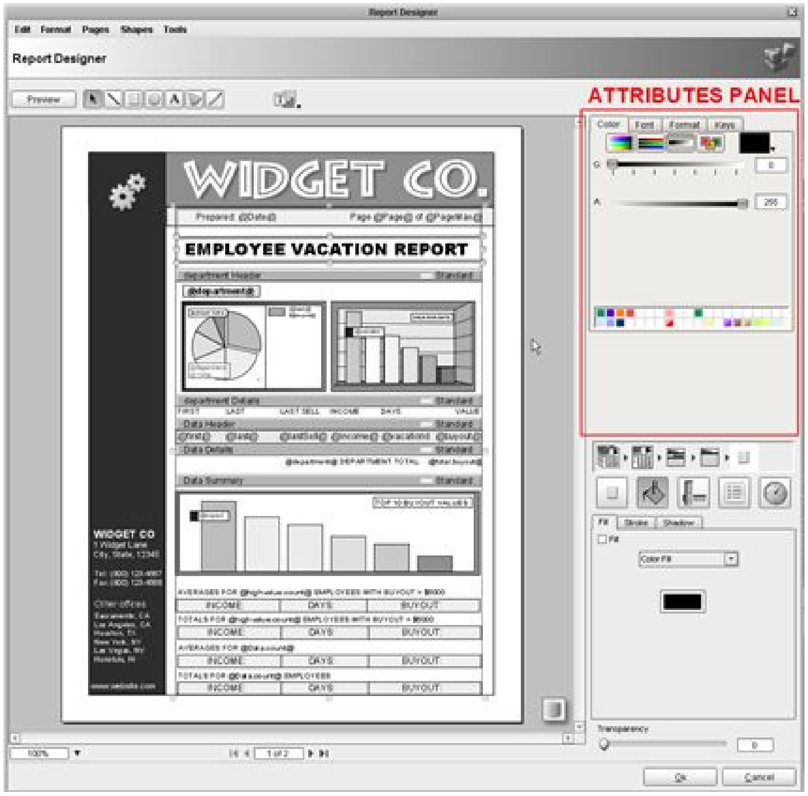
	Labels	Labels are printable labels that are compatible with standard Avery label sizes.
	Crosstab	Crosstabs summarizes a cross section of data, such as total downtime by both equipment and location.
	Simple Table	The Simple table is a table of a fixed size that doesn't support DataSets. It is easy to work with and ideal if you don't need the flexibility of a table.
	Image	Images make your report look good.
	Image	Image Placeholders provide different images based on conditions.

TIP	Know your basic shape tools and their properties. They can be used to produce professional reports!

Attributes Panel

The attributes panel is the top right panel on the Report Designer that is used to modify common attributes for simple objects, especially text.

Single click to select your object then make changes in the attributes panel. Often times you will have to double click to drill down to the simple object or property that you want to modify.

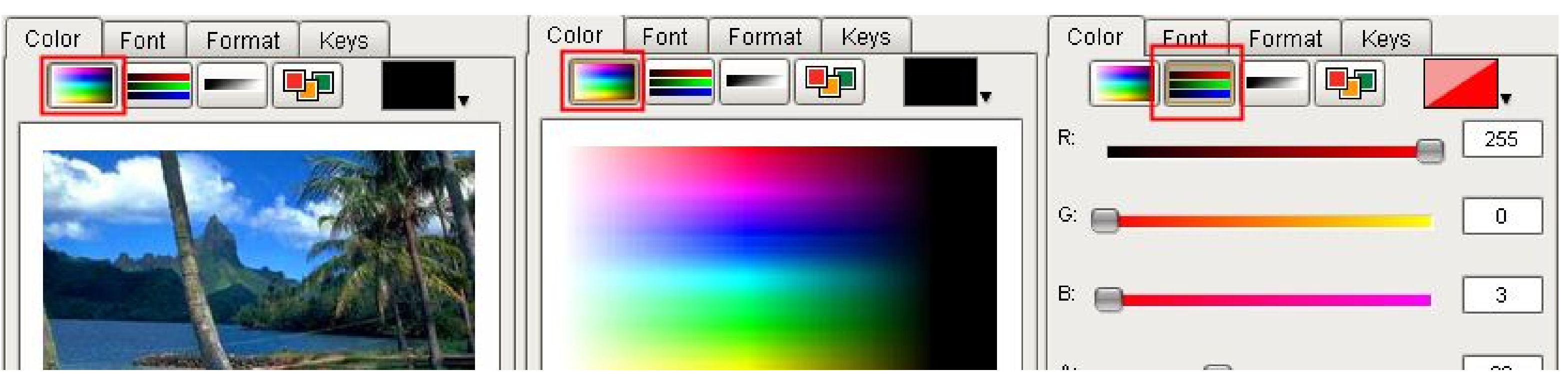


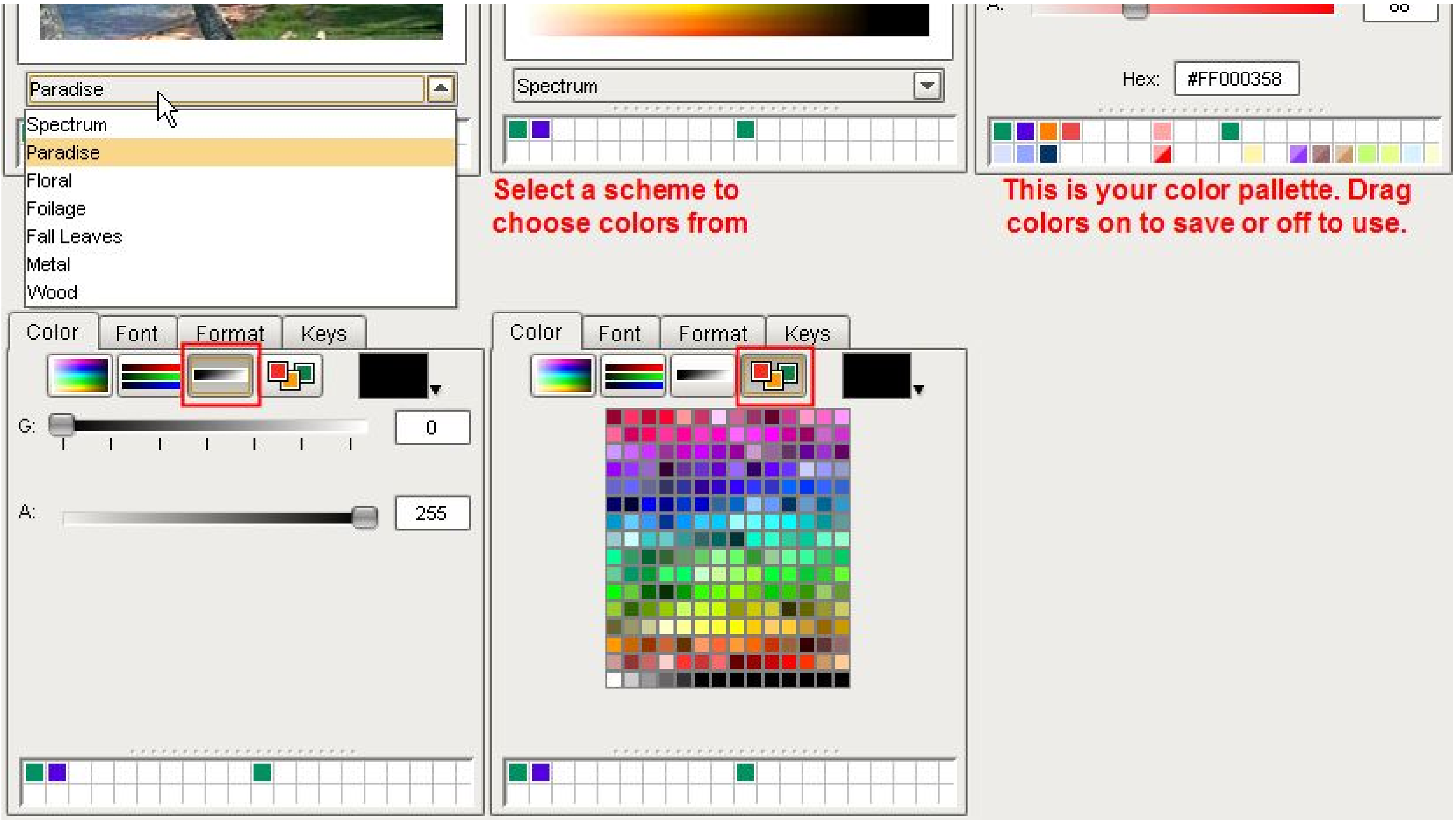
Color Tab

The *color tab* is used to change any color in your report. Suppose you wanted to change the fill (background) color of a text label. There are several ways to accomplish this:

1. Left click the label to select it. Click a color on the attribute panel. You'll notice that fill property gets enabled and the background color set to your choice.
2. Select the label. Click on the colored square under the fill tab of the inspector panel to select the color. Choose a color on the attribute panel.
3. Select the label. Drag a color down from the color panel to the colored square under the fill tab of the inspector panel.

All of these changed the fill color. To change the font color of that label you would double click the text label, highlighted the text, then changed the color. The key is getting used to the selection model to change the color of the desired property.

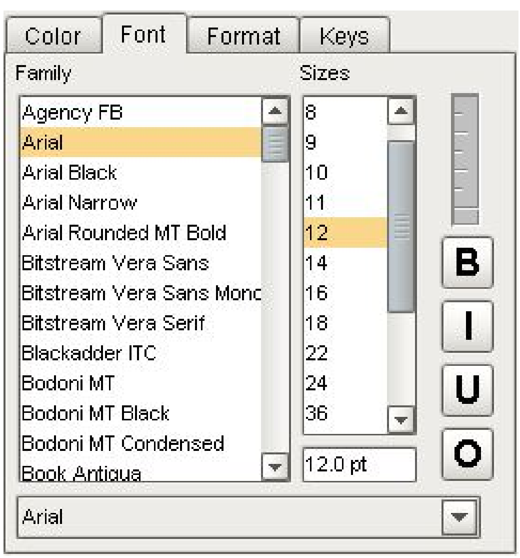




Font Tab

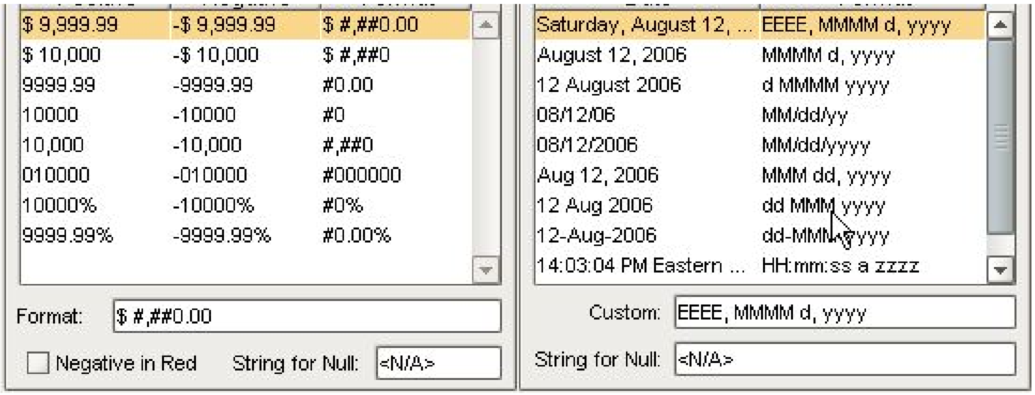
The *font tab* is used to change the family, size, and options of fonts. Selection tends to be much more forgiving since there are relatively few font properties. For example, selecting a label is the same as double clicking that label then highlighting **all of the text**, with respect to the font panel.

To change the color of text, highlight it, then go to the color tab.



Format Tab

The *format tab* is used to apply formatting to dates and numbers. Highlight desired text and choose formatting. Dates are formatted like the expression **dateFormat** function (shown below). **None** removes formatting.



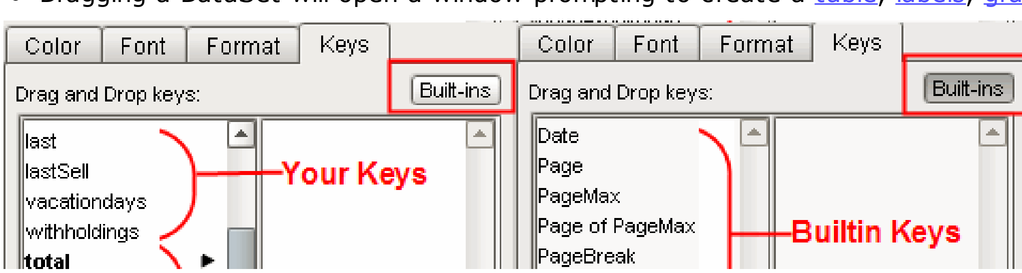
For the following table, assume the Date is 7/8/2005 3:05:00 PM (July 8th, 2005).

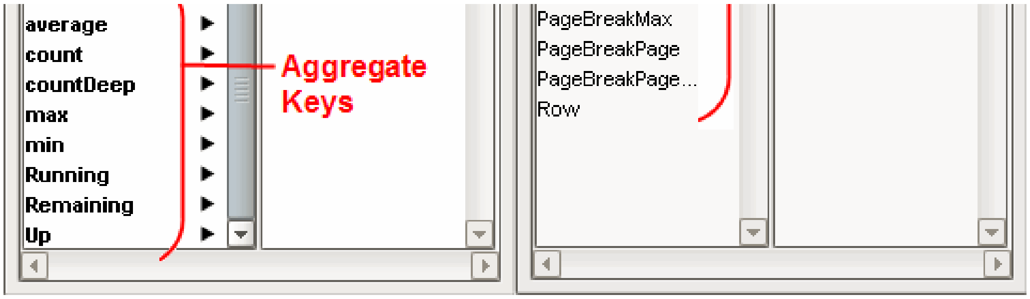
Date Pattern Components		
Character	**Function**	**Example**
M	Month	7
MM	Month, forced 2 digits	07
MMM	Name of month, abbreviated.	Jul
MMMM	Name of month, full	July
d	Day of the month.	8
dd	Day of the month,forced 2 digits.	08
E	Day of the week, abbreviated.	Sun
EEEE	Day of the week, full.	Sunday
yy	Year - abbreviated.	05
yyyy	Year - Full	2005
H	Hour of the day (0-23)	15
h	Hour of the day (1-12)	3
m	Minute	5
mm	Minute, forced 2 digits.	05
s	Seconds	00
a	AM/PM marker	PM
z	Time zone, abbreviated.	PST
zzzz	Time zone, full	Pacific Standard Time

Keys Tab

The *keys tab* is a convenience that displays your data and builtin functions. Clicking "Built-ins" will toggle between user data and builtin functions. The typical use of the Keys Tab is dragging keys into your report. Here are a few examples of how that could work:

- Dragging *last*, a string data key, to your report will create the text label, **@last@**
- Dragging *last* to text in a selected text label will add in the text **@last@**.
- Dragging a DataSet will open a window prompting to create a table, labels, graph, or crosstab.

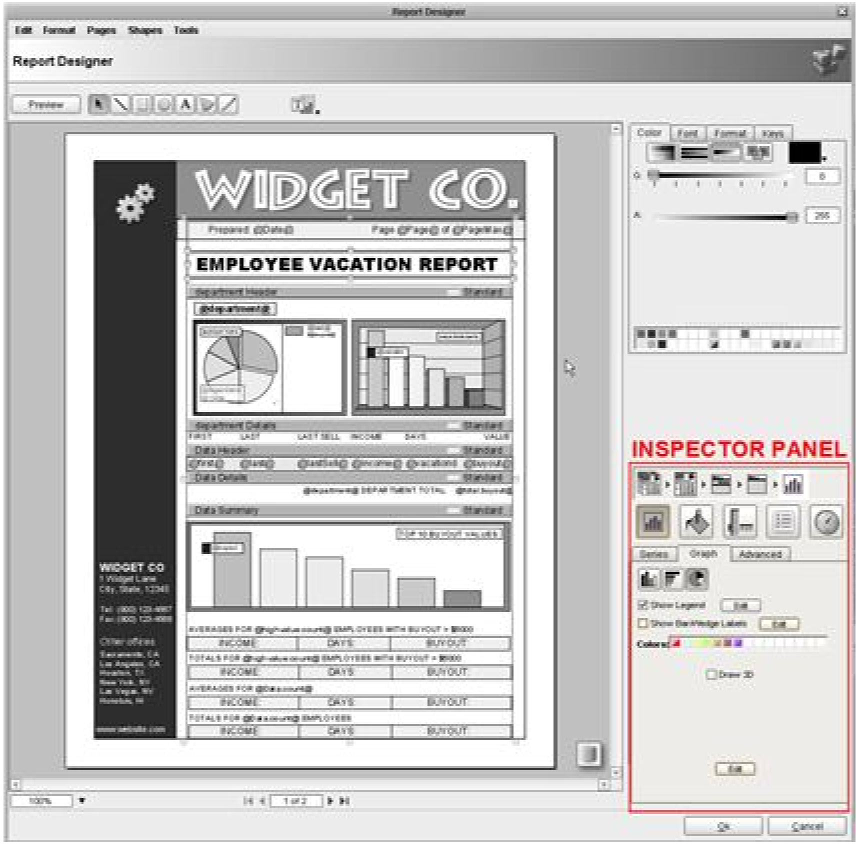




TIP	Get to know the attribute panel. Most shared properties reside here. The only other panel to know is the inspector panel, where more complex or object specific settings reside.

Inspector Panel

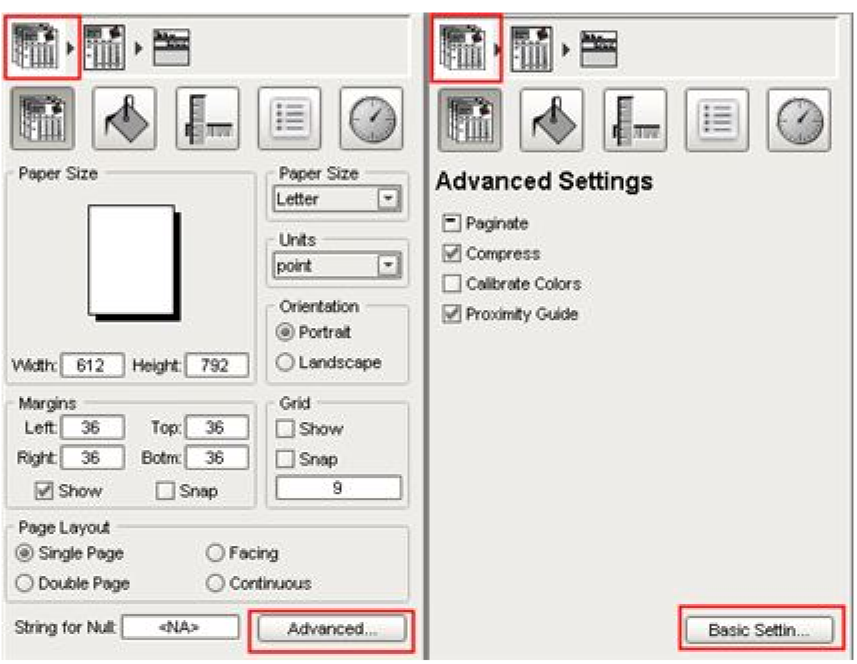
The inspector panel is the bottom right panel on the Report Designer. It is used to modify object attributes.



Tutorial #2 example report.

Document Inspector

The Document Inspector is where you set your page layout, paper size, margins, and other top level properties.



Page Inspector

The Page Inspector deals with document layers. "Layers" are logical grouping containing anything between no objects and every object that takes the space of the whole report. For example, you could create a background layer that contains borders and graphics. You would then create a main layer that is the bulk of the dynamic report. When working on one layer, you could make the other invisible. You can also lock a layer once you're finished with it.

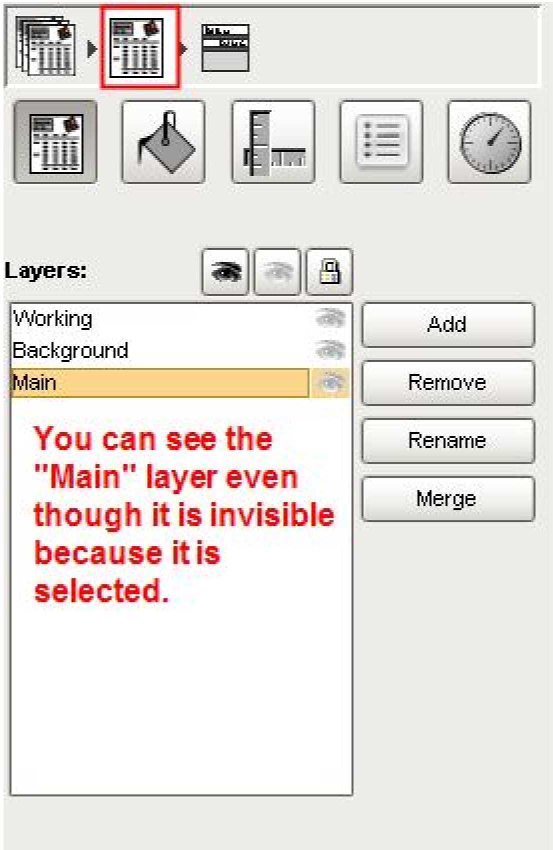
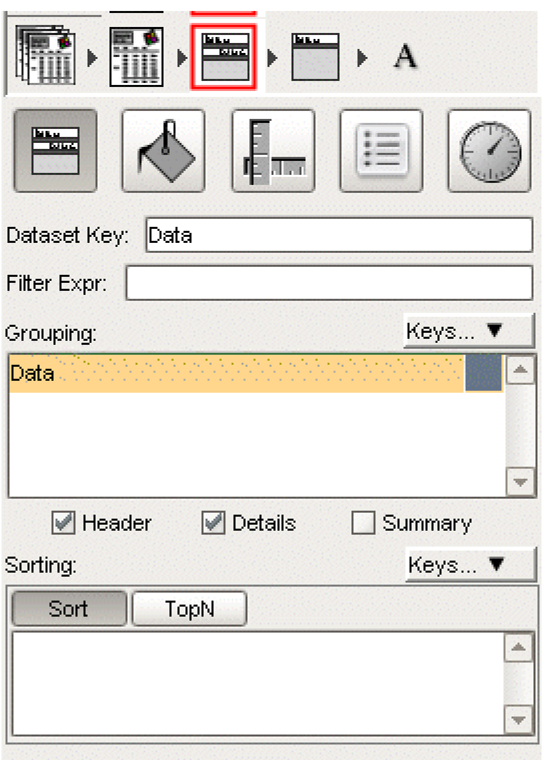
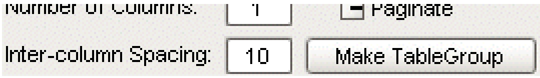


Table Inspector

The Table Inspector defines the dataset, sorting, grouping, and filtering for tables. It is where you choose to display a table's header, detail, and summary.

The Table Inspector can modify data processed by the table, as well as the general look of it.

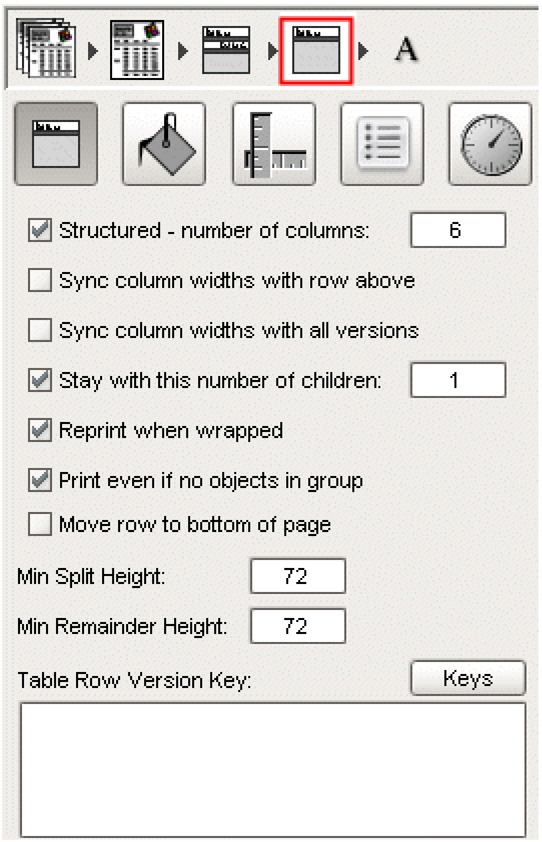




Paginate - Has three setting (Off , On , N/A) option that determines whether or not a table will use page breaks. Paginating tends to be useful for pdf files, not paginating tends to be good for Flash and CSV files. Typically leave this setting alone.

Table Row Inspector

The Table Row Inspector defines properties of [rows](#) in a table. This includes all [versions](#) of the [header](#), [detail](#), and [summary](#) rows, as well as specifying the [version key](#) expression and printing options. It is most easily accessed by [superselecting](#) the table, then selecting a table row.

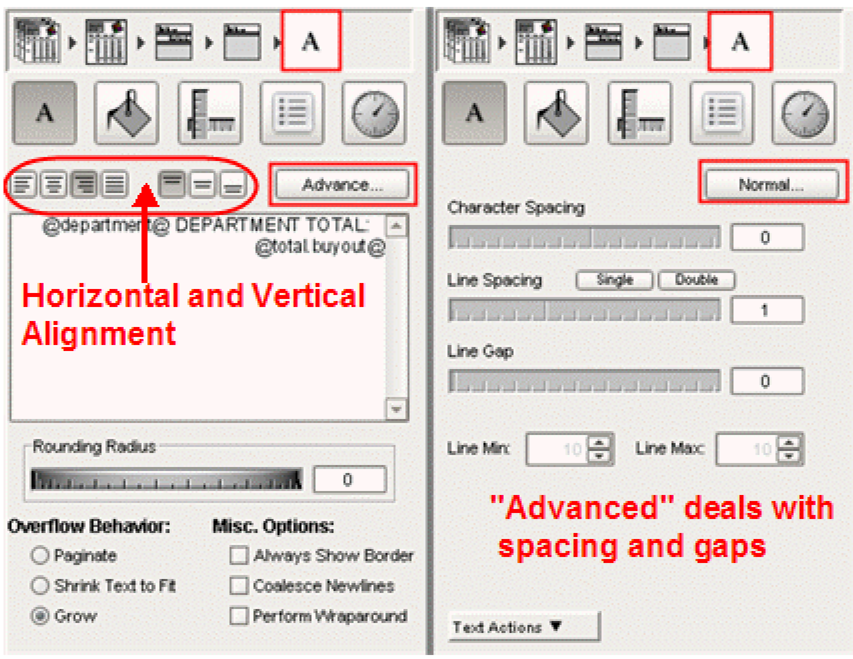


Row Property	Function
Structured Switch/Column Count	Sets row to unstructured or defines number of columns. This can also be done with table icons.
Sync with parent/alternates	With structured tables, it's often convenient to have column resizing be reflected in the row immediately above the current row (the parent) or with a table row's different versions (alternates). Once enabled, individual column resizing will affect the corresponding parent or alternate row width. This is useful for synchronizing detail/header row changes.
Stay with this number of children	This is the heart of widow/orphan control. By default, a row is guaranteed to have at least one child in its group on the same page. This prevents such rows from being printed by themselves, which can be confusing. Increase this number for additional family bonding. If it exceeds the number of objects in a group, the group will never be broken across a page boundary.
Reprint when wrapped	When data overruns the bottom of the page and starts on a new page, upper level grouping details and headers are reprinted to retain context. Occasionally this doesn't makes sense. Select the row and click this switch to suppress this behavior. An alternative is to configure a [Reprint version](#).
Print even if no	By default headers and summary rows for empty lists are suppressed. If you want an

objects in group	indication of the missing data turn this switch on.
Move row to bottom of page	Normally the Summary row will share a border with the last row on the table. **Move to Bottom** will move it down slightly so that it's always resting on the bottom border of the page. This is commonly used with the Running Summary feature.
Min Split/Remainder height	An advanced form of widow/orphan control is to be able to control how an exceptionally tall table row will break across a page (usually only the case when a large text block is involved). By default rows will only be split when at least an inch (72pts) was available on the first page (**min split height**) and at least an inch will be carried over to the successive page (**min remainder height**). Most table rows will never use these settings. If you prefer to have table rows use all of the potential page space and don't care about trying to keep related text on the same page, you would set both of these to 10pts. If you never want a row to split, set these to 999.
Table Row Version Key	Allows you to configure different looks for the same table row based on some condition to provide visual hints. The version key expression should return a string that is the name of a version that you've defined. Details here or example here.

Text Inspector

The Text Inspector is where you specify text alignment. More details under text editing. You can use this larger textbox to edit text instead of making text changes directly on objects.

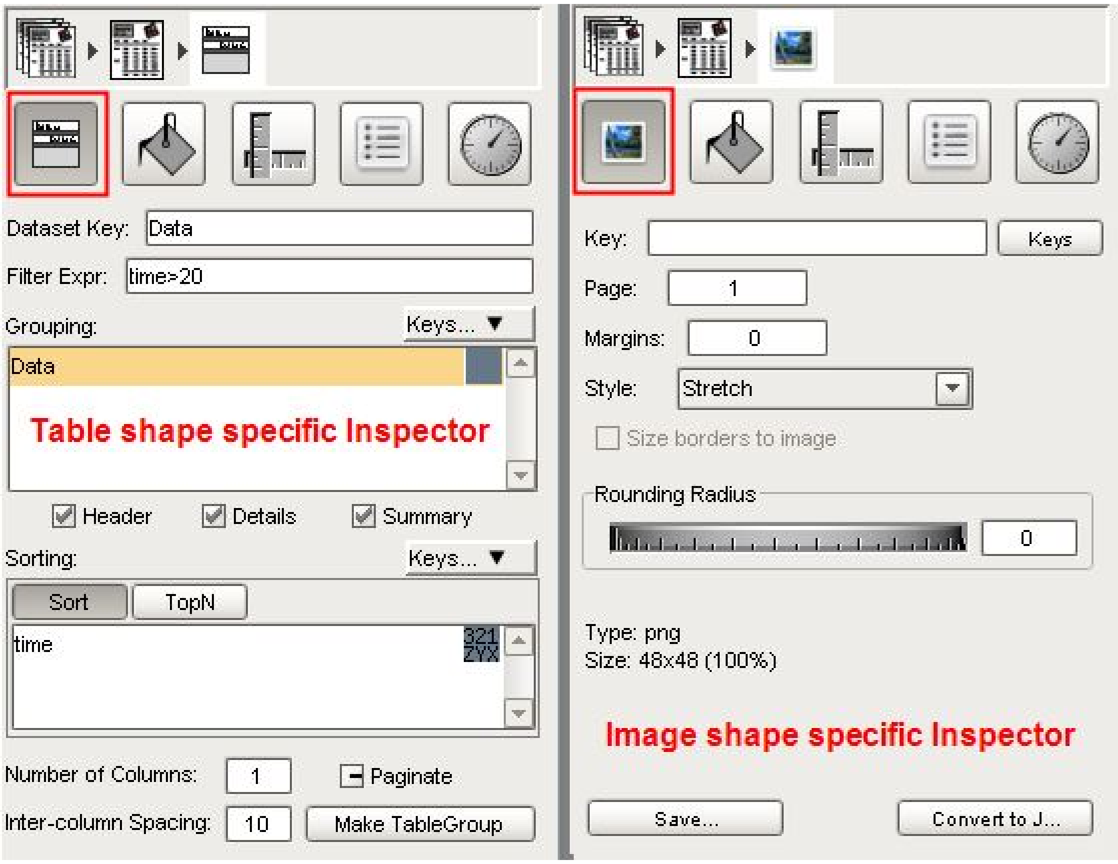


Option	Function
Rounding	This thumbwheel allows you to set the rounding radius for the text border. It's immediately reflected in the editor window.
Overflow Behavior	Text can be set to **paginate** for form letters, **shrink text to fit** for static text boxes that may receive arbitrarily long text, or **Grow** for text fields in table rows (which can grow to accommodate large text blocks).
Always Show Border	Draws a gray border around text even when not selected. Sometimes useful as a visual cue while editing, without marking generated reports.
Coalesce Newlines	Coalesced newlines will make sure text uses the minimum lines necessary. Useful for substituted data that might contain missing keys, eg, "@name@\n@address1@\n@address2@\n@phone@\n@fax@".

Tab Stops	If you turn on rulers for the editor window (Tools->Toggle Rulers menu), you will notice that it shows tab markers while editing text. These can be dragged and reset to change the tab stops of the text field.

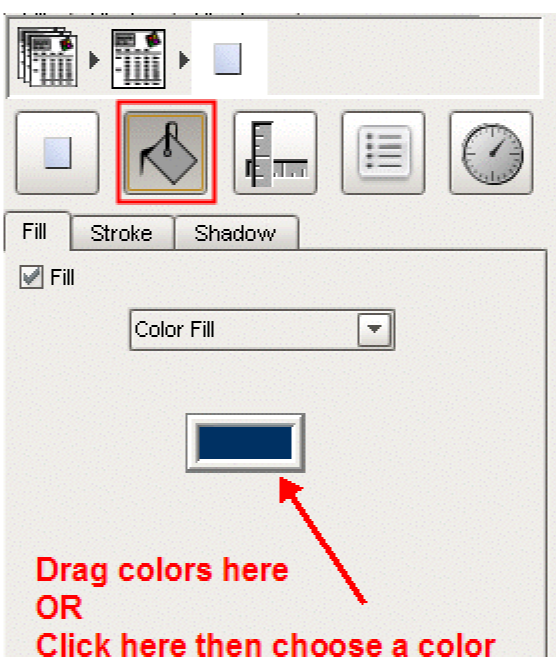
Shape Specific Inspector

The Shape Specific Inspector changes depending on the selected object. It often takes the form of the other inspectors listed on this page. Some objects have custom shape specific inspectors. The left example below is the shape specific inspector for a table, which happens to be the table inspector. The right inspector is the custom shape specific inspector for an image.



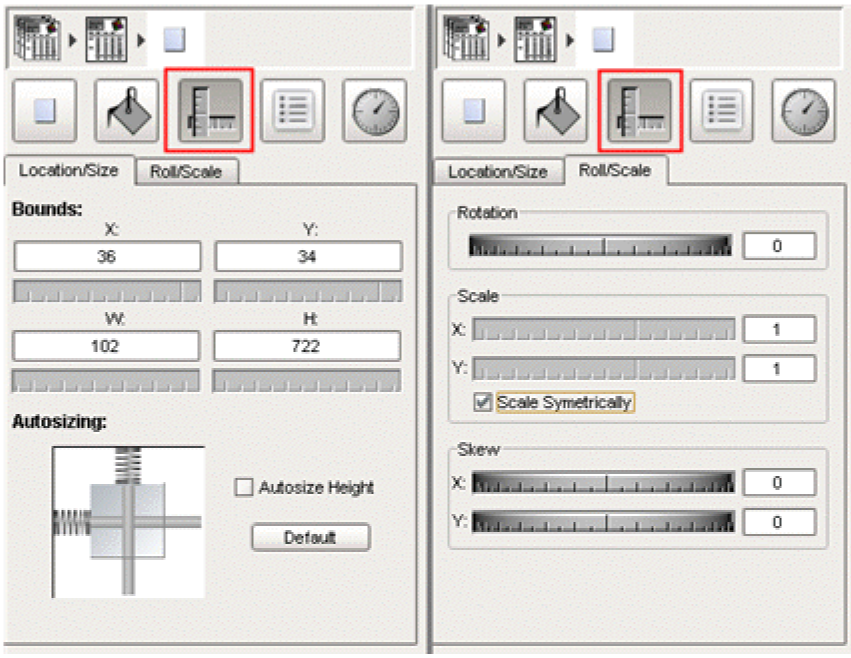
Fill & Stroke Inspector

The Fill & Stroke Inspector is where you set background (fill), outline (stroke), and shadow.



Location & Size Inspector

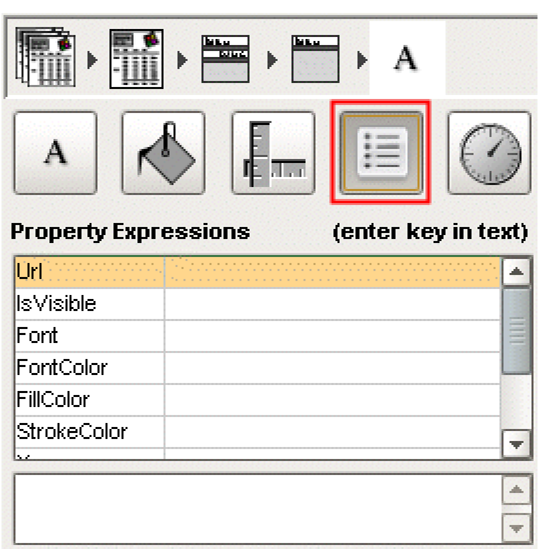
The Location & Size Inspector allows you to see actual positioning, set auto-sizing, and change properties such as: rotation, scale, and skew. Auto-sizing works by clicking the different regions in the auto-size boxes to draw or erase "springs".

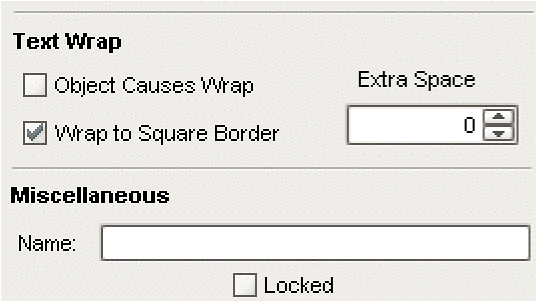


Roll, Scale, & Skew Inspector

The Roll, Scale, & Skew Inspector is a powerful panel that lets you set properties based on expressions (string or number based). You can do things like:

- Use **isVisible** property to display an image of a fancy checkbox or exclamation in the row of a table.
- Scale the **width** property of a rectangle with a gradient color within a table to indicate progress.
- Conditionally change **fontColor** or **fillColor**
- Dynamically position an object around with **X** and **Y** properties.

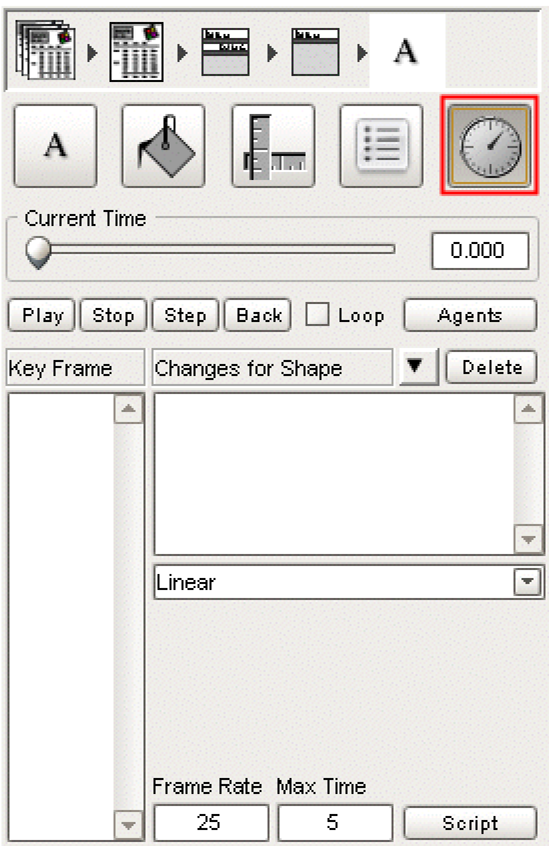




See Property Expressions to illustrate their dynamic use.

Animation Inspector

The Animation Inspector is used to set up animation, which works, but will not be useful unless Reporting enables Macromedia flash based reports. You set up snapshot times and the report will morph the scene from one time to the other.



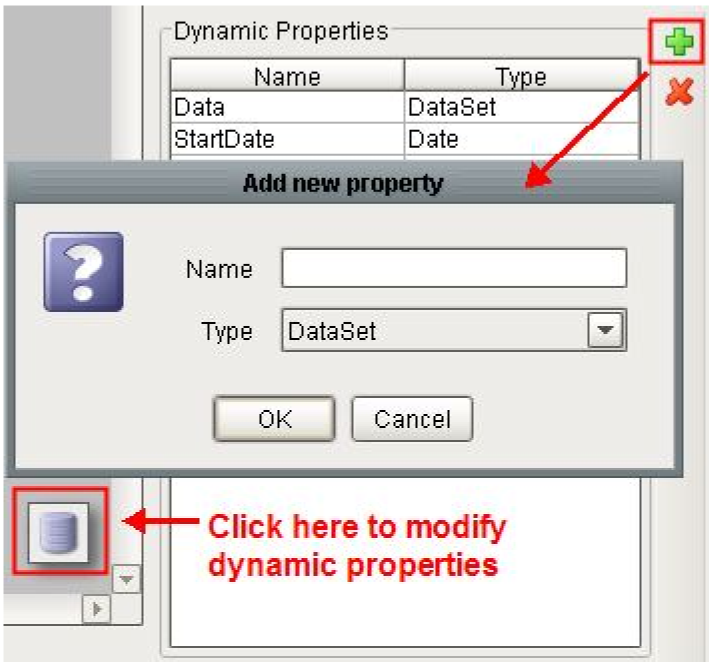
TIP	The inspector panel varies on an object by object basis. If you have trouble changing a property on a complex object, chances are it's here. Try clicking on different parts of the object then going through the Inspector Panel.

Dynamic Properties

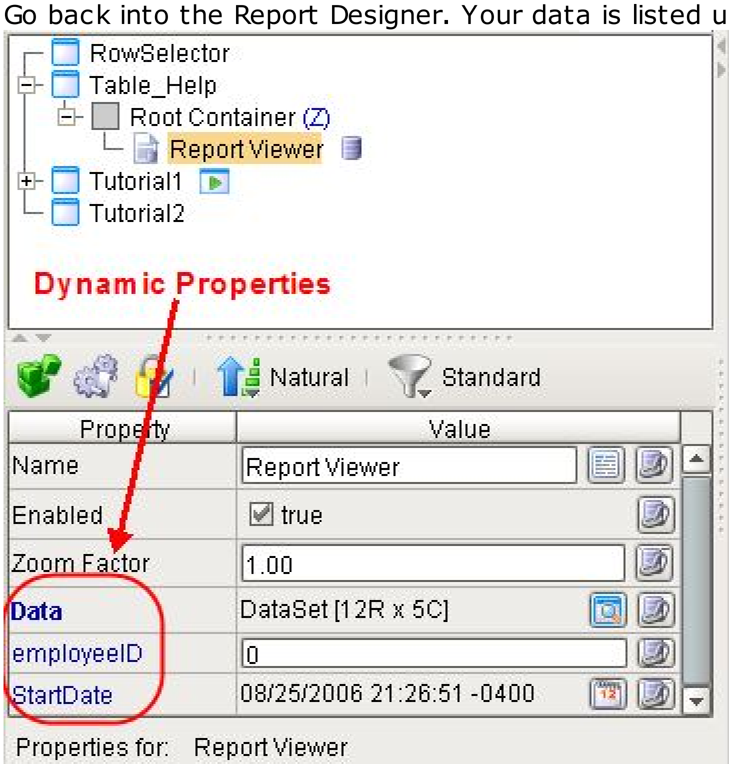
Dynamic Properties are user defined variables and DataSets attached to a report viewer. They allows your report to be populated by data within FactoryPMI. This paradigm is powerful because it gives you the flexibility of FactoryPMI features: use any database connection for SQL queries, expression functions, bindings, etc. This also allows selection changes within FactoryPMI to automatically update your report's data. Reporting dynamic properties work similarly to the dynamic properties of a graph or container.

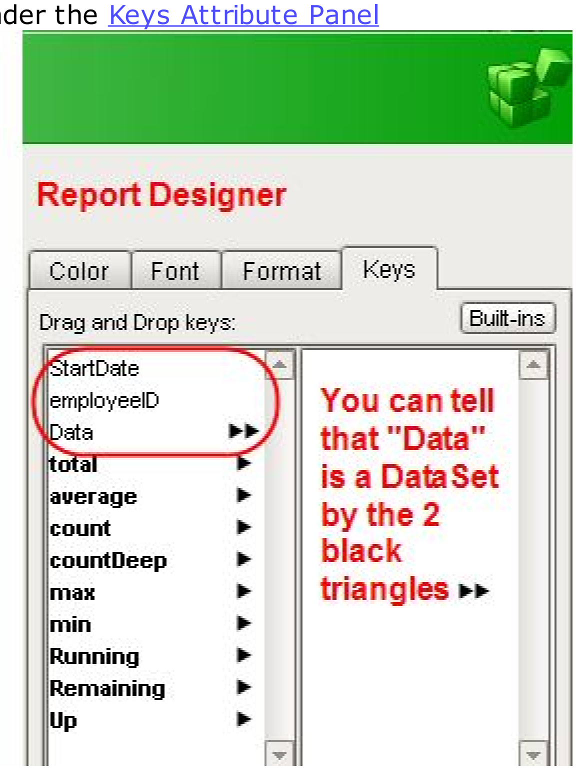
Report Viewer Dynamic Properties

1. Define dynamic properties in the Report Designer by clicking the database icon in the lower right hand corner of the screen.



2. Click "Ok" to get out of the Report Designer and back into the FactoryPMI designer.
3. Populate your dynamic properties as you would any other FactoryPMI properties.
4. Go back into the Report Designer. Your data is listed under the Keys Attribute Panel





Dynamic Property values are introduced into the report as "keys"

5. Your keys may now be referenced in the Report. For example, **@StartDate@** would display **08/25/2006**. It can be formatted however you wish via the Formatter attribute panel.

TIP	Dynamic Properties bring data into your report in the form of keys. To reference these keys, see substitution keys, a fundamental aspect of reporting

Substitution Keys

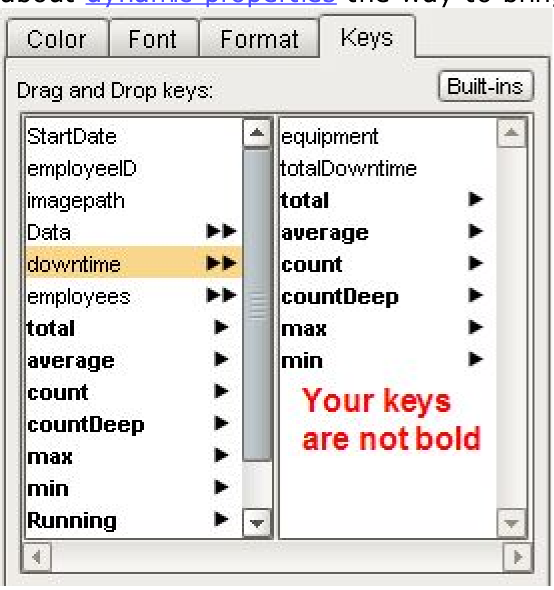
The most important part of any reporting system is data substitution. FactoryPMI Reporting uses a familiar mail-merge paradigm, allowing the user to intermingle keys with static text. Keys are delineated by "**@**" symbols, for example: @Date@ or @myVariable@. An example of mixed keys and text, might be "@Page@ of @PageMax@", perhaps resulting in the text "1 of 10".

An interesting thing about keys is that they can be @anything@! You can type any string between two "@" symbols and the Reporting engine will treat it as a key. At run-time it evaluate the key to your dynamic property or a built in key. The syntax for keys follows the rules of Java expressions, described here

If a key cannot be evaluated it will return the **String for Null** property on the document inspector (set to "N/A" by default) .

Your Keys

Your keys are the most important data in the report! Browse through them with the Keys Attribute Panel. Read more about dynamic properties the way to bring data into the report.



Builtin Keys

The following builtin keys may be typed or dragged from the keys panel

Menu Item	Function
Date	The current date/time. Can be formatted in the formatter panel
Row	The current row number (only in tables).
Page	The current page
PageMax	The total number of pages in the generated report
PageBreak	The number of explicit page breaks encountered
PageBreakMax	The total number of explicit page breaks in generated report
PageBreakPage	The number of pages since last explicit page break
PageBreakPageMax	The total number of pages in current explicit page break

Formatting Keys

Keys that return: dates, currency, or numbers can be formatted by highlighting then using the formatter.

Array Indexing

You can reference an individual object in a list using standard array indexing syntax (brackets) like this: "@Data [0].firstName@".

Aggregates (totals, min/max, average, count)

The Keys Browser contains a list of built-in keys at the bottom of any given list: **total**, **average**, **min**, **max** and **count**. These allow the user to easily specify aggregate calculations on a set of objects. Suppose we want to see @Data.total.revenue@ or the @data.min.runtime@ or perhaps just @data.count@. When performing an aggregate calculation on the objects in a table the DataSet **Data** is set as the **Dataset Key** so you can use @total.revenue@ instead of @Data.total.revenue@.

The "total2" key

An aggregate calculation will result in null if any of the individual values are null (rather than return a value that is technically incorrect). You can work around this by implementing a derived method that returns a default value if the original attribute is null and aggregating using that key/method. Also, most of the aggregates contain a second version (**"total2"**) that assume that null is equal to zero.

The "count" and "countDeep" keys

The **count** keys tell us how many objects are in a given list or group. This is most commonly used for tables with one or more levels of grouping. If, for instance, you have a table of Movies grouped by their studio and you add the @count@ key to the studio details, it will display the number of movies for each studio. So it might make sense to have a text field with "@studio.name@ has released @count@ movies" (Warner Brothers has released 15 movies).

The **count** key only counts the next level of grouping. If you have multiple levels of grouping and want to count all the root entities use the **countDeep** key. Suppose you have movies grouped by their category and their studio, and want to display a top level summary. You could use: "@studio.name@ has released @countDeep@ movies in @count@ different categories" (Warner Brothers has released 36 movies in 7 categories).

Heritage Keys (Running Totals, percentage totals)

There is an additional set of keys in the Attributes Browser which are used to access upper level groups: **Up**, **Running**, **Remaining**. @Up.count@ would tell us how many objects are in the current level of grouping.

The text field "Row @Row@ of @Up.count@" might show "Row 1 of 5".

By doing some simple arithmetic and using the "Up" key, we can calculate a percentage total: "% Total: @revenue/Up.total.revenue@"

The **running** key references a virtual array containing all of the objects processed thus far in a lower level grouping. This is useful to get a running total. For example, in a ledger: "Credit/Debit: @amount@ Current balance: @Running.total.amount@"

The **remaining** key is conceptually the same, but results in a virtual array of remaining objects.For example: "Credit/Debit: @amount@ Remaining Activity: @Remaining.total.amount@"

TIP	Check out substitution keys - expressions, operators, and functions for **even more** substitution keys!

Substitution Keys - expressions, operators, and functions

Key Expressions

You can type in expressions within the "@" symbols to perform calculations on the keys. Here are the operators in order of precedence.

Operator	Function	Example
Parenthesis	(expr) Nested expressions	Any portion of a Key Chain can be enclosed with parenthesis to guarantee precedence.
Multiplicative	*, /, % Multiply, divide, modulo	These are the most common and intuitive operators. You might want to display @quantity*price@ in an invoice line-item or calculate a percent like this @profit/revenue*100@.
Additive	+, - Add, subtract	See **multiplicative** above
Relational	>, <, >=, <= Greater-than, less-than, greater/less-than-equal	These are most useful for conditionals: @amount>=0? "Credit" : "Debit"@ or @name=="this"? "that" : name@.
Equality	==, != Equal, not-equal	See **Relational** above
Logical	AND &&	These operators make it possible to test multiple conditions: @revenue>100 && budget<50? "Winner!"@ or @name=="Jack" || name=="Sam"? "Good Name!"@.
Logical	OR ||	See **and** above
Conditional	? : If/then - with form "expr? true_expr : false_expr"	Provides IF/THEN/ELSE expressions. Note: a false expression is optional. 'null' will be evaluated to false and non-null as true. You can provide null substitutions like this: @name? name : "(None provided)"@. You can also nest conditionals for more conditions. For example, @age>=21?"Adult":(age>12?"Teen":"Child")@.
Assignments	=, +=	For the brave, you can create temporary variables for use in a report. Most of the functionality you might use this for is covered in more intuitive ways (such as the Running key), but it is possible to define a variable in a header row: @revTotal=0@ and update it in details rows @revTotal+=revenue@.

Math Functions

The following functions return floats.

Menu Item	Function
floor(float)	Round input down to the nearest whole number.
ceil(float)	Round input up to the nearest whole number.
round(float)	Round input to the nearest whole number.
abs(float)	Returns the absolute value of the input (if number < 0 return number * -1).
min(float, float)	Returns the input number with the least value.
max(float, float)	Returns the input number with the greatest value.
pow(float, float)	Returns first number to the second number power.

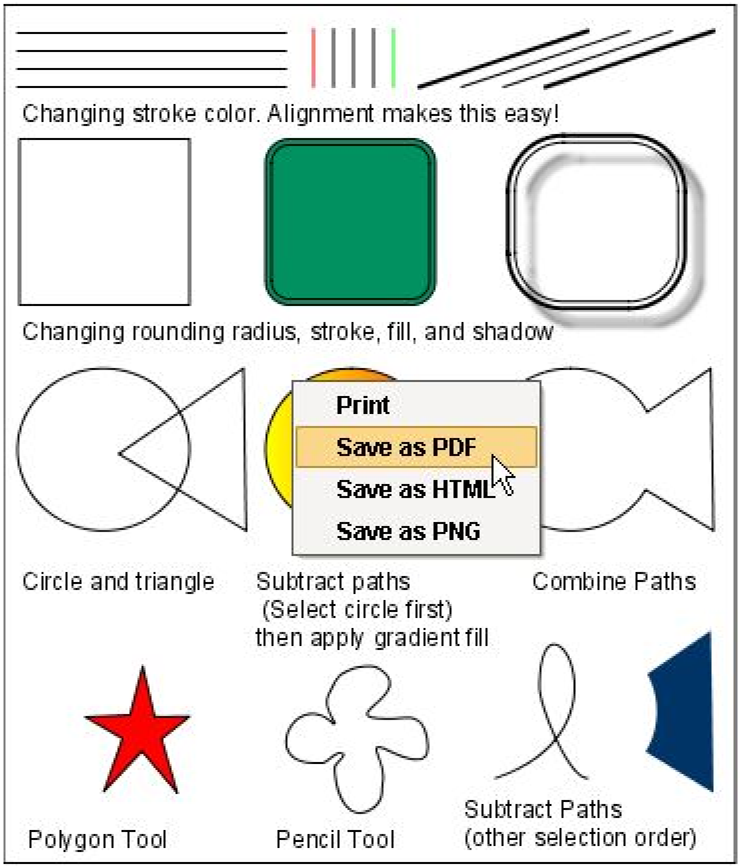
String Functions

The following functions return strings.

Menu Item	Function
startsWith(String, String)	Returns true if the first string starts with the second.
endsWith(String, String)	Returns true if the first string ends with the second.
substring(String, int start)	Returns a substring of String beginning at position start.
join(List aList, String aKeyChain, String aDelimeter)	Used to display an individual attribute of individual objects as a single String. Suppose you have a list of movies and want to show their titles in a comma separated list: @join(getMovies, "getTitle", ", ")@
substring(Object aString, int start, int end)	obtain a subset of a given string. This could be useful if you wanted to restrict a text field to a certain number of chars:@substring(title, 0, 10)@

Saving Reports

Saving a report is simply a matter of right clicking on the report in the FactoryPMI runtime or preview mode of the designer and selecting the format you wish to save it as.



You have the following options:

- Print - print your report to a printer
- PDF - Adobe Acrobat formatted document
- HTML - web page
- PNG - image file

BLOB images

Blob (**B**inary **L**arge **Ob**ject)is a data type for storing large amounts of binary data in an SQL database. *FactoryPMI Reporting* can use Blobs to display dynamic images within reports. This example will illustrate using blobs with MySQL and the free [MySQL Query Browser](#).

Dept Image	First	Last	Department
	Joe	Blogs	ADMIN
	Julia	Hinderson	ENGINEERING
	Adam	Blackwell	ENGINEERING
	Robert	Hill	SALES
	Brian	Green	ADMIN
	Elaine	Short	SALES
	Tim	Moore	ENGINEERING

Using an *Image Placeholder* and blobs to dynamically illustrate table row based on **department**.

Example

We begin with the **employee** table from Tutorial #1 and **emp_images**, a table that maps departments to images

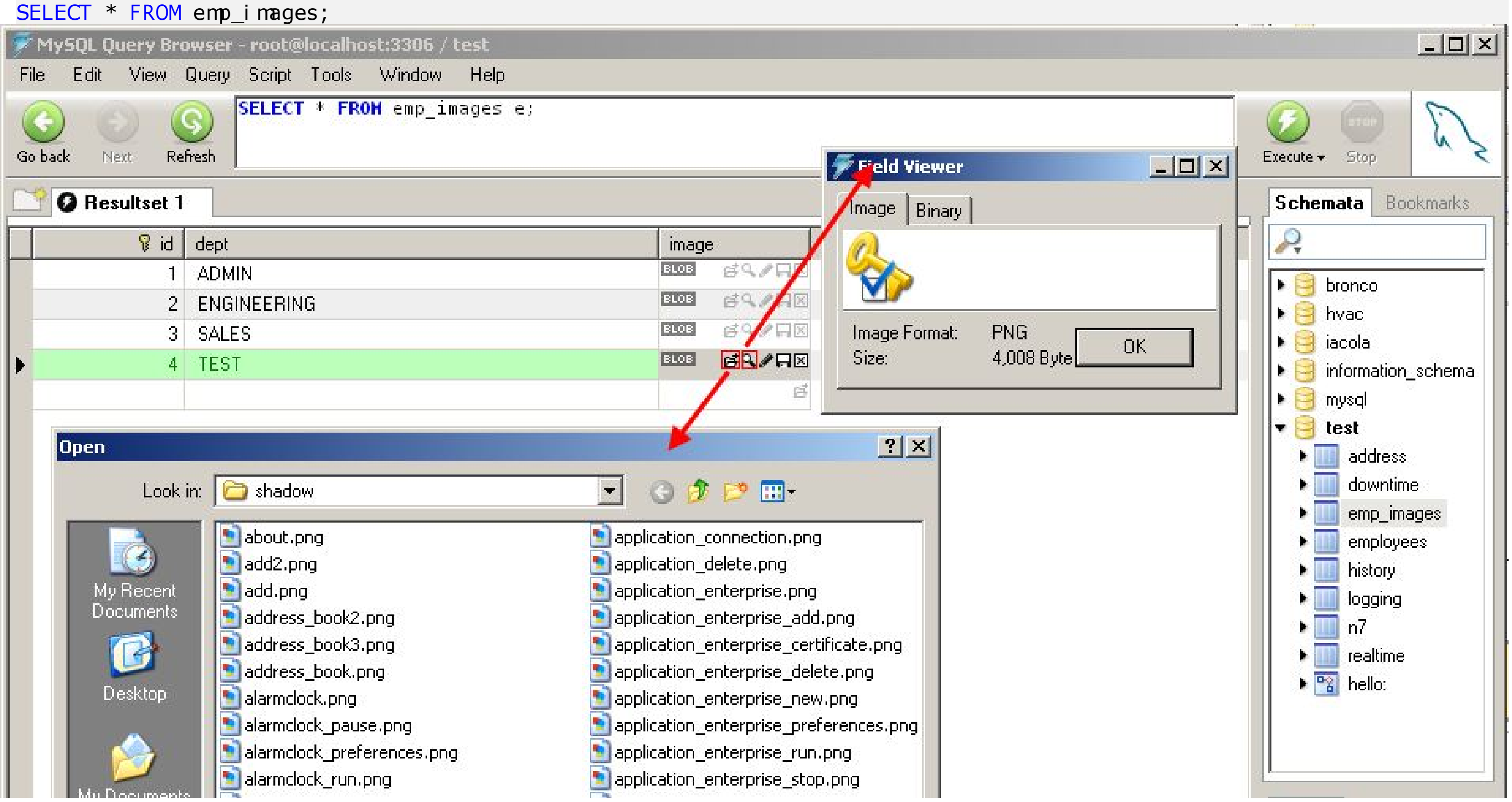
Employee data can be retrieved with the following SQL query:

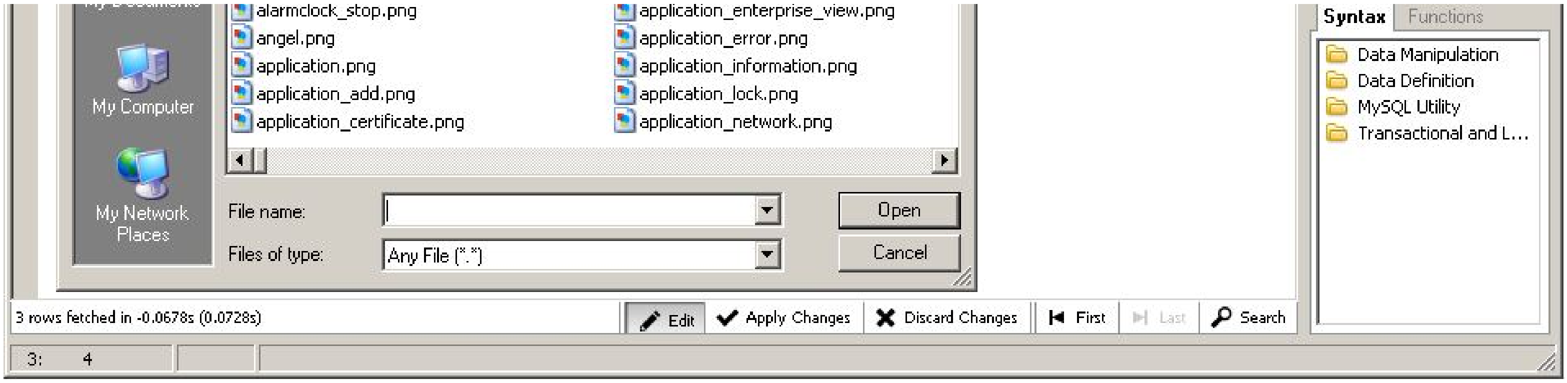
```
SELECT * FROM empl oyees;
```

	first	last	department	lastSell	income	withholdings	vacationdays
1	Joe	Blogs	ADMIN	2002-05-13 00:00:00	46000	5000	22
2	Julia	Hinders...	ENGINEERING	2005-01-01 00:00:00	121000	10000	19
3	Adam	Blackwell	ENGINEERING	2004-02-05 00:00:00	96000	8000	6
4	Robert	Hill	SALES	2004-05-02 00:00:00	35000	200	27
5	Brian	Green	ADMIN	2004-03-15 00:00:00	55000	4000	1
6	Elaine	Short	SALES	2006-05-05 00:00:00	32000	2000	5
7	Tim	Moore	ENGINEERING	2005-12-15 00:00:00	73000	4000	10
8	Chris	McDade	ENGINEERING	2006-07-01 00:00:00	52000	4000	31
9	Andrea	Sanchez	SALES	2006-01-01 00:00:00	73000	5000	16
10	Thomas	Jones	SALES	2006-09-15 00:00:00	115000	12000	4
11	Jefffrey	York	SALES	2006-02-15 00:00:00	69000	1000	18
12	Michael	Bolton	SALES	2004-12-01 00:00:00	29000	500	35
13	Edward	Oscar	ADMIN	2005-12-31 00:00:00	61000	3000	3

Images data can be retrieved with the following SQL query:

```
SELECT * FROM emp_i mages;
```





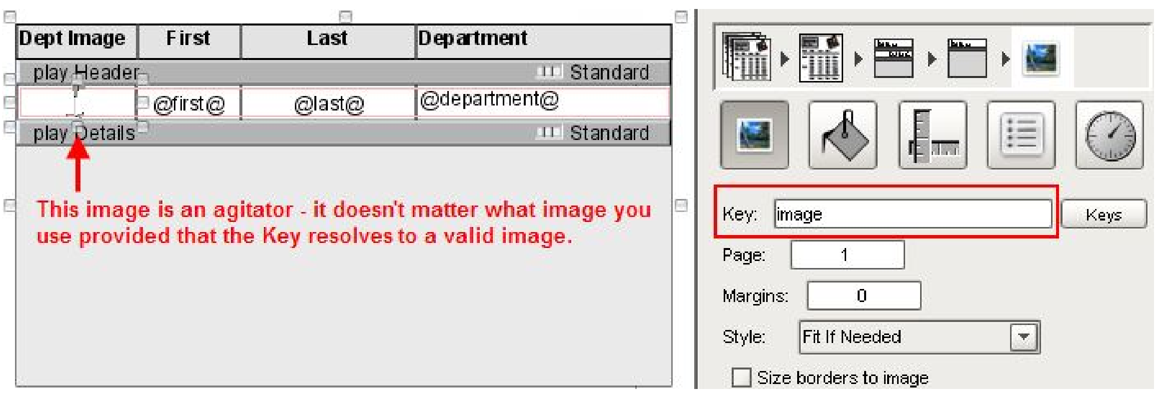
The MySQL Query Browser allows you to upload files as Blobs and view images

The following query will SELECT employees with the image as defined by their department

```
SELECT e.first, e.last, e.income, e.department, i.image FROM employees e, emp_images i WHERE e.department = i.dept;
```

first	last	income	department
Joe	Blogs	46000	BLOB
Julia	Hinderson	121000	BLOB
Adam	Blackwell	96000	BLOB
Robert	Hill	35000	BLOB
Brian	Green	55000	BLOB
Elaine	Short	32000	BLOB
Tim	Moore	73000	BLOB
Chris	McDade	52000	BLOB
Andrea	Sanchez	73000	BLOB
Thomas	Jones	115000	BLOB
Jefffrey	York	69000	BLOB
Michael	Bolton	29000	BLOB
Edward	Oscar	61000	BLOB
Nathan	Zirkowski	NULL	BLOB

Create a table. Select a column and create an image in it. Set the **Key** value to your key, *image*

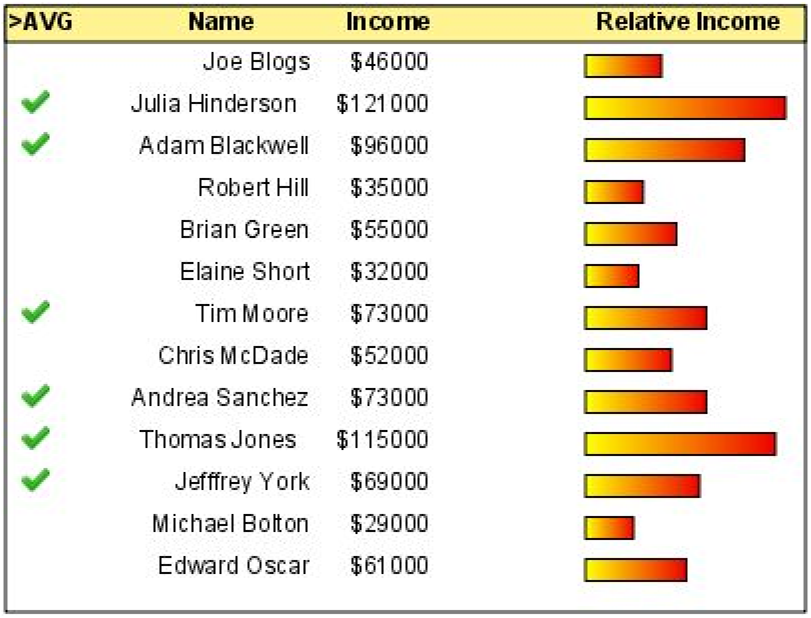


Here's what the table looks like with dynamic images

Dept Image	First	Last	Department
	Joe	Blogs	ADMIN
	Julia	Hinderson	ENGINEERING
	Adam	Blackwell	ENGINEERING
	Robert	Hill	SALES
	Brian	Green	ADMIN
	Elaine	Short	SALES
	Tim	Moore	ENGINEERING

Property Expresssions

Property Expressions are a way that you can dynamically change object properties based on key expressions.

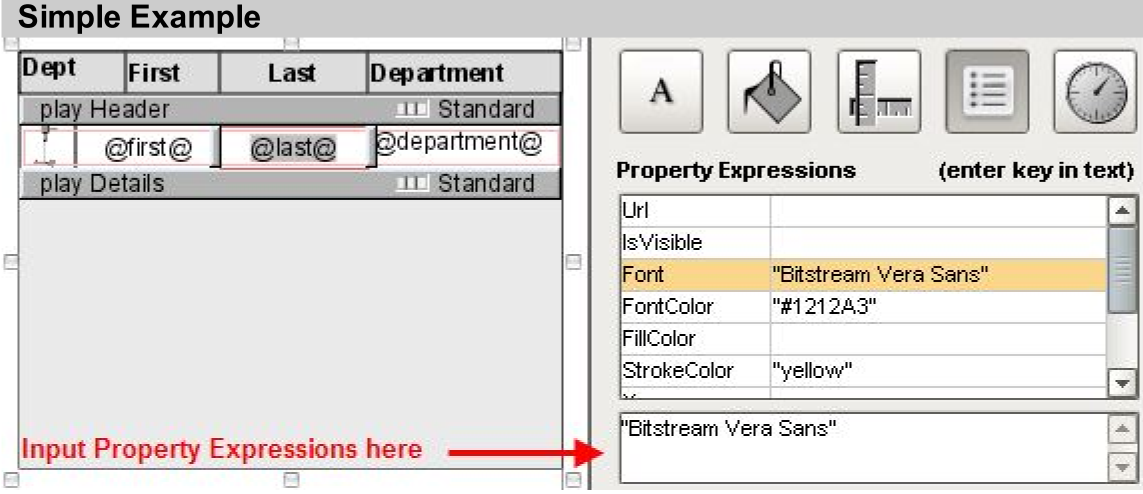


A few simple tricks using *Property Expressions*

Properties

Option	Function
URL	Used for Macromedia Flash based reports. Presently N/A.
IsVisible	1 makes the shape visible, 0 makes the shape invisible.
Font	String based font name to use. Property should evaluate to a string that is a font name in the [Font Attribute Panel]().
FontColor/FillColor/StrokeColor	Changes color of font, fill, and stroke, respectively. Expects colors by **name** or "**#RRGGBB**" (Hex) format.
X/Y	Object position that expects a number.
Width/Height	Dynamically Size an object

Simple Example



Playing with *Property Expressions* to illustrate their use.

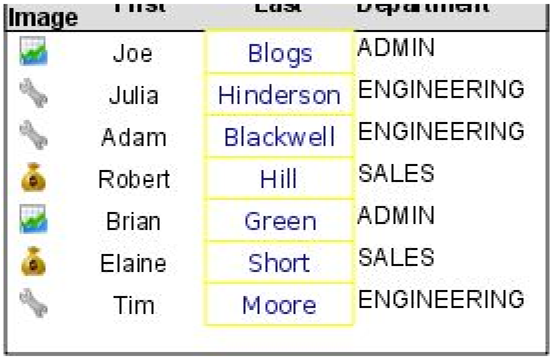
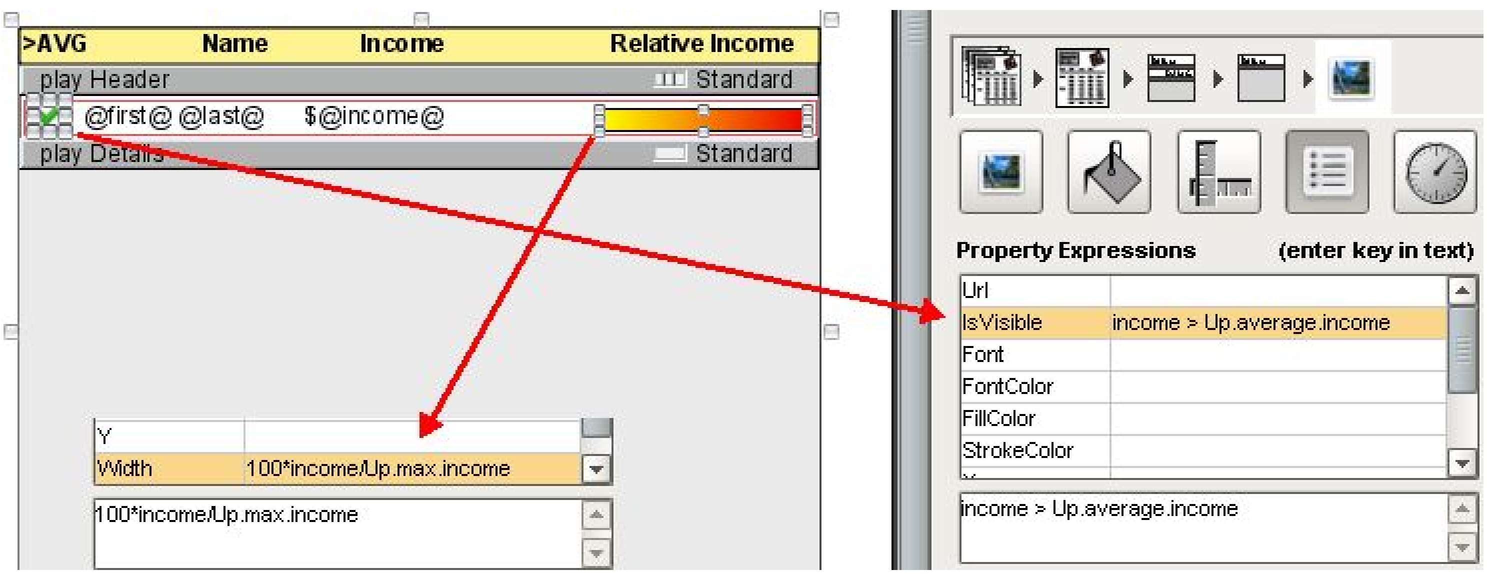


Image	First	Last	Department
	Joe	Blogs	ADMIN
	Julia	Hinderson	ENGINEERING
	Adam	Blackwell	ENGINEERING
	Robert	Hill	SALES
	Brian	Green	ADMIN
	Elaine	Short	SALES
	Tim	Moore	ENGINEERING

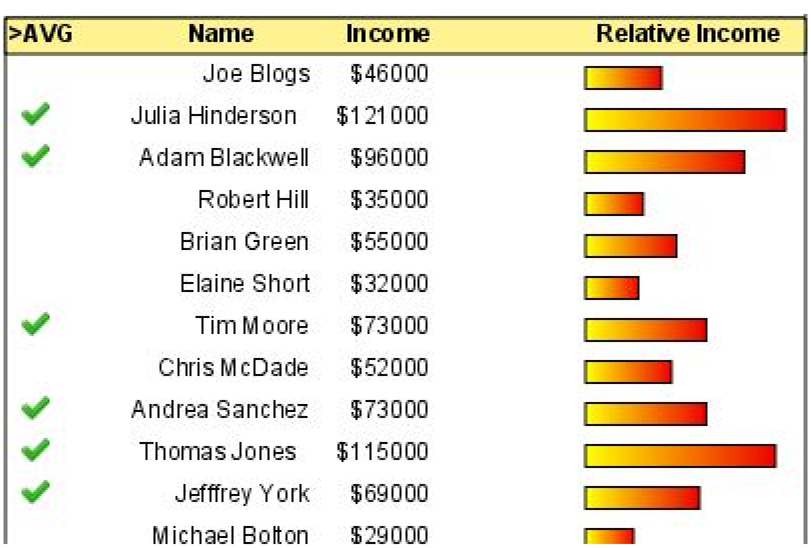
Dynamic Example

We will use 2 object's *Property Expressions* to illustrate their uses.

- Add an image of a check box. This is included under Builtin/icons/check2.png.
- We want a check box is the employees income is greater than the arithmetic average of employees.
- Set the *isVisible* property to "**income > Up.average.income**". Notice that we neglect the **@** for properties in the Inspector Panel.
- Add a rectangle with a gradient fill.
- We want the width of the rectange to be 100 pixels * the ratio of the employee's income versus the employee with the max income.
- Set the *Width* property to "**100 * income / Up.max.income**"



Playing with *Property Expressions* to illustrate their use.



>AVG	Name	Income	Relative Income
	Joe Blogs	$46000	
✔	Julia Hinderson	$121000	
✔	Adam Blackwell	$96000	
	Robert Hill	$35000	
	Brian Green	$55000	
	Elaine Short	$32000	
✔	Tim Moore	$73000	
	Chris McDade	$52000	
✔	Andrea Sanchez	$73000	
✔	Thomas Jones	$115000	
✔	Jefffrey York	$69000	
	Michael Bolton	$29000	

TIP	Get creative with property expressions. They're very powerful!