

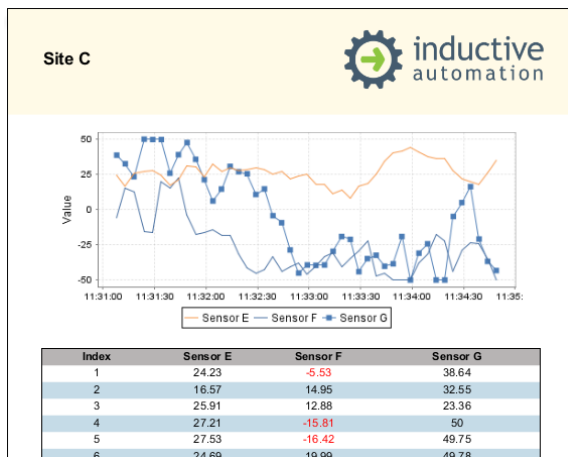
1. Reporting	2
1.1 Report Designer Interface	6
1.2 Report Data	12
1.2.1 Report Parameters	15
1.2.2 Named Query Data Source	19
1.2.3 SQL Query Data Source	21
1.2.4 Basic SQL Query	23
1.2.5 Tag Historian Query	27
1.2.6 Tag Calculation Query	29
1.2.7 Alarm Journal Query	31
1.2.8 Scripting Data Source	33
1.2.9 Static CSV	38
1.2.10 Nested Queries	39
1.3 Report Design	43
1.3.1 Report Design Tools	47
1.3.2 Stroke and Fill Properties for Reports	51
1.3.3 Data Keys	59
1.3.3.1 Keychain Expressions	69
1.3.4 Report Charts	73
1.3.4.1 Getting Started with Report Charts	78
1.3.5 Report Tables	83
1.3.5.1 Getting Started with the Report Table	89
1.3.5.2 Table Rows	93
1.3.5.3 Table Row Versioning	98
1.3.5.4 Charts Inside of Tables	102
1.3.5.5 Grouping Data Inside of Tables	112
1.3.5.6 Table Groups	119
1.3.6 Images in Reports	125
1.3.7 CrossTab and Simple Tables	131
1.4 Report Schedules	136
1.4.1 Scheduling Actions	140
1.5 Common Reporting Tasks	147
1.5.1 Tutorial: The Report Workflow	148
1.5.2 Labels with Embedded Barcodes	158
1.5.3 Converting Legacy Reports	163

Reporting

The Ignition Reporting Module makes creating professional reports easy with a rich library of tools including: images, graphs, tables, and basic shape tools. The Reporting Module enables you to create your own custom reports on the fly or generate them based on a schedule. Data is introduced through Ignition, providing access to any SQL database.

The Report Scheduler allows automatic report generation and automated distribution. Access to live reports is available through the web-based Ignition runtime (a Java application), providing authenticated users access from anywhere. Access is 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
- Inventory tracking
- Efficiency monitoring
- Historical trending
- Downtime tracking
- Quality assurance
- Data analysis



Features

Here are some of the other innovative features of the Reporting Module:

- Report designer Interface
- Scheduled report generation
- Powerful data collection utilities
- Drag-and-drop query builder
- Table and chart components
- 2D barcode generation
- Familiar property editing
- Flexible report distribution to file, email, and FTP
- Scripting capabilities

Intuitive Report Design

The [report designer interface](#) was designed with the same look and feel as the other systems in Ignition. Once you install the Reporting Module, you can open the Ignition Designer and see a Reports section in the Project Browser. Reports are a Project-Level resource. They can be created independently or set to be viewed in a Vision window. With a right-click on the **Reports** node in the Project Browser and the option to create a new report is a click away.

From the first informative panel to full previews of the report you are creating, the Reports workflow significantly reduces the time it takes to design, edit, and distribute reports. The Report workflow is understandable to the new developer, yet powerful enough for a seasoned analyst. Select the **Design** tab to access the Report Design Palette components, charts, and shapes, and other design tools.

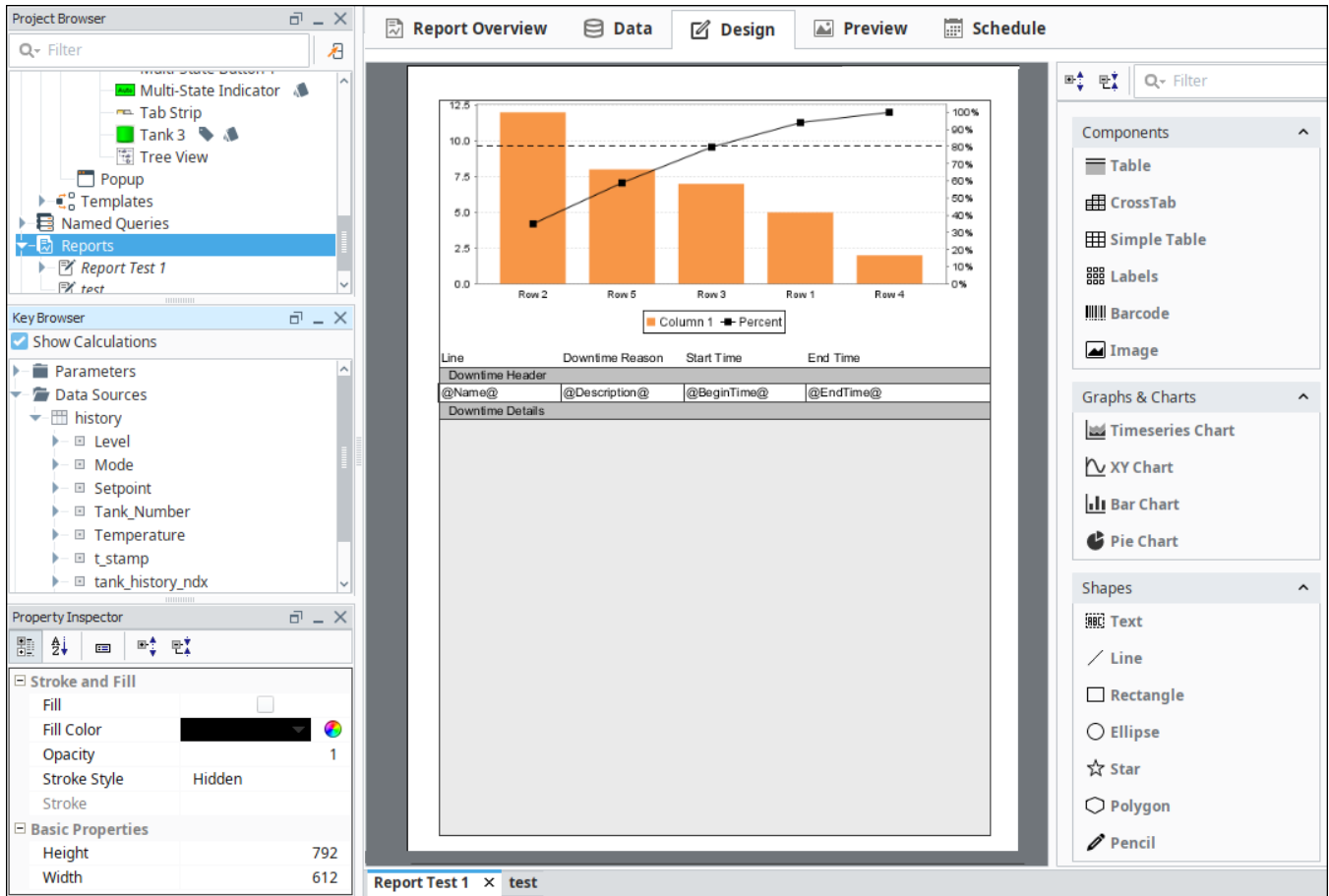
On this page ...

- [Features](#)
 - [Intuitive Report Design](#)
 - [Powerful Components](#)
 - [Reporting Module Components for the Vision Module](#)
 - [Scheduled Report Execution](#)
 - [Supports Multiple File Formats](#)
- [Scripting in the Reporting Module](#)
- [Trial Mode Functionality](#)
- [Legacy Reports](#)



Reporting Interface

[Watch the Video](#)



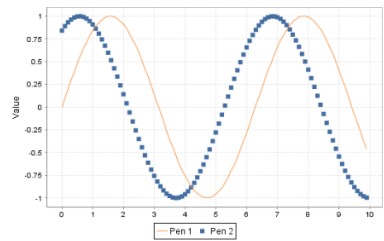
Powerful Components

The variety of [report design components](#) include tables, crosstab tables, XY charts, barcodes, pie charts, and bar charts. [Report tables](#) can dynamically add pages to account for varying amounts of data, or change appearance based on certain values. [Report charts](#) can provide visual representation of comparisons and trends in data.

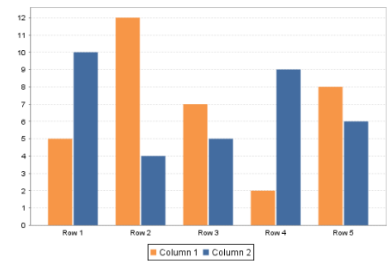
Barcode



XY Chart

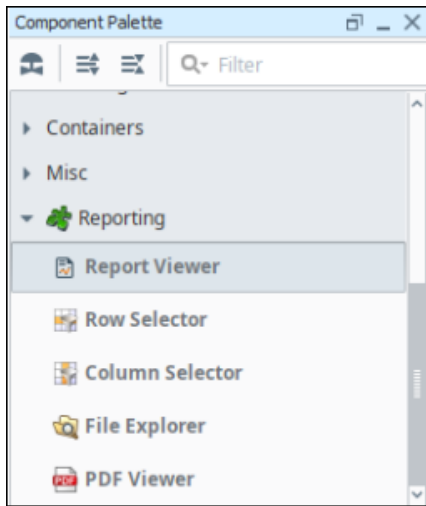


Bar Chart



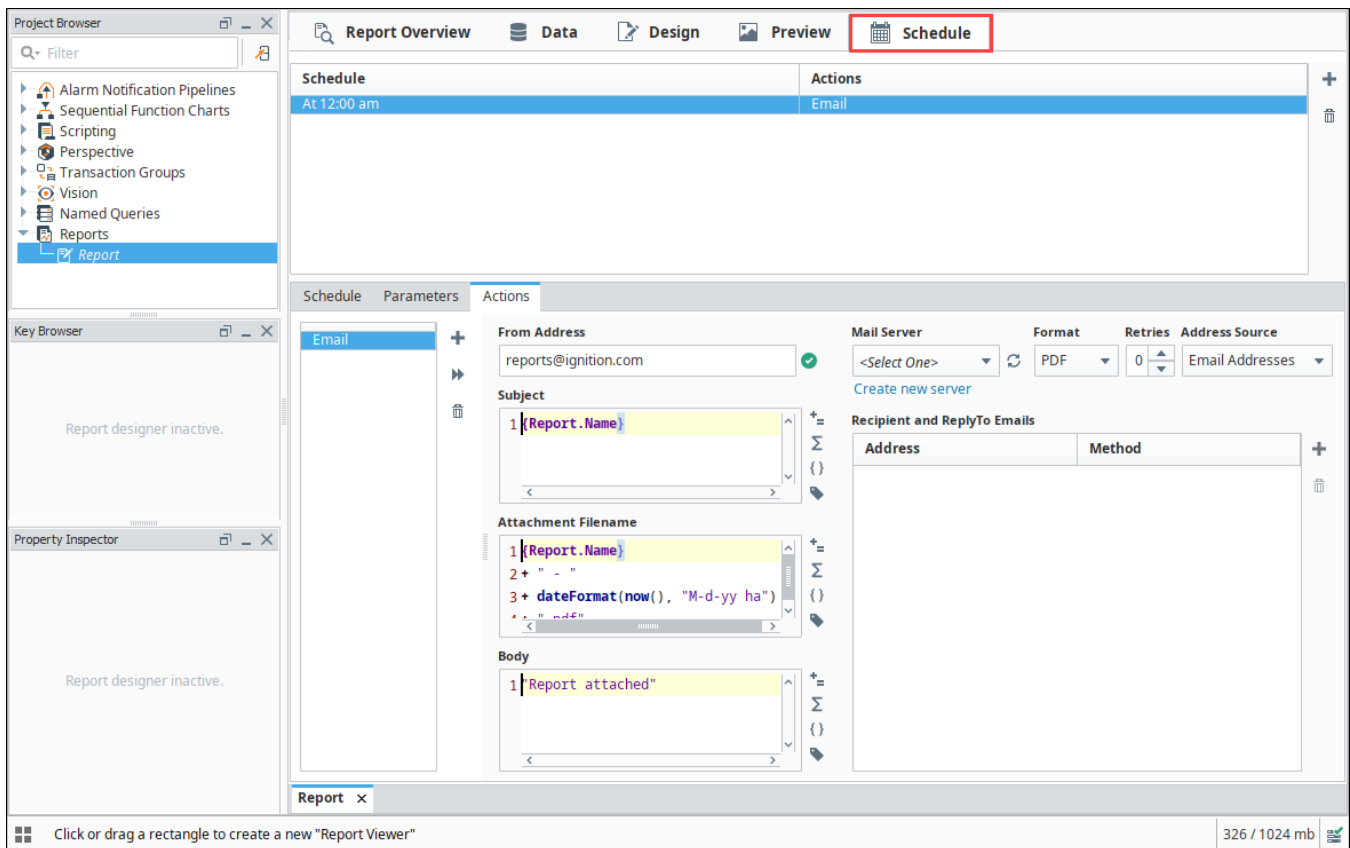
Reporting Module Components for the Vision Module

The Reporting Module provides several components that can be used with the Vision Module. The Report Viewer component allows reports to be viewed directly from the client. The Row Selector and Column Selector components let clients manipulate datasets graphically, while the File Explorer and PDF File Viewer components allow clients to access files outside of Ignition.



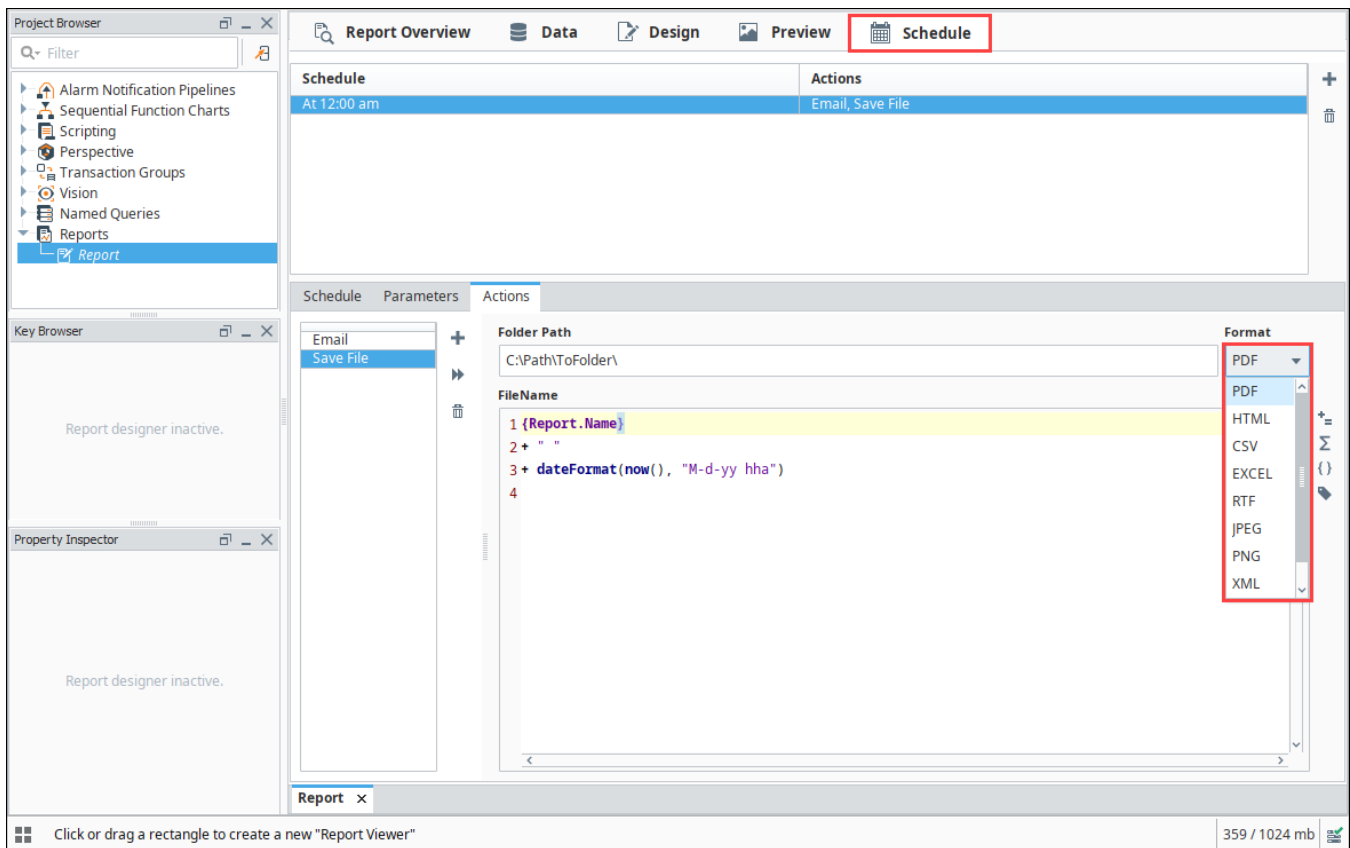
Scheduled Report Execution

The [scheduling system](#) allows you to set the date, time, and frequency for executing a report. You can also set options for distribution of the report, including email, print, and save. Select the **Schedule** tab to access these options.



Supports Multiple File Formats

Reports can be generated in the following file formats: CSV, HTML, JPEG, PDF, PNG, RTF, XML, and EXCEL. Use the **Format** dropdown under **Schedule > Actions** tabs to select a format type.



Scripting in the Reporting Module

The Reporting Module allows you to extend the existing functionality through scripting. You can customize reports with one of the following:

- **Script Data Source:** The Script Data Source type allows you to add to or modify your data set with Python code. For example, you can alter other Data Sources, combine results, and do complex calculations.
- **Scheduling:** The Run Script action in the Report Scheduling system enables you to create a script to do exactly what you want with a generated report.
- **Scripting Functions:** The Reporting Module adds additional functions to the list of `system.report.*` functions available throughout Ignition. You can use these functions to execute and/or distribute reports on demand.

Trial Mode Functionality

Like other systems in Ignition, the Reporting Module has full functionality in trial mode. There is no limit on the number of reports or how you can view or distribute them. Reports created in trial mode will have a watermark on each page.

Legacy Reports

To take advantage of Ignition's powerful platform, the Reporting Module was updated in version 7.8. You may view reports created by an older Reporting Module version in your project. If you need to modify an existing report, you'll still have access to the same customizer that you always had, double-clicking on them to open up their editor. They will continue to work as they always have without any modification. To learn more about converting reports created before Ignition version 7.8, refer to the section on [Converting Legacy Reports](#).

We recommend that you convert your older reports to the latest version to take advantage of the many features of the Reporting Module.

In This Section ...

Report Designer Interface

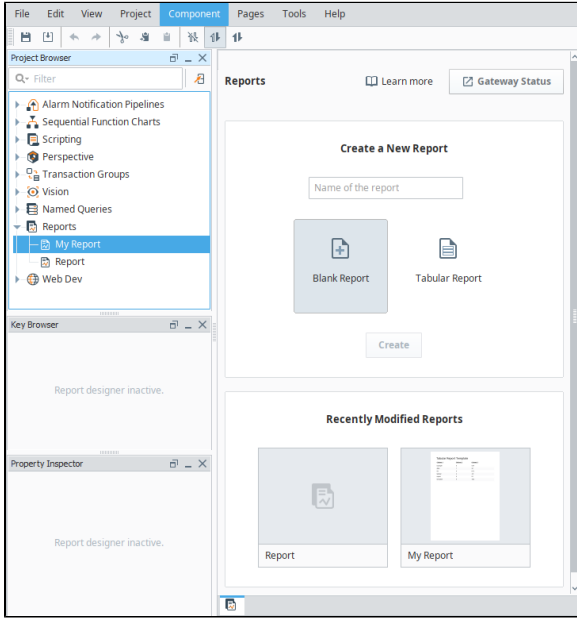
Built from the ground-up to be intuitive and familiar, the Reporting workspace has a logical workflow which makes it easy for anyone to create meaningful reports. Before we talk about the workflow, let's first mention where you create and find your Reports. Reports are located in the Project Browser of the Project Area under Reports. Any report that is created in your project will be found here.

The Reports Welcome tab presents you with two types of reports to help get you started: a Blank report and a Tabular report. The Blank report lets you design your report from the ground up. The Tabular report is a template with some sample data and a report design. This is helpful if you want to present the data in your report in tabular format.

To create a report, select a report type, enter a report name, and click 'create' and you are well on your way. It will even show you the most recently modified reports along with the thumbnail of the report. The Reports Welcome tab provides a quick way to create a new report and update an existing report.

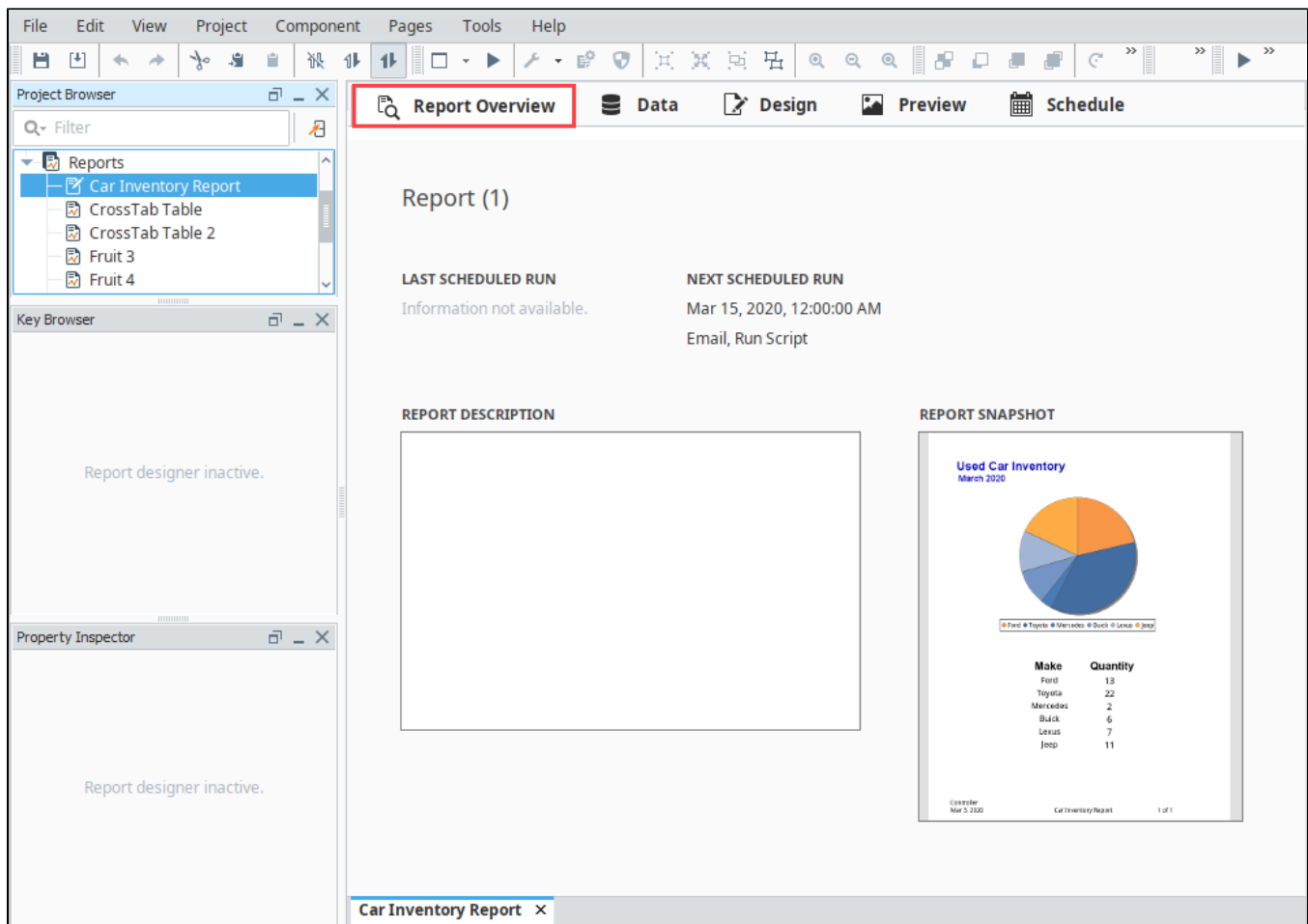
On this page ...

- [Report Overview Tab](#)
- [Data Tab](#)
- [Design Tab](#)
- [Preview Tab](#)
- [Schedule Tab](#)
- [Key Browser](#)
- [Property Inspector](#)




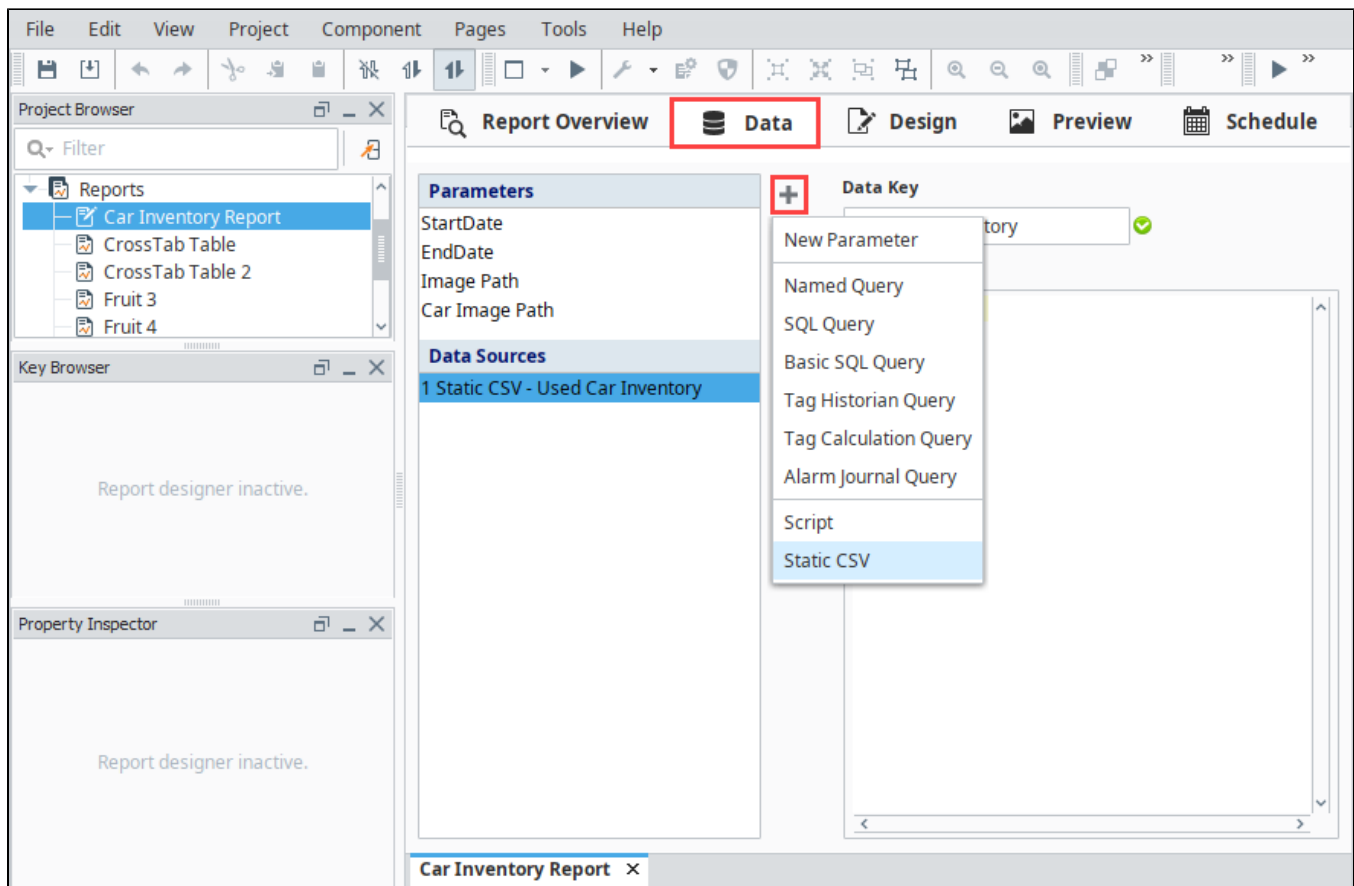
Report Overview Tab

The Report workspace has five tabs which make up the workflow of a report: Report Overview, Data, Design, Preview, and Schedule tabs. Once you create a report, the first step in the report workflow process is the Report Overview. The Report Overview tab provides valuable information at a glance about your report. There is space to add notes, which is a good place to provide background information and context about the report. The Overview will also show a thumbnail of the most recent report, its last execution time, and the next scheduled execution. The Report Snapshot is generated each time you visit the Preview tab.



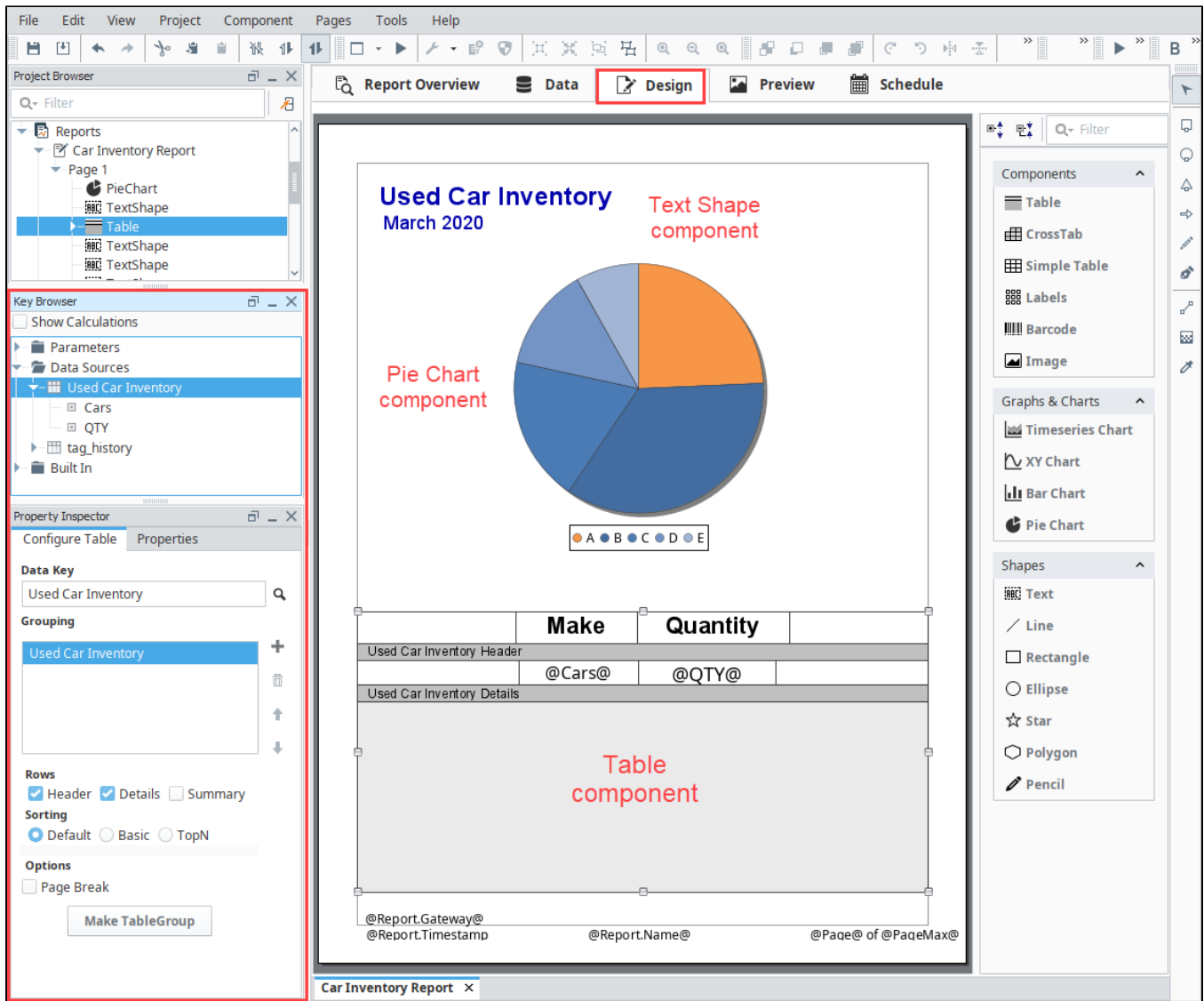
Data Tab

The **Data tab** is the first step for configuring a new report. You'll notice the list on the left that has labels for Parameters and Data Sources, as well as some buttons. On this page you can generate data that you want to make available on your report. Click the **Add**  icon to see a list of built-in Data Sources and Parameters. Selecting any one of them will open the configuration screen for that Data Source type and enable you to start configuring the data for your report.



Design Tab

If you've ever designed a Perspective View or a Vision window, the [Design tab](#) should make you feel right at home. Starting from the top-left – the structure of a Report is represented in the **Project Browser** tree, allowing easy visualization of the tree of elements on your report, but don't stop there! Grab a component off the Reporting Palette on the far right and add it to the page. In place of the Tag Browser, there is a **Key Browser** that gives you an easy way to add the Data Sources and Parameters that you configured in the Data tab to the Report. For complex components, there are configuration tabs in the **Property Inspector** to supplement the Property Inspector table that all the design objects have.



Preview Tab

The **Preview** tab, while not one of our three design steps, is a nice feature while building your report. This tab provides quick visual feedback on your report. Not only does it provide you an instant example of what your report looks like, but it also gives you the ability to view the actual data next to your report preview. Having a snapshot of your data can be incredibly helpful when trying to figure out why your report doesn't appear as you expect it to. Whether you have an error in your query, or a bad Data Key in a component, you'll be able to figure out the structure of the data your report is getting and quickly find ways to solve any problems.

The screenshot shows a report designer interface with the following components:

- Project Browser:** Lists 'Named Queries' and 'Reports', with 'Car Inventory Report' selected.
- Key Browser:** Shows 'Report designer inactive.'
- Property Inspector:** Shows 'Report designer inactive.'
- Preview Tab:** Displays the report content:
 - Title:** Used Car Inventory
 - Subtitle:** March 2020
 - Chart:** A pie chart showing the distribution of car makes.
 - Table:**

Make	Quantity
Ford	13
Toyota	22
Mercedes	2
Buick	6
Lexus	7
Jeep	11
 - Footer:** Controller Mar 3, 2020 | Car Inventory Report | 1 of 1
- XML Source Code (Right Panel):**

```

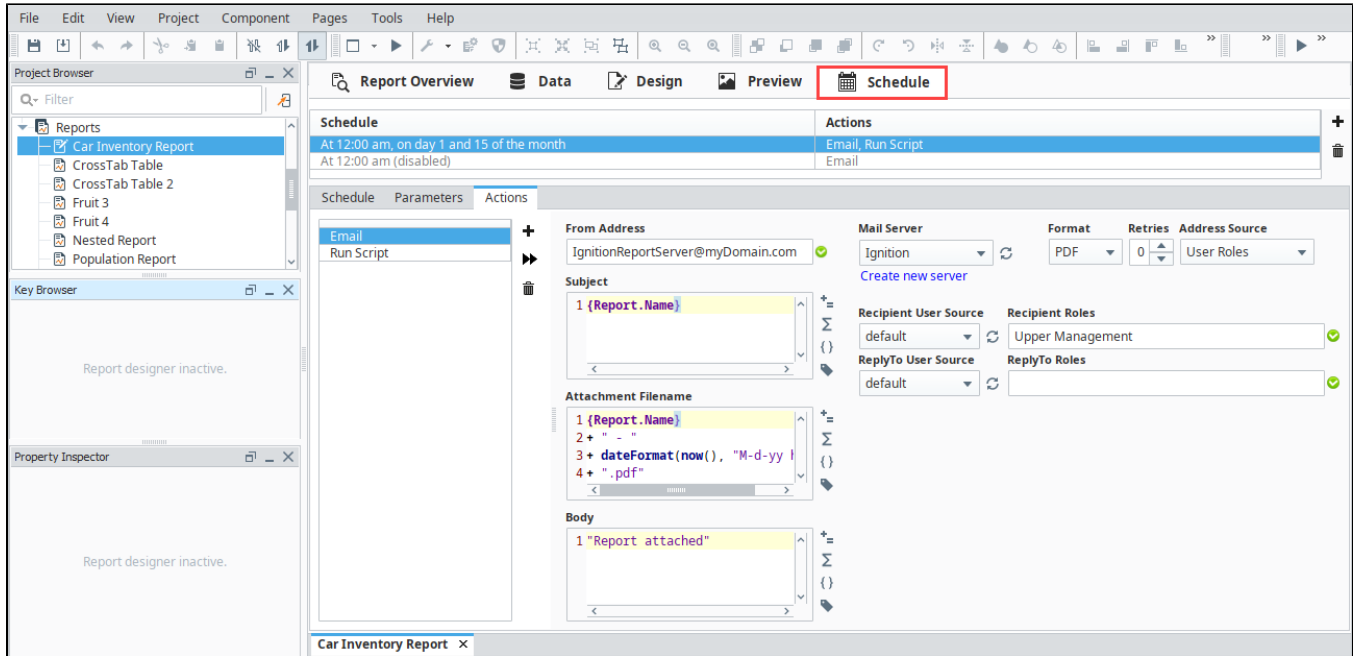
1 <?xml version="1.0" ?>
2 <sample-data>
3 <!--This is the raw data used to create the
4 <Car Image Path>http://i.ndtvimg.com/i/2016
5 <EndDate>2020-03-03 13:35:47.228</EndDate>
6 <Image Path>http://i.stack.imgur.com/WCveg.
7 <Report>
8 <Gateway>Controller</Gateway>
9 <Name>Car Inventory Report</Name>
10 <Path>Car Inventory Report</Path>
11 <Timestamp>2020-03-03 13:35:47.226</Tin
12 </Report>
13 <StartDate>2020-03-03 05:35:47.227</StartDa
14 <Used Car Inventory>
15 <row-0>
16 <Cars>Ford</Cars>
17 <QTY>13.0</QTY>
18 </row-0>
19 <row-1>
20 <Cars>Toyota</Cars>
21 <QTY>22.0</QTY>
22 </row-1>
23 <row-2>
24 <Cars>Mercedes</Cars>
25 <QTY>2.0</QTY>
26 </row-2>
27 <row-3>
28 <Cars>Buick</Cars>
29 <QTY>6.0</QTY>
30 </row-3>
31 <row-4>
32 <Cars>Lexus</Cars>
33 <QTY>7.0</QTY>
34 </row-4>
35 <row-5>
36 <Cars>Jeep</Cars>
37 <QTY>11.0</QTY>
38 </row-5>
39 </Used Car Inventory>
40 <tag_history>

```

Schedule Tab

[Scheduling a report](#) to run is built-in and easy to use. The **Schedule** tab is an incredibly powerful way to automate the execution and delivery of your reports. Whether you have extremely long running queries that you don't want tying up a window, need to deliver to multiple file servers via FTP, or want to simply email to an existing roster, this tab provides you the means to do it. It's broken up into two major areas. On top, you find a simple table of schedules that have been established. To add a schedule, click the **Add +** icon. With a schedule added, you can now choose when and how that scheduled Action will occur. Our schedules use the common Cron format, but don't worry if you aren't familiar with it. The preset options and combo-box configurability allow you to easily create just about any schedule you can imagine.

Once you selected your schedule, you can add parameters that may affect your Actions. Actions are incredibly powerful and allow you to choose how you want to distribute or act on your finished reports. Whether saving a file locally, posting to an FTP server, or emailing, you can get your report where it needs to be automatically. If our normal distribution Actions somehow fall short, you can choose to act on the finished report by triggering a script. We've also given you a convenient way of immediately executing the Action.



Key Browser

The Key Browser contains three very important elements: [Data Keys](#), [Built-in Keys](#), and the [Show Calculations](#) property. Data Keys are used to pull values from your data sources that you configure in the Data tab and display them in your report. They act as placeholders for your data and resolve to values when your report is generated. Built-In Keys are utility type functions that are commonly used when generating a report, such as adding a report name, date, page numbers, etc. The **Show Calculations** property adds several aggregates to each data key allowing you to display the total of a key. They are typically used in a summary row of a Table component.



Property Inspector

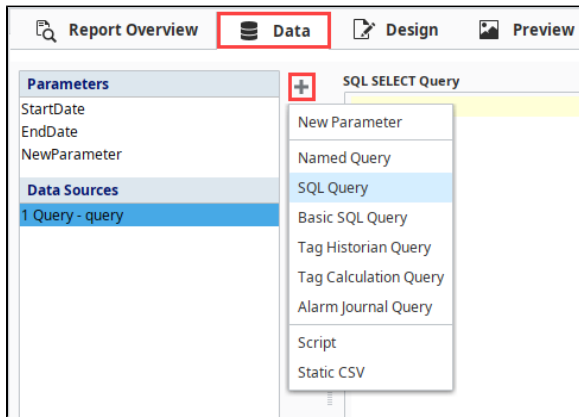
The Property Inspector is similar to the Property Editor in the Designer. It displays the properties of the selected component, but also includes configuration tabs for a number of the more complex report design objects such as the Pie Chart, Barcode, Tables, etc. The configuration tabs are specific to the selected Report component, and will only be used for the [Report Design Components](#).

Related Topics ...

- [Report Data](#)
- [Report Design](#)
- [Report Schedules](#)
- [Data Keys](#)


Report Data

The most critical part of any report is data. In the [Reporting Module](#), all data is collected as either a Parameter or a Data Source, and it is all configured in the Data tab of the report. Clicking on the **Add**  icon allows you to add new sources of data, bringing up a menu so you can choose the type of data to add. If a Parameter or Data Source is highlighted, clicking the **trashcan**  icon will delete that particular source of data.



On this page ...

- [Order Matters](#)
- [Parameters](#)
- [Data Sources](#)





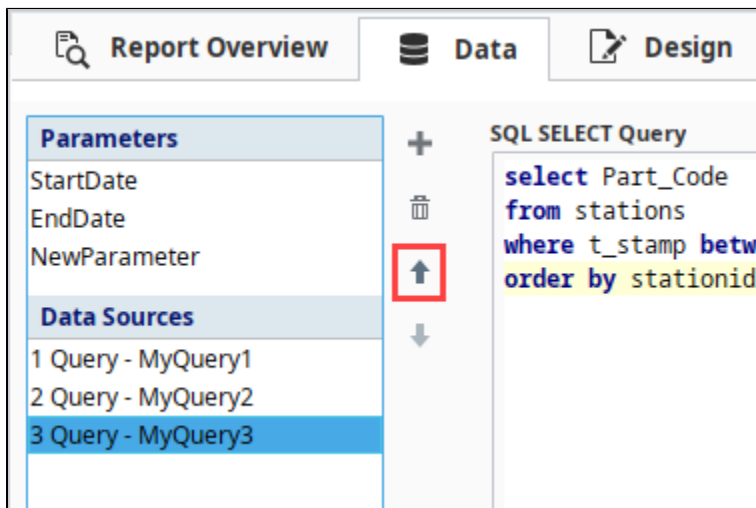
Report Data Tab

[Watch the Video](#)

Order Matters

The order of the Parameters and Data Sources is important. In some cases, a parameter or data source may have the capability to reference the results of another type, such as a Parameter referencing the value of another Parameter. Parameters and Data Sources may *only* reference other types of data that are listed vertically above them in the list. In short, the bottom-most Data Source may reference all other parameters and data sources, while the top-most Parameter may not reference any other parameter or data source.

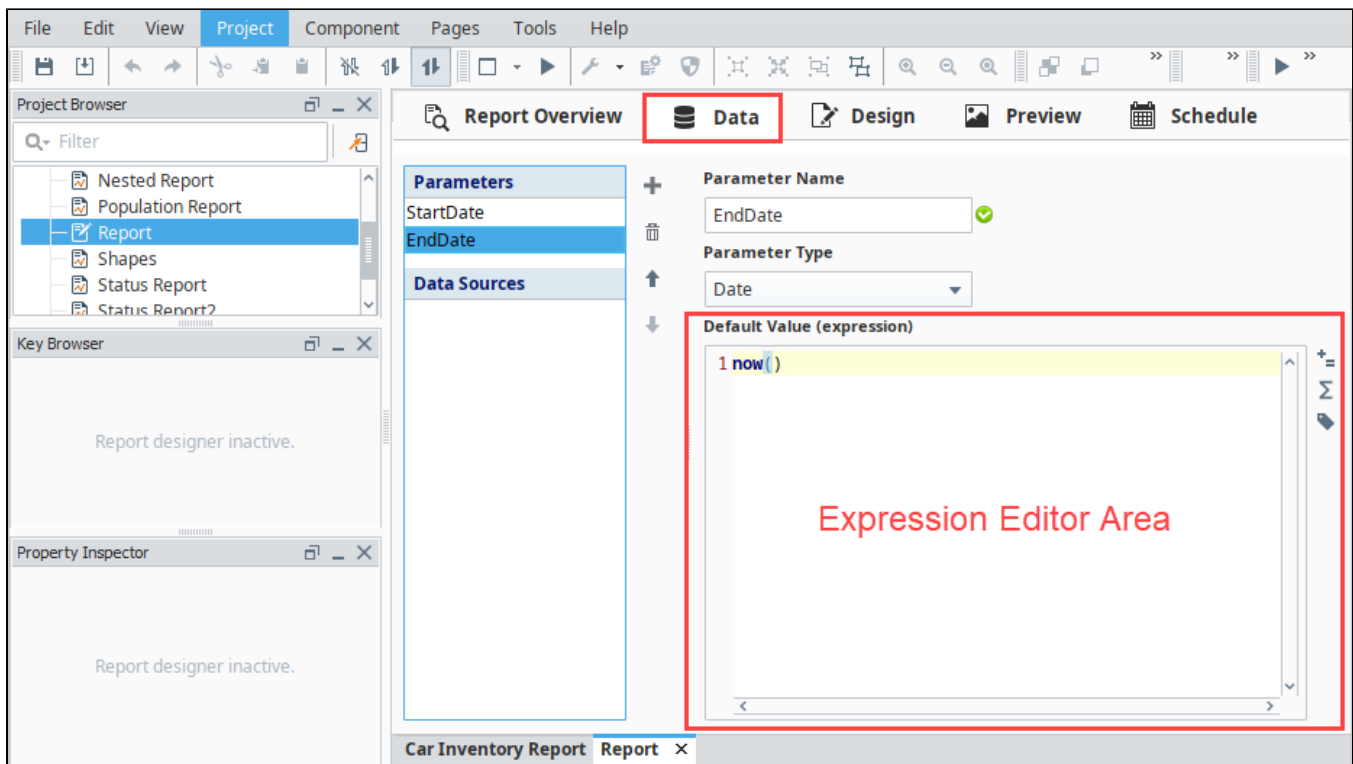
To reorder parameters, use the **up**  and **down**  arrow icons.



Parameters

[Parameters](#) are the way you get dynamic data into your charts. These properties are exposed on the [Report Viewer](#) component and can be bound to other components to allow your users to select what report data they want to see. You can use these to set up dates, time ranges, area selectors, titles, and anything else you want in your report.

By default, StartDate and EndDate properties are automatically created for you because they are almost always used to filter report data. You can, of course, delete them if you don't need them.



Data Sources

Data Sources are the primary means of getting data out of where it lives and into a report. There are eight Data Sources included in the Reporting Module: six Query types, a Script data source and Static CSV. Each data source offers a unique method of collecting (or amending in the case of *Script* Source) data. Aside from the *Script* data source, all will require you to specify a *Data Key*. The *Data Key* is a unique identifier that represents the top level of the data collected through this source – think of it as the label for the data source, or as the parent node of your source's collected 'data tree'. When your report is generated, the data collected under this identifier is passed to the reporting engine, which uses this identifier to appropriately place your data in the final report.

Here is a complete list of all the data source types:

- **Named Query:** A pre-configured query that runs as a prepared statement. If your Gateway is already using Named Queries to display data, then you can easily add those queries to your report.
- **SQL Query Data Source:** A straightforward query that allows parameters to be inserted with question marks (?) and has a graphical Query Builder.
- **Basic SQL Query:** A simplified version of the SQL Query that supports references directly in the query using the brace characters { }.
- **Tag Historian Query:** The same Tag History query builder you are familiar with from property bindings.
- **Tag Calculation Query:** Similar to the Tag Historian query, but this one allows calculations to be performed on the resulting data (min, average, duration on, count off, etc.).
- **Alarm Journal Query:** The same alarm journal query builder you are familiar with from the Functions property binding.
- **Script:** A blank script that you can use to create a dataset in any way you like.
- **Static CSV:** Static CSV data can be copied and pasted directly as an easy way to test your reports.

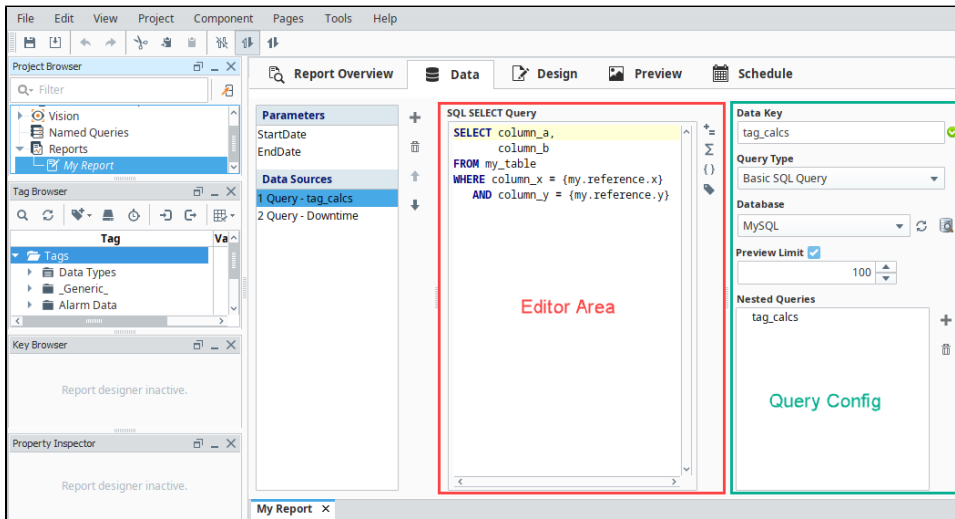
Developer Tip

Columns returned by data keys need to have unique names. When building a report, the builder interface refers to each column by its name, so if multiple data sources are returning columns with a similar name/alias, you'll run into name collisions.

The Query data source types share some common features in the GUI, and they all return what is essentially a dataset. If you take a look at the image below, you'll see we outlined two main areas of the Basic SQL Query data source type. By default, the largest part of any Query panel is taken up by the central Editor Area. This space is tailored for each query type. On the right side of the Data panel, we have a "Query Configuration" area. In the screenshot, you'll see there are options for the *Data Key*, *Query Type*, *Database*, *Preview Limit* and *Nested Queries*. Of these, *Data Key*, *Query Type* and *Nested Queries* are shared among all Queries, while *Preview Limits* are conveniently available in SQL query types. Here is some brief information on each type:

1. **Data Key** - The identifier we will reference when we design our reports. The Data Key needs to be unique, can not contain special characters (spaces, underscores, dashes are allowed), and must start with a letter. We added a convenient validator to help detect name collisions.
2. **Query Type** - Gives you a convenient place to change the type of query.

- Nested Queries** - All of the query-based Data Source types can have [nested queries](#) in them. That is, for each row of returned data, a second query can be run based on the results in that row. You can have multiple subqueries for each row, and nest queries as deep as you want. This is particularly useful when you have data organized into runs and want to see historical data for each run.
- Preview Limit** - Allows you to choose to limit the number of rows returned from a query when a report is being generated in the Preview Panel, and when data is sampled for the purposes of generating Data Keys in the Designer. If supported by your database, this makes it much easier to craft and preview deep query structures without the overhead and wait-times of long-running queries.



Report Parameters

As we covered in the overview, the Data Panel is pretty simple at first glance. When creating a new report, there is an empty list of Data Sources as well as pre-made parameters. **Parameters** are Ignition [Expressions](#). These parameters resolve at report execution time and provide a convenient way to dynamically specify content for your report. These expressions are expected to resolve to specific types.

The Parameter Types available in the ComboBox will be familiar to anyone who has used Expressions, and include:

- **Date**
- **String**
- **Long**
- **Double**
- **Boolean**
- **Dataset**
- **Binary Data**

Adding Parameters function similar to custom properties on Perspective views and Vision windows and their components. Their value can be shown directly on your report, or be referenced by data sources and other parameters. Parameters are given a default value when first created, but this value can be overridden once the report runs.

How Parameters Are Used

Parameters share some similarities to Custom Properties in Perspective and Vision components in that they allow an Ignition user to specify dynamic data to be used in reports at runtime. For example, a manufacturer may want to generate reports for the active production lines every Monday morning, but you don't know ahead of time which production lines will be running. We can create a parameter called *activeProductionLines* and use an expression or Tag value to determine which lines were active, and use the parameter in a SQL Query datasource to only include data from active lines.

Parameters can be declared with an empty (null) value. In this event, it can be fed values from the Vision Report Viewer component, or in the Report Scheduling panel. The Report Viewer can use parameters to bind to data through Ignition Bindings. If a parameter is supplied a default value in the Report Data panel (we call these *default parameters*), that value will be automatically be supplied to the Report Viewer component. If a parameter has no default, the Report Viewer and Preview Panel will notify the user that a parameter is undefined. It is important to note that both Scheduled Report and Vision component parameters take precedence over default parameters, so any default parameter can easily be overridden while also providing a baseline value.



Note that default values for parameters are evaluated in the Gateway, not by clients. As such, the expressions have a Gateway scope.

Default Starting Parameters

When you first create a report, you start with two parameters: **StartDate** and **EndDate**. EndDate will have a default value of 'now()', while StartDate will have a default value of 'dateArithmetic(now(), -8, "hr")'. The default value of the StartDate can be modified from 8 hours to 10 days by editing the Default Value expression.

StartDate Default

```
dateArithmetic(now(), -8, "hr")
```

StartDate Modified

```
dateArithmetic(now(), -10, "days")
```

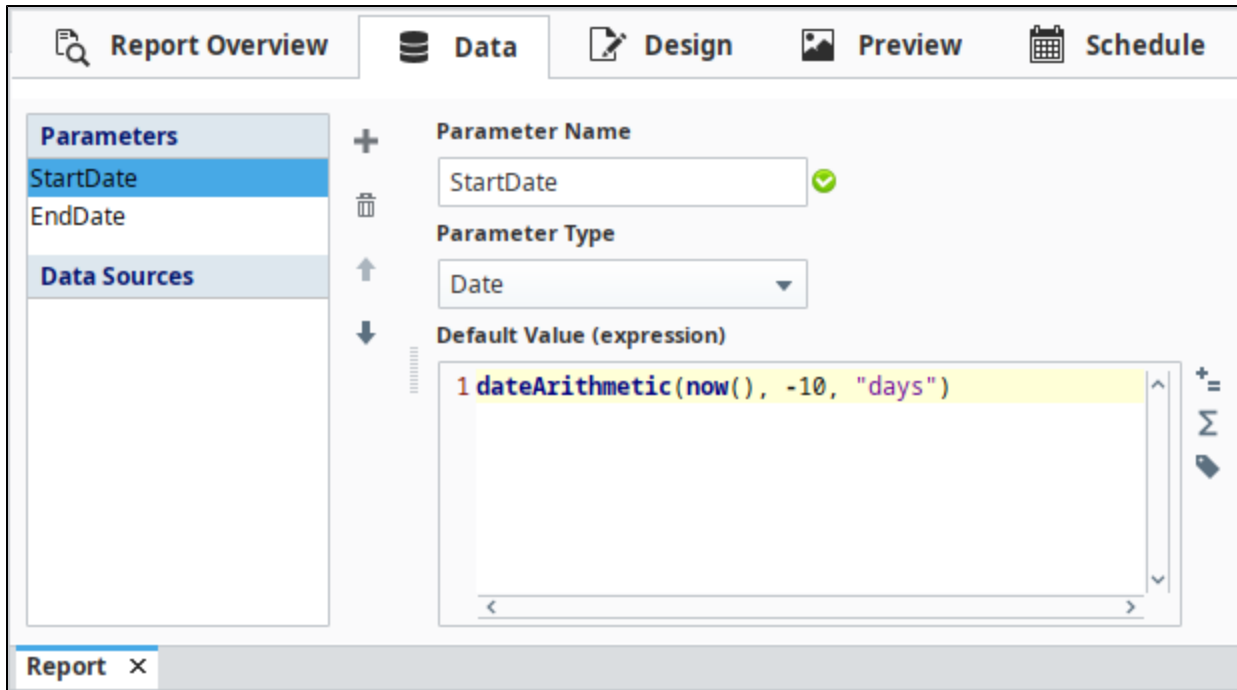
On this page ...

- [How Parameters Are Used](#)
- [Default Starting Parameters](#)
- [Creating New Parameters](#)



Parameters


[Watch the Video](#)

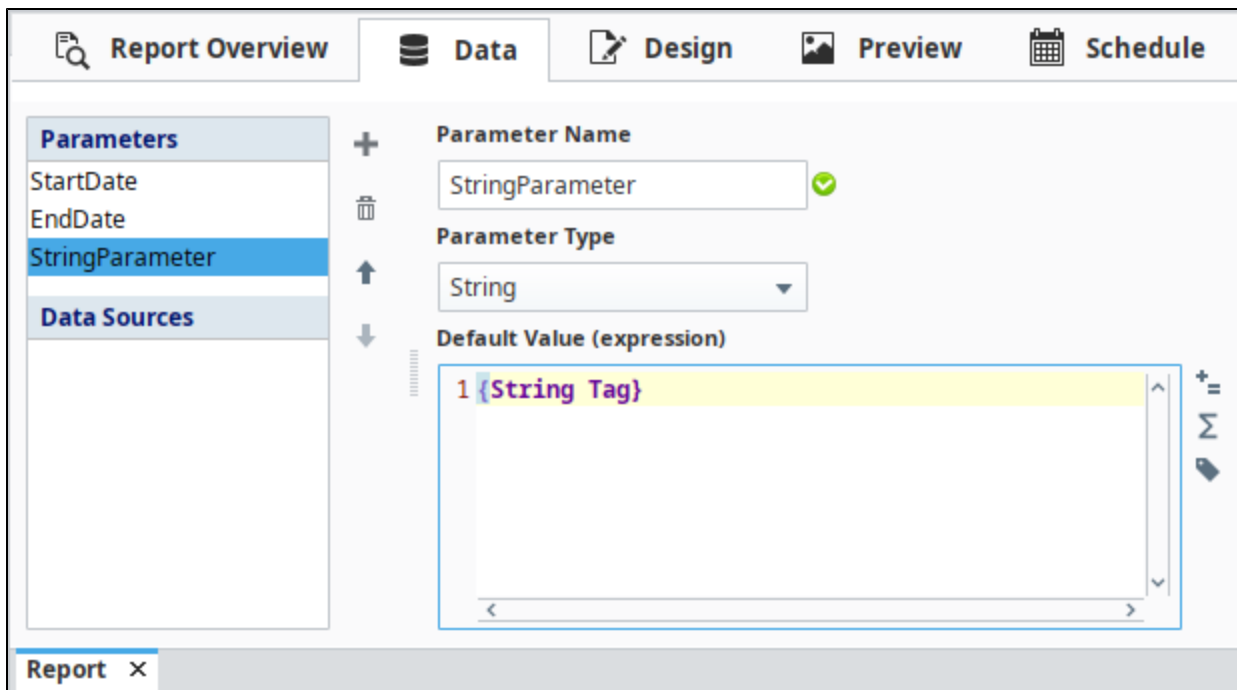


You can also delete both StartDate and EndDate parameters if you do not need date parameters in your report.

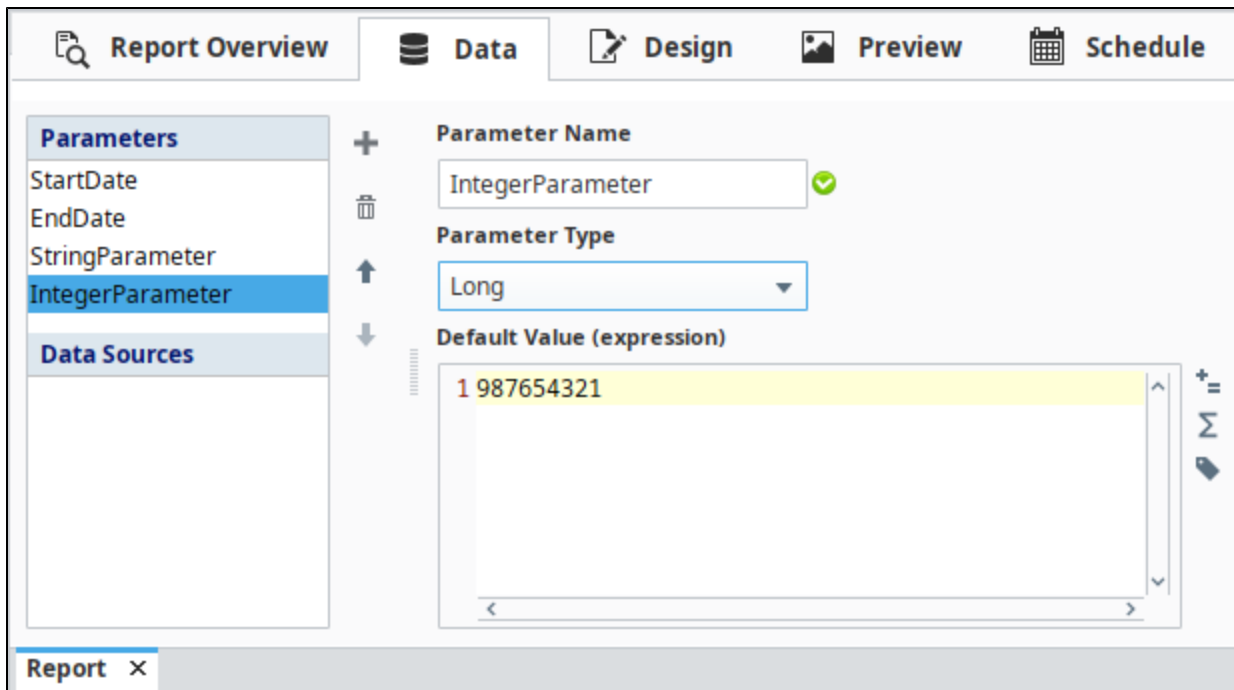
Creating New Parameters

New parameters may be added by clicking the **Add**  icon.

Once created, a default value should be given to the parameter. In addition to expressions, you can also have your default value reference a Tag using the **Tag**  icon on the right of the expression area.



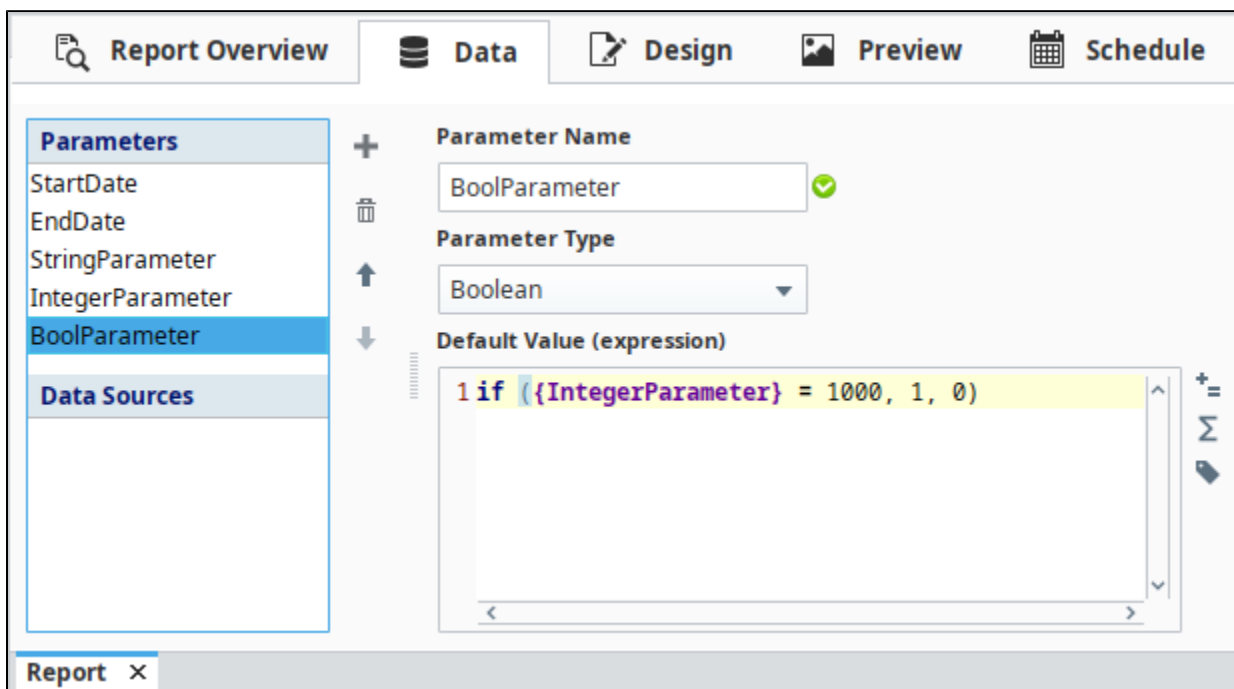
You can also use a literal value. Normal Expression languages syntax applies, so **dates** and **strings** must be wrapped in quotation marks, but **numerical** values can be written without quotes.



Finally, you can reference other parameters. In the example below, **BoolParameter** is referencing the value of **IntegerParameter**.

```
if ({IntegerParameter} = 1000, 1, 0)
```

It is important to note that when referencing other parameters, you must type the name of the parameter exactly within a set of curly braces { }, including capitalization. Also, you can only reference parameters that are above the current parameter. Thus, while **BoolParameter** can reference the **IntegerParameter**, the **IntegerParameter** may not reference **BoolParameter**.



Related Topics ...

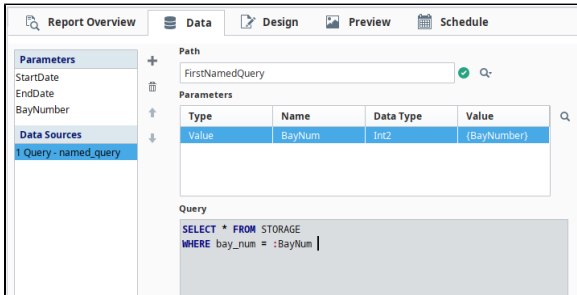
- [Basic SQL Query](#)

Named Query Data Source

Using the Named Query Data Source

Using the Named Query datasource is simple to configure provided you already have a [Named Query](#) created, and works in the same way that other Named Query bindings work. The Named Query report data source will execute the selected Named Query on the Gateway, and can use the [Report Parameters](#) as its parameters.

This data source supports [Nested Queries](#).



On this page ...

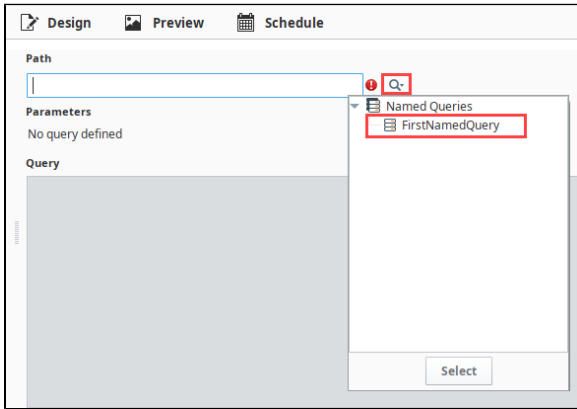
- [Using the Named Query Data Source](#)
- [Configuring a Named Query Data Source](#)

Named Queries in Reports

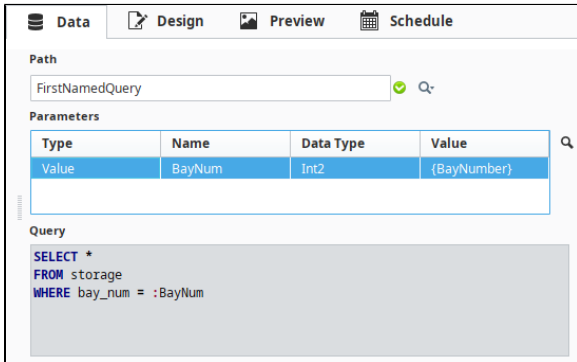
[Watch the Video](#)

Configuring a Named Query Data Source

First select a Named Query by clicking the **Selection** icon next to the Path field.



In the resulting list, select one of the configured named queries. The Named Query parameters will appear, along with a preview of the query.



One-by-one, **select** each parameter, and then click the **Selection** icon next to the **Parameters** table. This provides a popup of all report parameters that are available on the report. Select a Report parameter that will provide a value to the Named Query parameter. Once every parameter has a value, check the results in the Preview Panel.

Related Topics ...

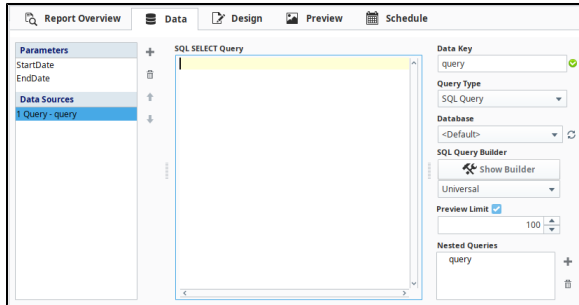
- [Named Queries](#)
- [Tag Calculation Query](#)
- [Report Charts](#)

SQL Query Data Source

Using the SQL Query Datasource

The SQL Query Data Source allows you to craft parameterized queries that run as a prepared statement. As Prepared Statements, these queries are more resistant to SQL injection offering additional security over basic queries.

The SQL Query type looks very similar to the Basic SQL Query type. It has a large text area where you can enter in a SQL select query. On the right, you have the option to rename the query, choose what database to run this query against, and add in a [Nested Query](#).



On this page ...

- [Using the SQL Query Datasource](#)
- [Parameters in SQL Query](#)
- [Crafting Queries with the Query Builder](#)
 - [Using the Builder](#)

SQL Query

[Watch the Video](#)

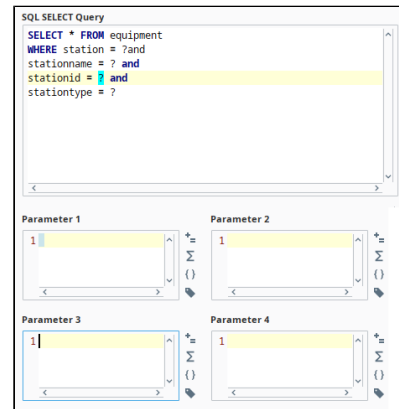
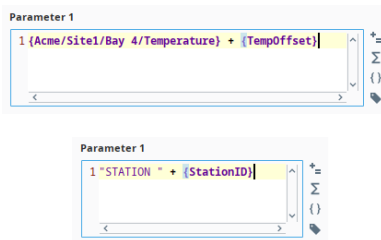
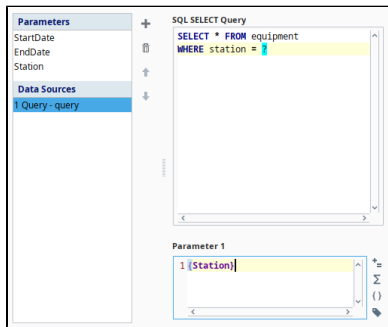
Parameters in SQL Query

Because the SQL Query Data Source runs as a prepared statement, passing parameters into this query type works a little differently.

Instead of placing a parameter within { } characters like the Basic SQL Query, we place a ? where we would like to pass in a parameter. Doing this will actually create a new text area below the query area for your parameter. This smaller text area is where you can pass a parameter into the query. You can pass in Tag values or Report Parameters, and since the parameter area allows expressions, you can use expressions to create any value you like from a combination of Tags and Report Parameters.

This is the preferred way to pass dates into your queries. You can use the {StartDate} and {EndDate} default Parameters directly in the Parameter fields.

You can add as many of these to a query as needed, with new parameter areas popping up underneath as they get added. To help keep track of what parameter corresponds to which "?" when entering in a value into the parameter area, the associated "?" will be highlighted.



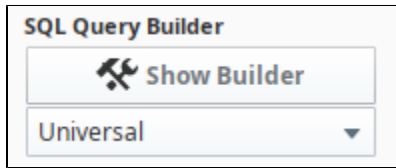
Crafting Queries with the Query Builder

The SQL Query data source includes the powerful SQL Query Builder tool. The Query Builder is a powerful Drag-and-Drop query building GUI that allows you to make complex queries from your connected databases. While a basic understanding of SQL helps make the most of the Query Builder

tool, most people will have no problem creating effective queries after a brief tutorial. The Query Builder is a third party tool that we brought into the Reporting Module. We go into detail on how to use it on the [Query Builder](#) page, but you can also check out the [Active Query Builder's documentation](#) for additional information.

Using the Builder

To activate the Query Builder in your **SQL Query** datasource type, start by selecting the SQL Syntax version from the drop down menu beneath the SQL Query Builder button. If your Database type isn't available (or you aren't sure), you can get most of the general functionality by selecting the **Universal** option. Then push the SQL Query Builder button to show the [Query Builder](#).



Related Topics ...

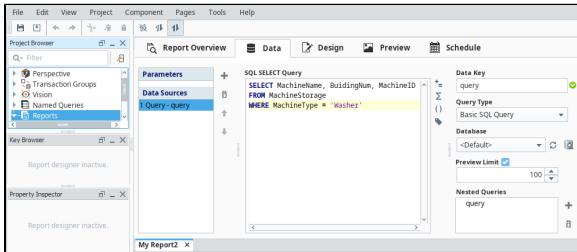
- [Tag Historian Query](#)
- [Report Parameters](#)

Basic SQL Query

This query type is the common type typically seen through much of Ignition before version 7.8. You can write queries which include Tag path references, expressions, or report parameters which resolve at run time.

You can enter the following in a static SQL query. The query will return rows that have a MachineType of 'Washer'.

```
SELECT MachineName, BuildingNum, MachineID
FROM MachineStorage
WHERE MachineType = 'Washer'
```



On this page ...

- [Report Parameters in a Basic SQL Query](#)
- [Working with Dates](#)
- [Create New Formatted Parameters](#)

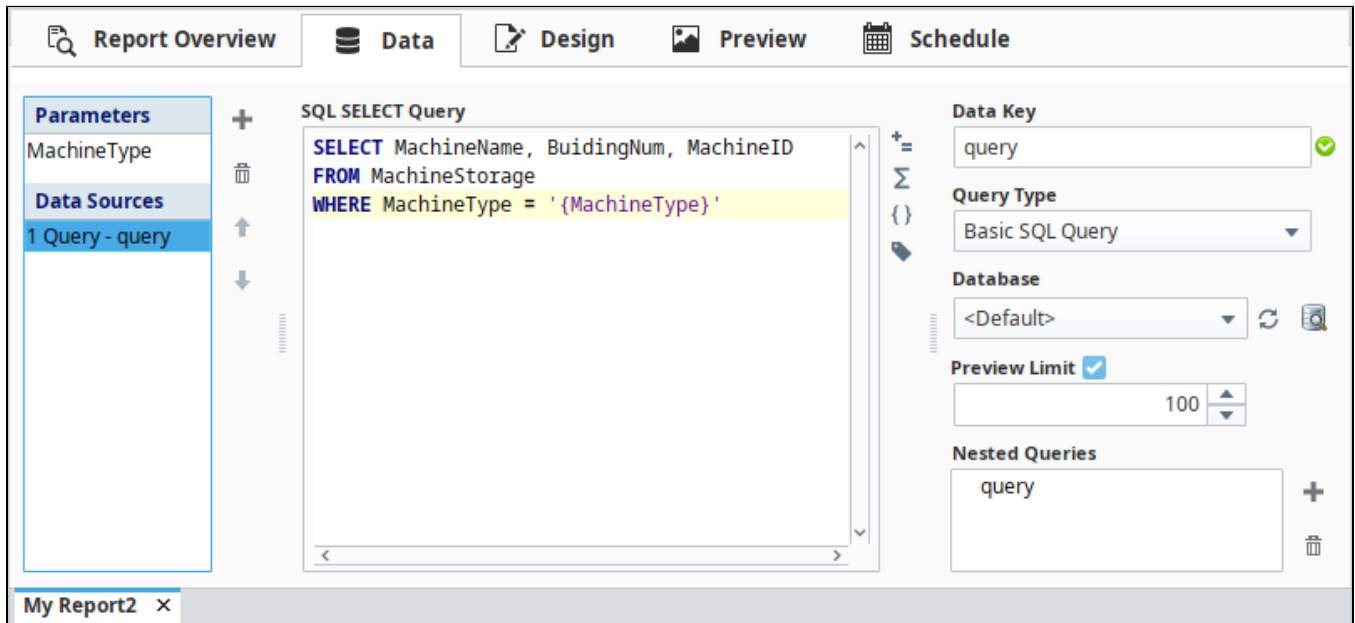


Basic SQL Query

[Watch the Video](#)

Report Parameters in a Basic SQL Query

Queries can also be made dynamic using things like report [parameters](#). To insert a report parameter, click the **Parameters** icon to the right of the query area and select your parameter. This allows for a more dynamic query, since new values can be passed into the parameter at runtime, giving the ability to change the type of machine this query is looking for.



Parameters are inserted directly into a report. This means that the datatype of parameter will affect how it should be referenced in the query. For example, since the parameter MachineType is a string, it will need a single or double quotes around it.

String Parameters

```
WHERE MachineType = '{MachineType}'
```

Since an integer does not need quotes around it, if your parameter is a Long, Double, or a Boolean, you can directly place the parameter in your query, without the quotes.

SQL - Long, Double, and Bool Parameters

```
WHERE MachineNum = {MachineNumber}
```

Working with Dates

If your parameter is a date object, then special consideration must be made.

The query will not accept a date object directly, it must first be converted to a string by putting quotes around it. However, database generally prefer datetime objects in very specific formats, such as **yyyy-MM-dd HH:mm:ss**. This means we need to reformat the date on any report parameters we want to pass to the Basic SQL Query. There are two main approaches to this:

Reformat the Date Parameters

By utilizing the expression language's [dateFormat\(\)](#) function, we can simply specify the format of the date.

SQL - Reformatted Date Parameter

```
dateFormat(  
    dateArithmetic(now(), -8, "hr"), //don't forget the comma at the end of this line...  
    "yyyy-MM-dd HH:mm:ss") //...as well as the outer closing ')'
```

The screenshot shows a report design tool interface with a top navigation bar containing 'Report Overview', 'Data', 'Design', 'Preview', and 'Schedule'. On the left, a 'Parameters' pane lists 'StartDate' and 'EndDate'. The main workspace shows the configuration for the 'StartDate' parameter: 'Parameter Name' is 'StartDate', 'Parameter Type' is 'Date', and 'Default Value (expression)' is a SQL query using the `dateFormat()` function. The expression is: `1 dateFormat(
2 dateArithmetic(now(), -8, "hr"), //don't forget the comma at the end of this line...
3 "yyyy-MM-dd HH:mm:ss") //...as well as the outer closing ')'`

Create New Formatted Parameters

In some cases, you may wish to leave the original "raw" Date parameter alone, and create a display-friendly version as a string.

To do this, simply make a parameter with type string and use the [dateFormat\(\)](#) expression on a date. In the image below, you can see that the **StartDate** parameter is used in a new **StartString** parameter. Additionally, an **EndString** parameter has been created that is using the **EndDate** parameter. This way, we can bind a calendar component directly to the StartDate and EndDate parameters and all the formatting will be done automatically in the report.

The screenshot shows the 'Data' tab of a report designer. On the left, there is a 'Parameters' list with 'StartString' selected. Below it is a 'Data Sources' list with '1 Query - query'. The main area shows the configuration for the selected parameter:

- Parameter Name:** StartString
- Parameter Type:** String
- Default Value (expression):** `1 dateFormat({StartDate}, 'yyyy-MM-dd HH:mm:ss')`

Note: This format was used with a MySQL database, so your database may take a different format. Refer to your database's documentation for suggested date formats.

Once the new string parameters have been created, we can then reference them in the Basic SQL Query just like a normal string.

```
SELECT *
FROM group_table
WHERE t_stamp BETWEEN '{StartString}'
AND '{EndString}'
```

The screenshot shows the 'Data' tab of a report designer. On the left, there is a 'Parameters' list with 'StartString' and 'EndString' visible. Below it is a 'Data Sources' list with '1 Query - query' selected. The main area shows the configuration for the selected data source:

- SQL SELECT Query:** `SELECT * FROM group_table WHERE t_stamp BETWEEN '{StartString}' AND '{EndString}'`
- Data Key:** query
- Query Type:** Basic SQL Query
- Database:** <Default>
- Preview Limit:** 100
- Nested Queries:** query

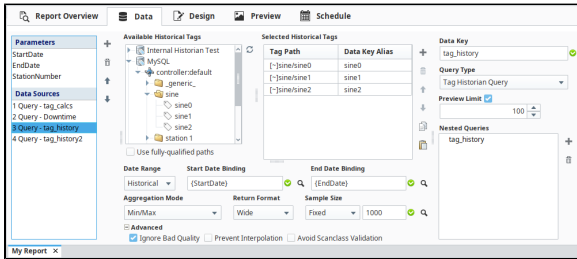
Related Topics ...

- [SQL Query Data Source](#)
- [Report Tables](#)

- [Report Charts](#)


Tag Historian Query

The Tag Historian Query provides a simple way to query data from Tag Historians. In the Tag Historian Query you can collect data from Historical Tags for specific date ranges, apply aggregates, and specify the sample size. It functions the same as a [Tag History Binding](#) and uses the same interface, so check out that page for more information on the individual properties.



On this page ...

- [Using Parameters in a Tag History Query](#)



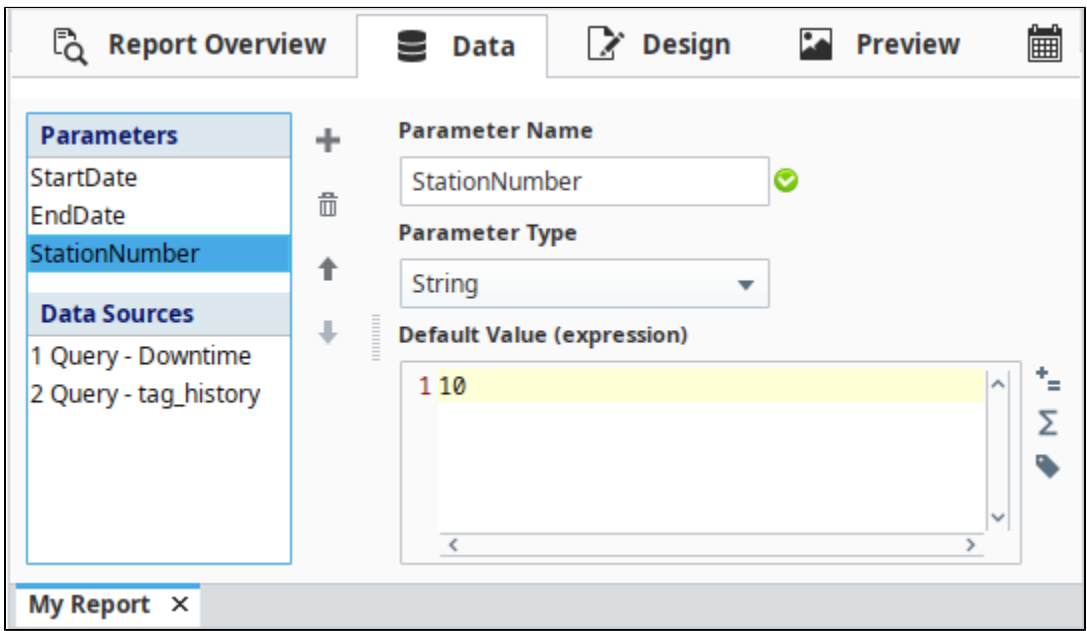
Tag Historian Query

[Watch the Video](#)

Using Parameters in a Tag History Query

The only difference between a Tag History Binding and the Tag History Query is when using indirection. In the Tag History Query, indirection is inserted directly into the Tag Path in the Selected Historical Tags section.

Assume we have a parameter named **StationNumber**:



Indirection is typically done using a parameter which like the rest of reporting, is the name of the parameter enclosed in { }. In the example below, we substituted the parameter '**StationNumber**' into my Tag Path for the number of the station. Note that you need to manually type out the name of the parameter, including exact spelling and capitalization.

Assuming base Tag Paths like the following:

```
[~]station 1/paint pressure  
[~]station 1/paint fill level
```

we can replace the "1" with a reference to **StationNumber**:
[~]station {StationNumber}/paint pressure,
[~]station {StationNumber}/paint fill level

The screenshot displays the 'Report Designer' interface with the following configuration:

- Parameters:** StartDate, EndDate, StationNumber
- Data Sources:** 1 Query - tag_calcs, 2 Query - Downtime, 3 Query - tag_history, 4 Query - tag_history2
- Available Historical Tags:** Internal Historian Test, MySQL, controller:default, _generic_, sine, station 1, paint fill level, paint pressure, station 2, speed
- Selected Historical Tags:**

Tag Path	Data Key Alias
[~]station {StationNumber}/paint fill level	paint fill level
[~]station {StationNumber}/paint pressure	paint pressure
- Date Range:** Historical
- Start Date Binding:** {StartDate}
- End Date Binding:** {EndDate}
- Aggregation Mode:** Min/Max
- Return Format:** Wide
- Sample Size:** Fixed, 1000
- Data Key:** tag_history2
- Query Type:** Tag Historian Query
- Preview Limit:** 100
- Nested Queries:** tag_history2

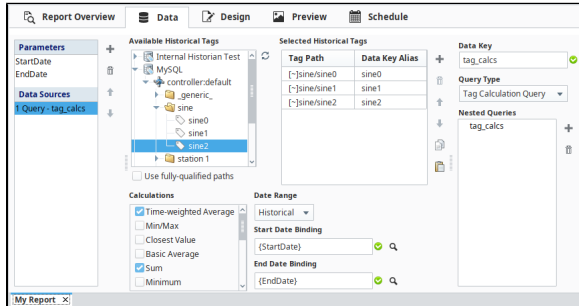
Click here to learn more about [Tag Historian Query Syntax](#) on the Gateway Configuration File Reference page.

Related Topics ...

- [Tag Calculation Query](#)
- [Report Charts](#)


Tag Calculation Query

Tag Calculations are executed by the Tag Historian, providing multiple calculated values for each Tag path. While the Tag Historian Query returns a history of values and only uses the aggregates to perform calculations on small chunks of data in the range of time, the Tag Calculation Query will aggregate, or run calculations, on the entire range to produce a single, calculated value per Tag Path. Assuming the following configuration:



On this page ...

- [Tag Historian vs Tag Calculation](#)
- [The 'Min/Max' Exception](#)



Tag Calculation Query

[Watch the Video](#)

We could show the query results on a table and view the following in the Preview Panel:

Average	Sum	Tag Path	Timestamp
0.28	1601.21	sine0	Mar 10, 2020
165.56	1260330.57	sine1	Mar 10, 2020
1.07	6024.47	sine2	Mar 10, 2020

Average	Sum	Tag Path	Timestamp
0.28	1601.21	sine0	Mar 10, 2020
165.56	1260330.57	sine1	Mar 10, 2020
1.07	6024.47	sine2	Mar 10, 2020

Tag Historian vs Tag Calculation

When deciding between a Tag Calculation Query and a Tag Historian Query, the main difference between the two is as follows:

- The Tag Calculation Query will always return a single row for each Tag Path, with multiple columns for each aggregate/calculation, except in the case of Min/Max (see below).
- Multiple calculations may be called on each Tag Path. This is not possible with a single Tag Historian Query.

The 'Min/Max' Exception

Since Min/Max returns two values (the minimum and maximum value over the range), this calculation will generate two rows per Tag Path. This extra row will appear in the underlying data, even if the MinMax key isn't used in a table or chart, as it is part of the underlying data.

If we added Min/Max to our example above, the preview panel would look like the following:

Average	Sum	Tag Path	Timestamp	MinMax
0.09	351.01	sine0	Mar 10, 2020	-50
165.71	1224266.37	sine1	Mar 10, 2020	321
0.12	379.52	sine2	Mar 10, 2020	100
0.09	351.01	sine0	Mar 10, 2020	50
165.71	1224266.37	sine1	Mar 10, 2020	10
0.12	379.52	sine2	Mar 10, 2020	-100

Average	Sum	Tag Path	Timestamp	MinMax
0.09	351.01	sine0	Mar 10, 2020	-50

165.71	1224266.37	sine1	Mar 10, 2020	321
0.12	379.52	sine2	Mar 10, 2020	100
0.09	351.01	sine0	Mar 10, 2020	50
165.71	1224266.37	sine1	Mar 10, 2020	10
0.12	379.52	sine2	Mar 10, 2020	-100

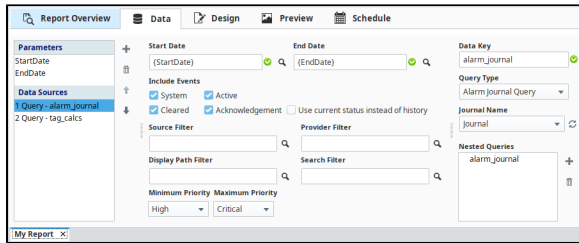
Note: The other calculations (Average and Sum) are simply duplicated for both MinMax rows.

Related Topics ...

- [Alarm Journal Query](#)
- [SQL Bridge Module](#)

Alarm Journal Query

The Alarm Journal Query data source is a simple way to access Alarming data within a report. It can pull from both the alarm journal data in the database as well as the current live events that would show up in the Alarm Status Table.



On this page ...

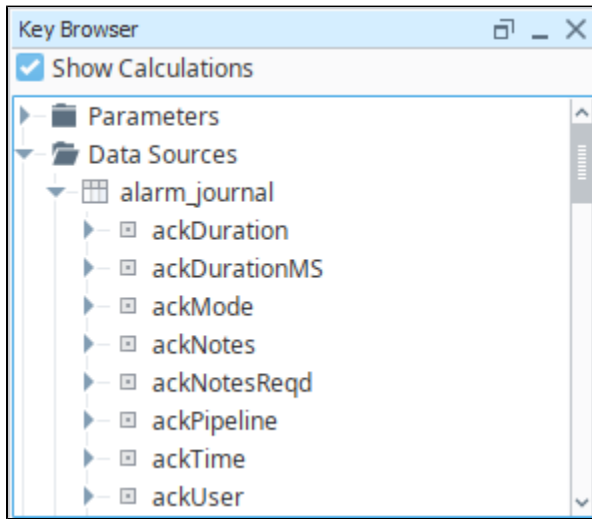
- [Properties](#)



Alarm Journal Query

[Watch the Video](#)

Once created, the new Data Source will expose many alarming related keys in the Design Panel's Key Browser.



Properties

The Alarm Journal Query has a few properties that can be configured to filter the types of events you see in the returned dataset.

Property	Description
Start Date and End Date	The time range from which to pull Alarms from. Start Date being the furthest date in the past, while End Date is the most recent date.
Include Events	Allows you to select which type of events will be returned. The options are: <ul style="list-style-type: none"> • System • Cleared • Active • Acknowledgment <p>There is also a Use current status instead of history checkbox. As the text implies, this will use the current alarm status instead of querying from an Alarm Journal profile. Enabling this checkbox will cause the data source to ignore the Journal Name property on the right.</p>

Source Filter	Will filter the alarms based on the source path. Multiple filters can be specified by separating them with a comma, and they accept the wildcard (*) symbol. Works in the same way that the filters on the Alarm Status Table component work.
Provider Filter	Will filter the alarms based on the Tag Provider that they originate from. Can specify multiple filters by separating them with a comma.
Display Path Filter	Will filter the alarms based on the display path. Multiple filters can be specified by separating them with a comma, and they accept the wildcard (*) symbol. Works in the same way that the filters on the Alarm Status Table component work.
Search Filter	Will filter alarms by searching for the given string in both the display path and source path. Accepts comma separated paths.
Minimum Priority and Maximum Priority	The minimum and maximum priority of alarms that will be returned.
Journal Name	Located on the right side, this property allows you to specify the Journal that will be used for the query.

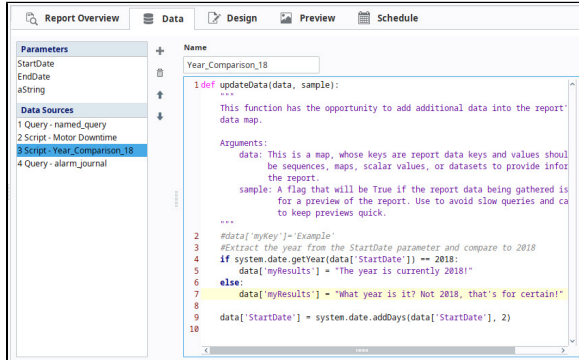
Related Topics ...

- [Static CSV](#)
- [Report Tables](#)

Scripting Data Source

Script Data Source

The Script data source allows you to use [scripting](#) to add additional data into a report, or modify existing data. With this data source, you can pull in data from the database and then modify it using a script before pushing it out as a data key for use in the report.



On this page ...

- [Script Data Source](#)
- [Accessing Parameters and Data Sources](#)
 - [Creating a New Key](#)
 - [Parameters](#)
 - [Data Sources](#)
 - [Nested Queries](#)



Scripting Data Source

[Watch the Video](#)

Accessing Parameters and Data Sources

One of the main uses of the Scripting Data Source is to allow for data access and manipulation as the report is being generated, allowing you to replace the original results, or add new additional content.

The syntax for creating or referencing a data key in the Script datasource is described below:

Pseudocode - Data Key Syntax

```
data[ 'keyName' ]
```

Order Matters

As mentioned on the [Report Data](#) page, the order of Data Sources determines which parameters and data sources they may reference. Because of this, it is **highly recommended** that Scripting Data Sources are placed at the bottom of the Data Sources list.

Creating a New Key

To create a new key that the report can use:

Python - Creating a New Key

```
data['newKey'] = "This key was generated on the fly!"
```

The type of data assigned to the key determines where it appears in the Design Panel.

- Simple data types, like strings and integers, will appear as **Parameters**.
- Datasets will appear as **Data Sources**.

Report Data

The screenshot shows the 'Data' tab in the Report Designer. On the left, there are two lists: 'Parameters' and 'Data Sources'. The 'Parameters' list contains 'StartDate', 'EndDate', and 'aString'. The 'Data Sources' list contains '1 Query - named_query', '2 Script - Motor Downtime', '3 Script - Year_Comparison_18', and '4 Query - alarm_journal'. The 'aString' parameter is selected. On the right, the configuration panel for the selected parameter shows: 'Parameter Name' is 'aString', 'Parameter Type' is 'String', and 'Default Value (expression)' is '1 \"I'm a string!\"'.

Key Browser

The Key Browser window shows a tree view of the report's structure. It has a 'Show Calculations' checkbox at the top. The tree is organized into folders: 'Parameters' (containing StartDate, aString, and EndDate), 'Data Sources' (containing alarm_journal and myResults), and 'Built In'.

String Data Sources can be given a name using the Name field.

The screenshot shows the 'Data' tab in the Report Designer. On the left, there are two lists: 'Parameters' and 'Data Sources'. The 'Parameters' list contains 'StartDate', 'EndDate', and 'aString'. The 'Data Sources' list contains '1 Query - named_query', '2 Script - Motor Downtime', '3 Script - Year_Comparison_18', and '4 Script - filtered_dataset'. The '2 Script - Motor Downtime' data source is selected. On the right, the configuration panel for the selected data source shows: 'Name' is 'Motor Downtime'. Below the configuration panel is a code editor with a Python function definition:

```
1 def updateData(data, sample):  
    """  
    This function has the  
    data map.  
  
    Arguments:  
        data: This is a map  
        be sequences
```

Parameters

Where the 'keyName' is the name of your data key. Thus, reading the value of a parameter, such as the initial StartDate parameter, can be accomplished by using `system.date.getYear()` and an if-statement.

Python - Accessing a Parameter's Value

```
# Extract the year from the StartDate parameter and compare to 2017.
if system.date.getYear(data['StartDate']) == 2017:
    # Do work here if the year lines up.
```

Of course, we can write back to the key and override its value:

Python - Overriding the Default StartDate Parameter

```
# This line would override the default StartDate parameter by add adding a day.
data['StartDate'] = system.date.addDays(data['StartDate'], 1)
```

Additionally, this allows you to expand the Accessing a Parameter's Value example above by creating a new key:

Python - Accessing a Parameter's Value

```
# Extract the year from the StartDate parameter and compare to 2017.
if system.date.getYear(data['StartDate']) == 2017:
    # Create a new key and assign it one value if our condition is true...
    data['myResults'] = "The year is currently 2017!"
else:
    # ...or assign a different value if the condition is false. This way we can always assume the key
    'myResults' exists.
    data['myResults'] = "What year is it? Not 2017, that's for certain!"
```

Data Sources

Static CSVs

While uncommon, Static CSVs may be accessed in a Scripting Data Source. The syntax is similar to working with Parameters.

Python - Static CSV example

```
# Take a Static CSV data source, and replicate its contents in a new key.
data['static_data'] = data['Area Data']
```

Query-Based Data Sources

Reading the contents of a query-based Data Source, such as a [SQL Query Data Source](#) or [Tag Historian Query](#), requires the `getCoreResults()` function, which returns the results in a [standard dataset](#):

Python - Accessing a Query-Based Data Source's Value

```
# Query-based Data Sources are slightly different than parameters, so we must use getCoreResults() to
extract the data.
rawResults = data['keyName'].getCoreResults()

# getCoreResults() returns a dataset, so we can utilize getValueAt() and rowCount within our scripts.
resultsCount = data['keyName'].getCoreResults().rowCount
```

When working with data sources, it is unusual to attempt to write back, since datasets are immutable. Instead, the preferred approach is to create a new key with the modified results.

Say we have a query data source named "Area Data" which contains four columns: **month**, **north_area**, **south_area**, and **t_stamp**. If we need to build a new data source without the **t_stamp** column, we can use the following code:

Python - Building a New Data Source

```
# build a header and initialize a pydataset
header = ['month', 'north_area', 'south_area']
filteredDataset = []

# get the results from the Area Data data source
rawDataset = data['Area Data'].getCoreResults()

# build the new pydataset out of only some of the Area Data's data keys
for row in range(rawDataset.rowCount):
    valCategory = rawDataset.getValueAt(row, 'month')
    valNorthArea = rawDataset.getValueAt(row, 'north_area')
    valSouthArea = rawDataset.getValueAt(row, 'south_area')
    filteredDataset.append([valCategory, valNorthArea, valSouthArea])

# convert the pydataset to a standard dataset
filteredDataset = system.dataset.toDataSet(header, filteredDataset)

# create a new data source with the filtered results
data['updated Area Data'] = filteredDataset
```

The screenshot displays a software interface with a top navigation bar containing icons and labels for 'Report Overview', 'Data', 'Design', 'Preview', and 'Schedule'. On the left side, there is a 'Parameters' section with fields for 'StartDate', 'EndDate', and 'aString', and a 'Data Sources' section listing four items: '1 Query - named_query', '2 Script - Motor Downtime', '3 Script - Year_Compariso...', and '4 Script - filtered_dataset' (which is highlighted in blue). The main workspace is titled 'Name' and contains a text input field with 'filtered_dataset'. Below this, a code editor shows a Python function named 'updateData' with the following code:

```
1 def updateData(data, sample):
2     """
3     This function has the opportunity to add additional data into the report's
4     data map.
5
6     Arguments:
7     data: This is a map, whose keys are report data keys and values should
8         be sequences, maps, scalar values, or datasets to provide informat:
9         the report.
10    sample: A flag that will be True if the report data being gathered is
11        for a preview of the report. Use to avoid slow queries and calcul:
12        to keep previews quick.
13    """
14
15    # build a header and initialize a pydataset
16    header = ['month', 'north_area', 'south_area']
17    filteredDataset = []
18
19    # get the results from the Area Data data source
20    rawDataset = data['Area Data'].getCoreResults()
21
22    # build the new pydataset out of only some of the Area Data's data keys
23    for row in range(rawDataset.rowCount):
24        valCategory = rawDataset.getValueAt(row, 'month')
25        valNorthArea = rawDataset.getValueAt(row, 'north_area')
26        valSouthArea = rawDataset.getValueAt(row, 'south_area')
27        filteredDataset.append([valCategory, valNorthArea, valSouthArea])
28
29    # convert the pydataset to a standard dataset
30    filteredDataset = system.dataset.toDataSet(header, filteredDataset)
31
32    # create a new data source with the filtered results
33    data['updated Area Data'] = filteredDataset
```


Nested Queries

What if our 'Area Data' query has a nested query called 'Area Details' that we would like to manipulate in a script? This is useful when using [Table Groups](#).

Python - Script data source for nested query

```
nested = data['Area Data'].getNestedQueryResults()    # Gets results from our parent query
subQuery = nested['Area Details']    # Gets the subquery we want -- there can be more than one
header = ['productName', 'cost', 'triple']
alteredDataset = []
for child in subQuery:
    children = child.getCoreResults()    # children is a dataset
    for row in range(children.rowCount):
        valProductName = children.getValueAt(row, 'productName')
        valCost = children.getValueAt(row, 'cost')
        valTimesThree = None
        if valCost != None:
            valTimesThree = 3 * valCost
        alteredDataset.append([valProductName, valCost, valTimesThree])

# convert the pydataset to a standard dataset
alteredDataset = system.dataset.toDataSet(header, alteredDataset)

# create a new data source with the altered results
data['Updated Area Details'] = alteredDataset
```

Static CSV

The Static CSV data source allows you to quickly craft a data source to use in your report. This data source is ideal when you need some test data to begin creating a new report. All values are brought in as strings, which means that you don't need quotation marks around any of the string values unless the value needs to include a comma.

As the name implies, this Data Source is strictly static, and does not have any built-in means to update or import new values in. In these scenarios, you may want to look into the [Scripting Data Source](#) to import CSV directly into the report, or first store the data into a SQL database and retrieve the results with a [SQL Query Data Source](#) data source.

Format

The first line of the Static CSV will be a list of columns separated by commas. Each subsequent line after that will then be a row in the dataset, with each value separated by a comma as well. See the text below for an example dataset that has three columns and thirteen rows. You can double click the block to copy its text.

CSV - Creating a Custom Static CSV

```
Equipment, Time, Site
Motor, 15, Site A
Motor, 23, Site A
Conveyor Line, 148, Site B
Pallet Wrapper, 58, Site A
Motor, 96, Site C
Conveyor Line, 23, Site B
Palletizer, 40, Site B
Conveyor Line, 56, Site A
Pallet Wrapper, 45, Site C
Motor, 43, Site C
Conveyor Line, 87, Site D
Motor, 23, Site D
```

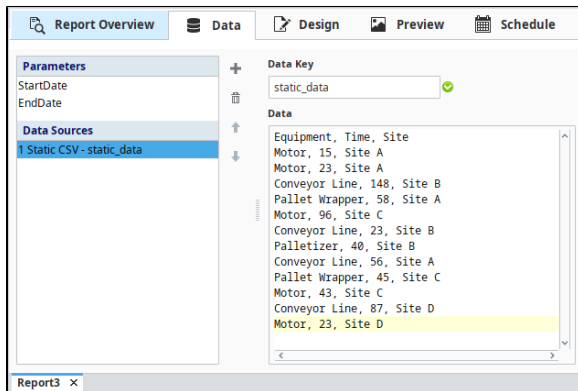
On this page ...

- [Format](#)



Static CSV

[Watch the Video](#)



Related Topics ...

- [Nested Queries](#)

Nested Queries

What Are Nested Queries?

The simple definition is that a *Nested Query* uses the results of a previously executed query to collect data. The general structure of a Nested Data Source is one in which you have a Parent query and child queries. Each child query can then use the columns from the parent query as parameters using {columnName}. For example, if a parent query had a column called machineID, the child query can then use {machineID} as a parameter, and the child query will run for each row of the parent query, using the different values of machineID for each run. Those well-versed in SQL are probably thinking that this sounds like a JOIN and in fact, there are some similarities. There are also some major differences which allow Nested Data to be both easier and more powerful:

- Nesting relationships are not restricted to data in a single schema, database or even source! Nesting is easy to configure across tables, between different databases, or even with sources like the Tag Historian!
- Writing queries for nested query sources can be far simpler and easier to maintain than writing complex JOIN operations.
- Nested structures allow more control in how data is collected, allowing data structures and relationships that are more expressive.

On this page ...

- [What Are Nested Queries?](#)
- [How Nesting Works](#)
- [Special Considerations](#)
- [Nested Query Example - Equipment Downtime](#)



Nested Queries

[Watch the Video](#)

How Nesting Works

Let's use a simple data relationship to help illustrate how nesting occurs. Imagine we have data collected from two unrelated sources that look the ones seen in this table.

Codes		Frequency		
CodePK	Code	FrequencyID	CodeID	FreqValue
1	ZG	1	3	11
2	GB	2	1	41
3	DC	3	2	13
		4	3	26
		5	1	13
		6	3	32
		7	1	11

We want to create a data source connecting all these things for reporting using nesting. The trick is to identify where the two datasources connect. You may notice in the data above The CodePK column in the Codes datasource matches up with the CodeID column of the Frequency datasource. This is where we will connect the two datasources. We will make our Codes datasource the parent and the Frequency the child. The queries would look something like below.

Pseudocode - Select Statement Examples

```
-- Parent Query
SELECT CodePK, Code FROM Codes
```

```
-- In the Child Query, accessed by clicking on the Frequency leaf of the Nested Queries Tree
-- This assumes the value of Parameter 1 would be equal to {CodePK}
SELECT FrequencyID, FreqValue FROM Frequency WHERE CodeID = ?
```

The screenshot shows a database query tool interface with the following components:

- SQL SELECT Query:** A text area containing the query:


```
SELECT
  FrequencyID
  ,FreqValue
FROM
  Frequency
Where
  CodeID = ?
```
- Data Key:** A dropdown menu set to "Frequency" with a green checkmark.
- Query Type:** A dropdown menu set to "SQL Query".
- Database:** A dropdown menu set to "<Default>" with a refresh icon.
- SQL Query Builder:** A section with a "Show Builder" button and a dropdown menu set to "Universal".
- Preview Limit:** A checkbox checked and a numeric input set to "100".
- Parameter 1:** A list containing "1 {CodePK}" with a yellow highlight.
- Nested Queries:** A tree view showing a parent query "Codes" with a child query "Frequency" selected.

What this means for our resulting data is that the parent query is called first, and a set of results is returned to the parent query, named Codes. After this data has been retrieved, the Child query will execute, once for each row of the parent, substituting the value of CodePK into the child query where we have the {CodePK} reference.


The resulting data will have a structure like this:

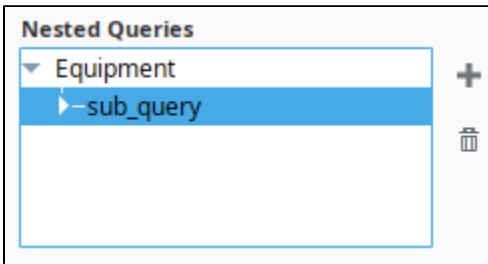
Pseudocode - Code Frequency Structure

```
Codes
|-Row 1
|  |_- CodePK - 1
|  |_- Code   - ZG
|  |   Frequency
|  |       |-Row 1
|  |       |_- FrequencyID - 2
|  |       |_- FreqValue  - 41
|  |       |-Row 2
|  |       |_- FrequencyID - 5
|  |       |_- FreqValue  - 13
|  |       |-Row 3
|  |       |_- FrequencyID - 7
|  |       |_- FreqValue  - 11
|-Row 2
|  |_- CodePK - 2
|  |_- Code   - GB
|  |   Frequency
```



```
SELECT id AS EquipmentIDNumber,  
       Name AS EquipmentName,  
       Description AS EquipmentDescription  
FROM Equipment_Table
```

6. On the right of the Nested Query section, click the **Plus**  icon to create a sub_query.



7. Select the sub_query, and rename the Data Key property to rename it. In this example, we will use the name **EquipDowntime**.
8. Type the following query into the query area.

SQL - Downtime Table Query

```
SELECT cause AS DowntimeCause,  
       minutes_down AS DowntimeMinutes  
FROM Downtime_Table  
WHERE Equipment_id = ?
```

Here we pull in our downtime table, but we only need the cause and minutes_down, since we are already grabbing the Equipment_id from the first query. The WHERE clause is where we link this query to the parent with the equipment ids.

9. In the **Parameter 1** field, type the following:

Expression Language - Referencing a Parameter

```
{EquipmentIDNumber}
```

Note that we are directly referencing one of the aliased column names from the **Equipment** query.

10. That is all it takes to make a Nested Query! We now have two separate tables being called and linked together by the equipment id.
11. To check that your queries worked, go to the Preview tab and look at the XML data that comes up. You should have a data set inside (indented) each row of Equipment data.

If you'd like, you can continue with this example and either use this data [in a table group](#), or in a [nested chart](#).

Related Topics ...

- [Table Groups](#)
- [Charts Inside of Tables](#)

Report Design

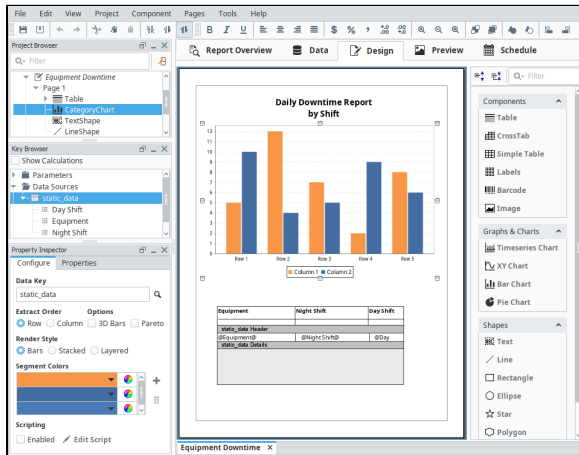
Report Designer Interface

The Report Workspace's Design tab lets you design reports with the same intuitive feel, familiar layout, and drawing tools that you get when designing windows or views in Ignition. Just like designing Vision windows and Perspective views, you can design reports using the same familiar drag-and-drop method and choose from a library of graphs, charts, tables, and images.

Looking at the screenshot of the Report Designer, you'll notice that an open Report Resource in the **Project Browser** can be expanded to provide information at a glance. This tree lets you visualize the relationship and hierarchy of Report Design elements on the page. Find an element in the tree by selecting items on the Design Panel, or find an element on the page by selecting it from the tree. As you do so, you'll notice that the bottom left of the default Report Designer will change to provide configure panels and/or property tables depending on the selected item.

Just below the Project Browser is the **Key Browser**, which provides the **Data Keys** we use to reference data in our report. The Key Browser is home to all your Data Sources, **Parameters**, as well as a number of built-in calculation keys to speed you through the report design process. When you configure a data source and switch to the Design tab, a sample of all your queries is run on the Gateway to get information about the structure of your data. Columns in your data are represented as children in the Key Browser tree.

The right side of the panel is where the component palette lives, providing a number of powerful components, charts, and shapes which are used to build the layout and visualization of your report.



On this page ...

- [Report Designer Interface](#)
- [Report Design Components](#)
 - [Report Objects](#)
 - [Reporting Components](#)
 - [Reporting Charts](#)
 - [Shapes](#)
- [Selection and Alignment](#)
 - [Super Selection](#)
 - [Multiple Selection](#)
 - [Resizing and Moving Objects](#)
 - [Alignment](#)
 - [Shift Drag](#)



Report Design Tab

[Watch the Video](#)

Report Design Components

The Design Panel has a Report Design Palette with a host of components, charts, and shapes that help you design and create meaningful, informative, and professional reports. **Tables** are used to collect and store varying amounts of data. Sophisticated charts can be created from the data collected in a table. **Images** and shapes can add that extra touch of class and polish making a report stand out.

Just like designing in Vision windows and Perspective views, Report Design components have properties which are editable in the Property Inspector which is located on the lower left side of the Design Panel. Below is a brief overview of the report design tools you can choose from for building your reports.

Some property changes don't take effect in the Design panel

Note that modifying some component properties, such as the **Font** or **Fill Color**, will immediately cause a visual change in the **Design Panel**.

However, other properties, such as the **Date Format**, **Number Format**, and **Overflow Behavior** properties on a Text Shape component will not cause an immediate change. The properties described here wait until the report executes. Because of this, you will only see the results of these changes in the **Preview Panel**.

It is highly recommended that you review your report in the Preview Panel often. This provides an opportunity to visualize what the resulting report will look like.

Report Objects

Report Objects are the actual report and page components, and each have their own properties. They have a host of settings and properties that allow you to customize the appearance of your reports based on your requirements.

- **Report Object** - Allows you to set the report properties for paper size, dimensions, orientation, and margins.
- **Page Object** - Allows you to add and remove pages in your report, as well as configuring additional properties that can add that creative touch to your reports.

Reporting Components

The Reporting Components are the foundation for creating useful reports. There are variety of different tables that you can choose from based on the type of data you're collecting. There are even Label and Barcode components you can use to create shipping and product labels. Here is a brief description of all the report design components.

- **Table** - Allows you to display tabular data in a variety of ways using the Table properties. Tables are configurable and highly flexible giving Reporting users the ability to flexibly layout and organize tabular data. They also support advanced grouping and pagination.
- **CrossTab** - Similar to a Table component, the CrossTab component is commonly used to summarize the relationship between two categories of data by showing summaries of cross sections of the data source. The CrossTab component creates a dynamic pivot table from your data.
- **Simple Table** - A basic grid table that dynamically creates new rows and columns for rows returned by the Data Keys on the component.
- **Labels** - Used to print out a host of different types of labels such as mailing labels. You can create a set of labels using preset or custom sizes.
- **Barcodes** - Allows reports to contain dynamically generated barcodes. The following formats are supported: Code 128, Code 39, QR Codes, PDF 417, Aztec, Data Matrix, to name a few. Refer to [Barcode](#) in the Appendix section to see all the supported formats.
- **Images** - Can be embedded into reports. They can be dropped onto reports, pulled as binary data from a data source, or even a URL.

Reporting Charts

[Report charts](#) allow you to display your data in a variety of graphical ways. There are four types of charts so you can show the data any way you like. Below is a brief description of the different types of charts you can choose from to build your reports.

- **Timeseries Chart** - A type of line chart whose domain, or X-Axis, represents a timestamp or datetime. The Timeseries Chart is a great way to display Tag History.
- **XY Chart** - Similar to the Timeseries chart with the main difference being the domain, or X-Axis which is a numeric value and not a date.
- **Bar Chart** - A very easy chart to use that provides a bar representation of any numeric values, and because it has many properties, it can greatly impact how the data is presented. It is useful for displaying series with relative values.
- **Pie Chart** - Displays items as pie wedges. It represents relative quantities as wedges of a circle.


Shapes

You may have seen some of these shapes when designing your Vision windows and Perspective views. The shapes you see in the Report Design Palette are shape tools. They are not dragged from the design palette on to your report, instead you select a shape tool and begin to draw your shape on to your report. You create its size, height, and shape. Some of the shape tools allow you to create various shapes as well as edit them. Just like components, once a shape is created, you can change its fill color, stroke color, and stroke style. Each shape has its own properties. Here is a list and brief description of the available shapes in the Report Design Palette.

- **Text Shape** - Creates a text area for static or data key bound content.
- **Line** - Creates a straight line. It can run north-south, east-west, or diagonally.
- **Rectangle** - Creates a square or a rectangle. Once a rectangle or square is created, you can use the handles to change its height and width.
- **Ellipse** - Creates circles and ellipses. Once an ellipse or circle is created, you can use the handles to change its size.
- **Star** - Creates a star or a polygon. Once the star is created you can use the handles to resize it.
- **Polygon** - Creates custom polygon shapes by drawing and connecting the lines. Once all the lines are connected to form the polygon shape, you can use the handles to change its size. Double click on any of the individual lines to change the size and shape of the selected line.
- **Pencil** - Draws freehand shapes with smooth paths.

To learn more about report design components, refer to the [Report Design Tools](#) section, and to the Appendix for details on each component.

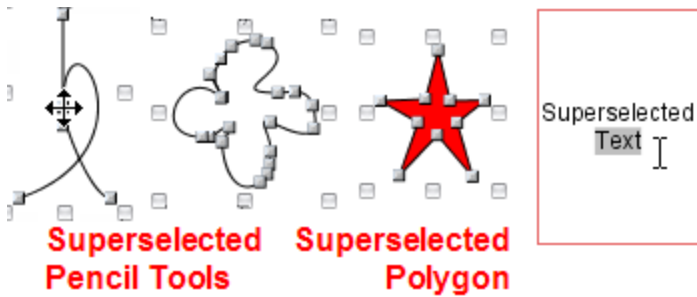
Selection and Alignment

Selecting components is done with the selection tool (), which is the default tool in the Report Designer whenever another tool is not activated.

Reporting has a "deeper" selection model than the Ignition Designer. Simple object selection is done by single clicking an object, and is typically used to move an object around or resize it. "Selecting deep" is done by double-clicking on the object 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. Visualizing selection is simplified by viewing the Project Browser to see which node in the Report Resource tree is selected.

Super Selection

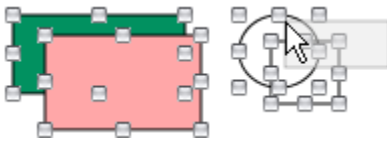
Super Selection refers to an editing state that some shapes go into when double clicked. Text is the most common of these. When a text box is selected you can move and resize it. When it's super-selected, you can place the text cursor or select a range of characters and insert or delete text. The polygon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



Multiple Selection

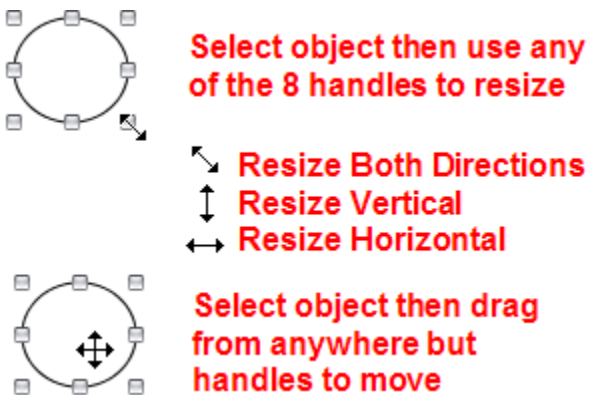
Multiple Selection of components can be done two ways:

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



Resizing and Moving Objects

To resize an object, first select it with a single click, then left or right click, and drag one of the 8 resizing handles. To move an object, left or right click on the shape, and drag it anywhere on the report. Both resizing and moving operations support shift-dragging.



Alignment

Alignment is accomplished by selecting multiple objects, then choosing from any of the **Make** commands in the [Component Menu](#) to align objects. You can align objects in rows by their top, center or bottom border, and align columns by their sides or center. You can make objects the same size (width and height), and equally space rows and columns horizontally and vertically.

In report design, z-order defines relative order of objects when they overlap. Select the object and click **Bring to front** or **Send to back** in the Component Menu to reorder the objects.

Shift Drag

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

[Related Topics ...](#)

- [Tutorial: The Report Workflow](#)

- [Report Charts](#)
- [Report Tables](#)

In This Section ...

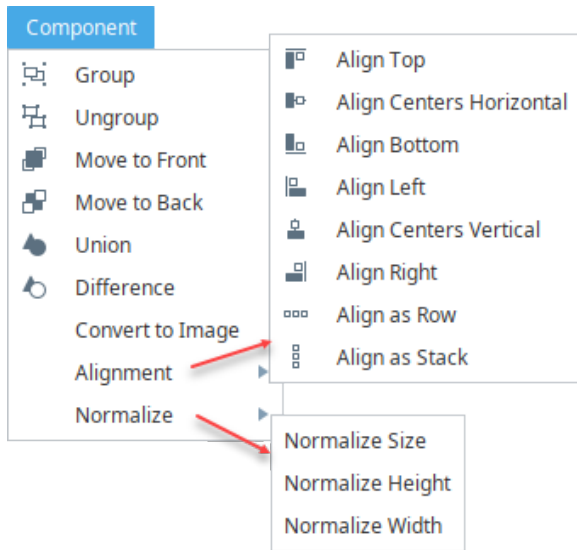
Report Design Tools

Object layout is an important aspect in creating professional reports. The Reporting Module provides a host of intuitive tools and functions to help you design and edit reports. Ignition Reporting uses a WYSIWYG (what you see is what you get) approach.

Report Menus

The Ignition Designer menubar provides quick access to many common Reporting Design functions. The Component and Pages menu items are added to the menubar when the Report Design Panel is open. Many of the menus, such as **File**, **Edit**, and **Tools** are similar to other areas of the Designer. Reporting-specific Toolbar and Menu functions are described below.

Component Menu



The Component menu allows you to modify the layout of objects in a report. To perform any of the Component menu functions, you must first select an object or multiple objects before you're able to perform the function.

Menu Item	Function
Group/Ungroup	Grouping allows you to merged selected shapes and makes them behave as one with respect to: selection, moving, and resizing. To "drill down" to individual objects, superselect the grouped object. Ungroup separates grouped shapes.
Move to Front /Move 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 move a selected shape either forward or backward. This is also known as the Z Order.
Union/Difference 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 to Image	Converts the report page to an image.
Align Selected Items - Make Row Top/Center /Bottom	Quickly align several shapes in a row, either by their top, center, or bottom border. This is particularly useful when shapes are of different heights.
Align Selected Items - Make Column Left /Center/Right	Same as above, but for columns, aligning their sides or center.
Align as a Row /Stack (column)	Equalizes the distance between shapes horizontally or vertically.

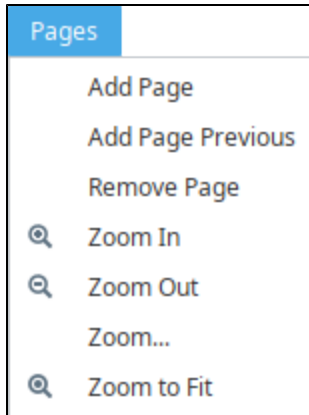
On this page ...

- [Report Menus](#)
 - [Component Menu](#)
 - [Pages Menu](#)
- [Report Design Toolbar](#)
- [Shape Tools](#)
 - [Text Shape](#)
- [Headers and Footers](#)

Normalize

Make several shapes the same width, height or both.

Pages Menu








The Pages menu allows you to add or remove pages in the report. Adding and removing pages in the Design Panel is easy. Under the **Page menu**, click on either **Add Page** to add a page after the currently selected page, or **Add Page Previous** to add a page before the currently selected page. To remove a page, open the page you want to remove and select **Remove Page**.























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/visible page in the current open document.
Zoom In	Increase the size of the report in view.
Zoom Out	Decrease the size of the report in view.
Zoom	Enter the percentage to zoom to.
Zoom to Fit	Have the report fill the available space.



Report Design Toolbar



The Toolbar provides a variety of functions to assist in editing while designing a report. Here is a complete list of editing functions and their descriptions in the table below.

Toolbar Item	Function
	Toggles bold of the selected text.
	Toggles <i>Italic</i> the selected text.
	Toggles <u>underline</u> of the selected text.
	Aligns the selected text left.
	Centers selected text.

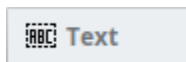
	Aligns the selected text right.
	Justifies the selected text.
	Adds a \$ to the data key.
	Adds a % to the data key.
	Forces a decimal place on an integer value in a text shape.
	Add the number of allowed decimal digits.
	Subtract the number of allowed decimal digits.
	Increase the size of the report in view.
	Decrease the size of the report in view.
	Have the report fill the available space.
	Move the selected components to the back of the Z-order.
	Move the selected components to the front of the Z-order.
	Union: alters the first selected shape to become the union of all selected shapes.
	Difference: alters the first selected shape such that the other shapes are subtracted from it.
	Align the left edge of a group of components.
	Align the right edge of a group of components.
	Align the top edge of a group of components.
	Align the bottom edge of a group of components.
	Align two or more components centerpoints' horizontally.
	Align two or more components centerpoints' vertically.
	Align three or more components in a row.
	Align three or more components in a stack.
	Combine the selected components in a group.

	
	Break apart the selected group into its child components.

Shape Tools

You may have seen some of these same shapes tools when designing your Vision windows or Perspective views. The shapes you see in the Report Design Palette are shape tools: Line, Rectangle, Ellipse, Star, Polygon, and Pencil. Shape tools allow you to create various shapes as well as edit them to create other shapes. They behave a little differently from the typical drag and drop function that you perform on a Vision window. They are not dragged from the design palette on to your report, instead you select a shape tool on the design palette to make the tool active and begin to draw your shape on to your report. You create its size, height, and shape. You'll also notice the Property Inspector will change to reflect that specific tool's properties once the tool is active. Just like components, once a shape is created, you can change its fill color, stroke color, stroke style, and more. Each shape has its own properties. To learn more about shapes, refer to the section on [Report Design](#).

Text Shape



The [Text Shape](#) is worth mentioning here because it is a fundamental component used to create text within a report. The Text Shape is used for report titles, customized page headers and footers, and of course, any additional text you want to add to your report. Simply drag a Text component on to report and begin typing. You can move the Text component around, expand it, or shrink it by using the handles when the component is selected. You can even change the font type, size, and color. The Text Shape also has a lot of properties associated with it, including Data Key Format Properties for Date and Number formats by choosing a format from the list of available templates.

Headers and Footers

Creating headers and footers is just like creating any other set of objects in your report. There is no explicit header or footer functions, or specific area on the report where they belong. Typically, headers and footers are placed at the top and bottom of the first page respectively, using the Text component or the [Built-In Keys](#).

When designing your report, the key is sizing and positioning of components within your page around your header and footers. If you have a lot of data in your data table, your report will automatically create additional pages. Each new page that the table creates will have the same header and footers.

You can use the Text component to customize your header and footers or you can use the Built-In Keys from the Key Browser. The Built-in Keys allow you to drag and drop keys to your header and footer areas such as Path, Timestamp, Date, Page number, and more. So place your header and footers where you want them on the first page, and the Report Designer will automatically place them on all additional pages.

Related Topics ...

- [Data Keys](#)

Stroke and Fill Properties for Reports

As you probably learned working with Perspective and Vision components, each component has its own unique properties. The [Report Design components](#) are no different. The Report Designer has some of the same components like tables, bar charts, graphs, and various shape tools, but their properties are different from the Perspective and Vision components because they are designed to go in reports. For example, Stroke and Fill properties control the appearance of the reporting components so you can change stroke style or border, create a dashed line around the shape, and change the width of the lines and the color. Stroke Style and Stroke properties can be particularly useful when using a Table in a report. You can make objects in a table and add an outline or set any edges of they outline to show to make your report stand out.

Stroke and Fill Properties

Stroke and Fill properties are commonly used by many reporting components. All of the components, charts, graphs, and shapes in the Report Design Palette use the same Stroke and Fill properties. These properties affect how the components look in a report. There are five basic Stroke and Fill properties:

- **Fill** - if this property is marked true, the shape will fill its space with color.
- **Fill Color** - if the Fill property is selected, the color you select will fill the shape.
- **Opacity** - defines how opaque the color is in the shape, between 0 and 1.
- **Stroke Style** - defines what type of stroke or border to use.
- **Stroke** - An expandable list of properties for the chosen stroke style.



The Fill, Fill Color, and Opacity properties are pretty easy to understand, so we are going to focus on Stroke Styles and Stroke properties in the following sections so you can take full advantage of how to use these properties to make professional looking reports.

Stroke Styles

Stroke Style defines what style of stroke or border to use. Primarily, the Stroke Style controls the type of line drawn around the component (table, chart, shape, etc) that you are using. There are four pre-configured Stroke Style templates you can choose from.

- **Hidden** - no stroke or border
- **Shape Outline** - a solid line outline
- **Border** - border or rectangle shape
- **Double** - two parallel lines

Each of these Stroke Styles come with their own set of properties. In the Design panel, select a component, and choose any one of the style templates from the Stroke Style property in the Properties tab. You can use the Stroke Style as is, or double click on any of the rows or cells to change its individual properties. For example, you might want to change the thickness of the lines or create a dashed line around the border of a component. This is easy to do: select the component, row, or cell, and change the Width under the Stroke property (thickness of the lines is specified in pixels). Then view your report with the Preview tab. You can keep repeating this step until you get the desired result.

Stroke Properties

Each Stroke shares some properties with other strokes, but have their own unique properties. The only exception is that the **Hidden** Stroke Style does not have any properties. **Note:** If you switch between stroke styles your values will be overwritten by defaults. This is not true when switching to the Hidden type.

The properties for each Stroke Style are shown below.

Shape Outline Stroke Properties

Shape Outline is a solid line outline around a component or shape. It has the following properties that you can use as is, or change.

Property	Description
Color	Color to use for the stroke.

On this page ...



- [Stroke and Fill Properties](#)
- [Stroke Styles](#)
- [Stroke Properties](#)
 - [Shape Outline Stroke Properties](#)
 - [Border Style Stroke Properties](#)
 - [Double Style Stroke Properties](#)

Dash Pattern	A "Dash Pattern" string to specify the number of pixels on and off. For example "5,10"
Width	Width of the shape in pixels.

Stroke Style - Shape Outline Example

This example shows a report with a Shape Outline around the Table component at the bottom of the image. Select the whole table to set stroke and fill for it. The property values used to create the outline around the data are shown in the properties list in the table below.

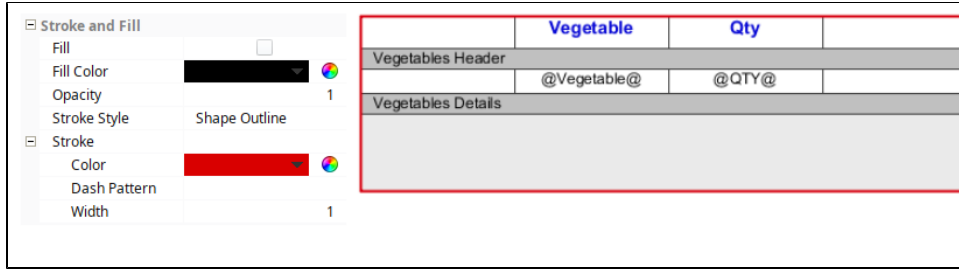
Vegetable Report

● Squash	● Zucchini	● Carrots
● Potatoes	● Beans	● Peppers

Vegetable	Qty
Squash	10
Zucchini	18
Carrots	25
Potatoes	8
Beans	33
Peppers	9

Table Stroke and Fill Properties



Border Style Stroke Properties

The **Border** Stroke Style is similar to the Shape Outline, but it gives you the option to disable the any of the lines around the border; bottom; left; right; or top.

Property	Description
Color	Color to use for the stroke.
Dash Pattern	A "Dash Pattern" string to specify the number of pixels on and off. For example "5,10"
Border Bottom	If true, show the border on the bottom side of the shape.
Border Left	If true, show the border on the left side of the shape.
Border Right	If true, show the border on the right side of the shape.
Border Top	If true, show the border on the top side of the shape.
Width	Width of the stroke in pixels.

Stroke Style - Border Example

This example used the Border Stroke Style for the Equipment Summary cell at the bottom of a Table to highlight the total minutes of downtime for the included equipment (Total: @total.Downtime@Minutes). Three images appear at the top because they are in the Equipment Header ([unstructured](#)) row. The property values are shown for the Table, Row, and Cell in the property lists below.

Equipment



Labeler

Out of labels	Duration (Minutes)	Date
	50	Jan 20, 2017
	21	Feb 12, 2017
	71 Minutes	
Misalignment	Duration (Minutes)	Date
	33	Jan 20, 2017
	33 Minutes	

Filler

Overflow	Duration (Minutes)	Date
	22	Feb 20, 2017
	30	Feb 28, 2017
	52 Minutes	
Maintenance	Duration (Minutes)	Date
	19	Feb 25, 2017
	19 Minutes	

Conveyor Line

Backup	Duration (Minutes)	Date
	45	Jan 21, 2017
	45 Minutes	
Maintenance	Duration (Minutes)	Date
	47	Feb 13, 2017
	47 Minutes	

Total: 267 Minutes

Equipment Table in the Designer

Equipment			
Equipment Header			
@Equipment@			
Equipment Details			
	@Cause@	Duration (Minutes)	Date
Cause Details			
		@Downtime@	@T_stamp@
Data Details			
		@total.Downtime@ Minutes	
Data Summary			
		Total: @total.Downtime@ Minutes	
Equipment Summary			



Table Property Values

Table Stroke and Fill Properties	Property	Value
<div style="border: 1px solid black; padding: 5px;"> <p>Stroke and Fill</p> <p>Fill <input type="checkbox"/></p> <p>Fill Color </p> <p>Opacity 1</p> <p>Stroke Style Border</p> <p>Stroke</p> <p>Color </p> <p>Dash Pattern</p> <p>Border Bottom <input checked="" type="checkbox"/></p> <p>Border Left <input checked="" type="checkbox"/></p> <p>Border Right <input checked="" type="checkbox"/></p> <p>Border Top <input checked="" type="checkbox"/></p> <p>Width 1</p> </div>	<p>Stroke Style</p> <p>Stroke</p> <ul style="list-style-type: none"> • Border Bottom • Border Left • Border Right • Border Top 	<p>Border</p> <p>All Enabled</p>

Data Summary Cell

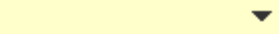



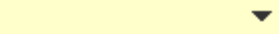



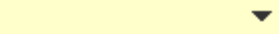



Shows the Total Downtime Minutes for each downtime cause for each piece of equipment.

Cell Text Properties for Data Summary	Property	Value
	Text Color	Red, Bold

Text Properties	
Text	@total.Downtime@Minutes
Text Color	 
Character Spacing	0
Coalesce Newlines	<input type="checkbox"/>
Font	Arial Bold 12.0 (Arial)
Horizontal Alignment	Left
Line Spacing	1
Margin	1; 2; 0; 2
Overflow Behavior	Grow row
Underlined	<input type="checkbox"/>
Vertical Alignment	Top

Equipment Summary Cell

Shows the Total Downtime Minutes for all causes and for all equipment.

Cell Stroke and Fill Properties for Equipment Summary		Property	Value																									
<table border="1"> <thead> <tr> <th colspan="2">Stroke and Fill</th> </tr> </thead> <tbody> <tr> <td>Fill</td> <td><input type="checkbox"/></td> </tr> <tr> <td>Fill Color</td> <td> </td> </tr> <tr> <td>Opacity</td> <td>1</td> </tr> <tr> <td>Stroke Style</td> <td>Border</td> </tr> <tr> <td colspan="2">Stroke</td> </tr> <tr> <td>Color</td> <td> </td> </tr> <tr> <td>Dash Pattern</td> <td></td> </tr> <tr> <td>Border Bottom</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>Border Left</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>Border Right</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>Border Top</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>Width</td> <td>2</td> </tr> </tbody> </table>	Stroke and Fill		Fill	<input type="checkbox"/>	Fill Color	 	Opacity	1	Stroke Style	Border	Stroke		Color	 	Dash Pattern		Border Bottom	<input checked="" type="checkbox"/>	Border Left	<input checked="" type="checkbox"/>	Border Right	<input checked="" type="checkbox"/>	Border Top	<input checked="" type="checkbox"/>	Width	2	Stroke Style	Border
	Stroke and Fill																											
	Fill	<input type="checkbox"/>																										
	Fill Color	 																										
	Opacity	1																										
	Stroke Style	Border																										
	Stroke																											
	Color	 																										
	Dash Pattern																											
	Border Bottom	<input checked="" type="checkbox"/>																										
Border Left	<input checked="" type="checkbox"/>																											
Border Right	<input checked="" type="checkbox"/>																											
Border Top	<input checked="" type="checkbox"/>																											
Width	2																											
	Fill	Enabled																										
	Fill Color	Yellow																										
	Width	2 pixels																										

Double Style Stroke Properties

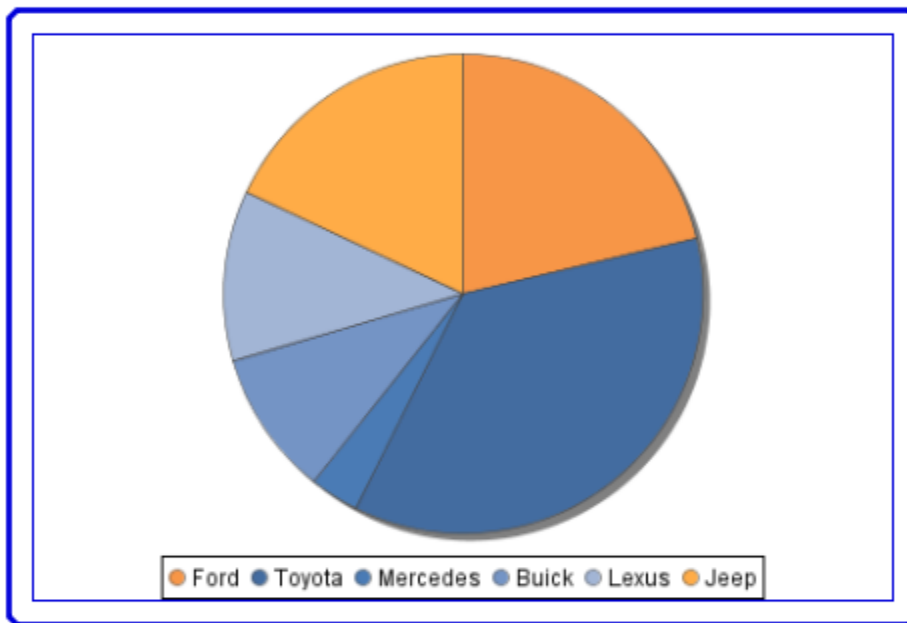
By default, the **Double** Stroke Style consists of two parallel lines around the component. You can choose to disable any of the lines at the bottom, left, right, or top. You can also change the width of both lines and the separation between these lines, if desired.

Property	Description
Color	Color to use for the stroke.
Inner Width	Width of the inner border in pixels.
Outer Width	Width of the outer border in pixels.
Position	Position of borders relative to the bounds of the shape. <ul style="list-style-type: none"> Outer on path - places the outer border on the edge of the component and the inner border inside the component. Inner on path - places the inner border on the edge of the component and the outer border outside the component. Centered about path - centers the whole border (based on inner+outer+gap widths) on the shape edge. Gap on path - centers the gap on the shape edge.
Separation	Separation between inner and outer borders in pixels.

Stroke Style - Double Example

Here is an example of a Pie Chart with a Double border used in a report. The property values for the Border Stroke Style are listed in the properties list table below.

Used Car Inventory Report August 8, 2017



Ford	13
Toyota	22
Mercedes	2
Buick	6
Lexus	7
Jeep	11



Controller
Aug 8, 2017

Car Inventory

1 of 1

Pie Chart Stroke and Fill Properties

Property	Value
Stroke Style	Double
Stroke Color	Blue

Opacity		1
Stroke Style	Double	
Stroke		
Color		
Inner Width		1
Outer Width		3
Position	Inner on path	
Separation		10

Inner Width	1
Outer Width	3
Position	Inner on Path
Separation	10

Related Topics ...

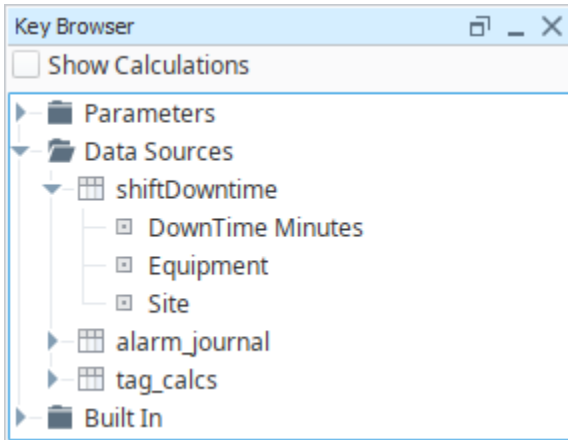
- [Report Design](#)
- [Report Design Tools](#)

Data Keys

Data Keys

In the Reporting Module, we use **Data Keys** to pull values from data sources and show them on the report. In simple terms, Data Keys are placeholders for your data. The simplest reference to data is a simple Data Key. At report generation time, these keys resolve to the values (or sets of values) provided by the data source. Additionally, Data Keys may be used as expressions, which are referred to Keychain Expressions.

As you add Parameters and Data Sources to the [Data section](#) of your report, they will appear in the Key Browser's **Parameters** and **Datasources** folders.



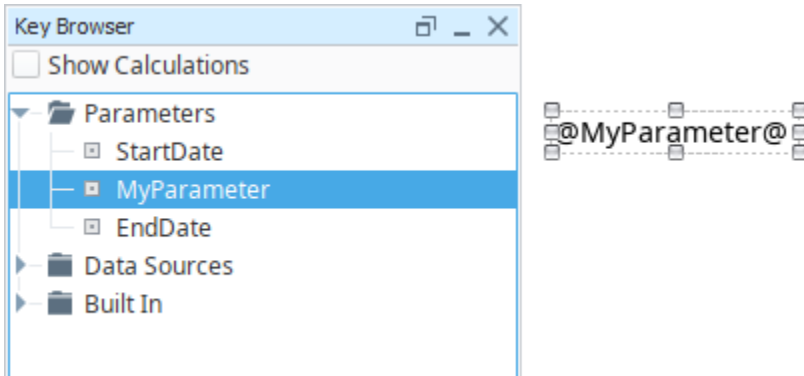
On this page ...

- [Data Keys](#)
 - [Data Keys on the Report](#)
 - [Escaping the @ Character](#)
- [Built-In Keys](#)
 - [Built-In Data Key Description](#)
- [Show Calculations Property](#)
- [Dynamic Data Keys](#)
 - [Configuring a Dynamic Data Key](#)
- [Data Key Usage](#)
 - [Data Keys as Paths](#)
 - [Array Index of Data](#)
 - [Colors in Expressions](#)

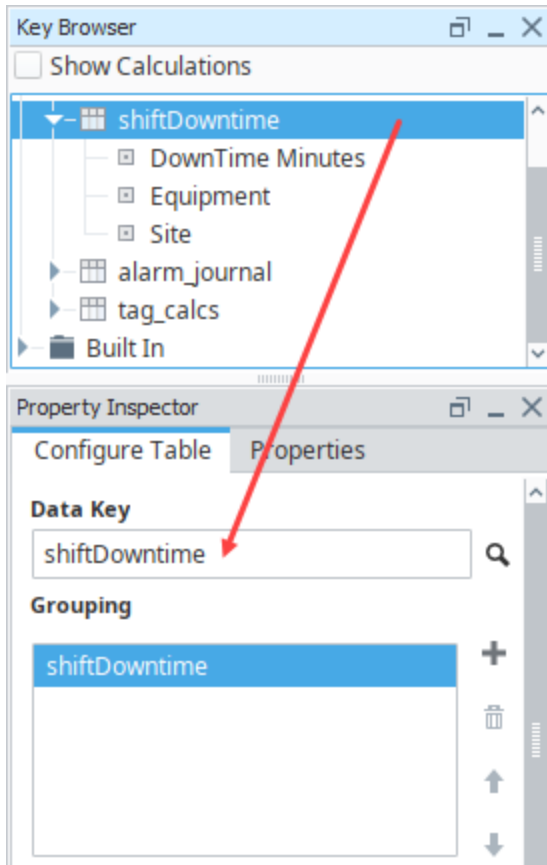
Data Keys on the Report

Data Keys are enclosed in the "@" character when utilized by components in the report. They may be typed manually, or dragged directly from the Key Browser.

Keys that contain a single value will create a [Text Shape](#) when dragged onto the report.



Keys that represent datasets will create a [Table](#) component, and configure the Data Key property to use the key.



Escaping the @ Character

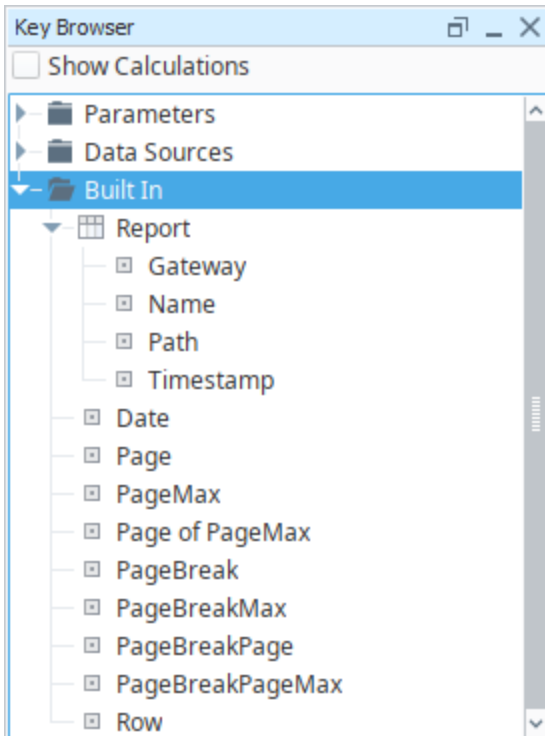
In some cases, your report may need to contain multiple strings containing the "@" character. Since these characters are used to denote keys, multiple instances of these characters may lead to undesirable behavior. You can escape this key lookup with "@@". For example, if a Text Shape needed to contain multiple email addresses, they could be typed in the following way:

```
user1@@company.com  
user2@@company.com
```

When the report generates, the double "@@" characters will be replaced with a single "@" character.

Built-In Keys

Built-In Keys provide a lot of useful information on your report at a glance. The Built-In keys are found in the Key Browser. Expand the Built-In folder and you'll see all the default keys, including a Report folder. The keys in the Report folder are specifically related to the report: Gateway name that the report is located in, report name, folder path from the Project Browser to the report, and the Timestamp of the Gateway. The other Data Keys are related to information you may want to add to a report like the date you are viewing or printing the report, page number, and number of total pages.



Built-In Data Key Description

The tables below list the Built-In Report Data Keys and Built-in Data Keys along with a brief description of each key.

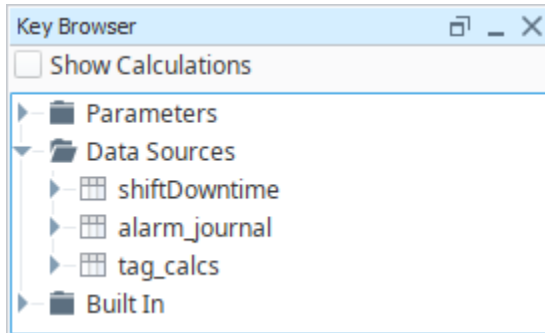
Key	Description										
Report	<p>This key has multiple sub-keys that provide meta-data about the report.</p> <table border="1"> <thead> <tr> <th>Key</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Gateway</td> <td>Name of the Report's Ignition Gateway</td> </tr> <tr> <td>Name</td> <td>Name of the Report</td> </tr> <tr> <td>Path</td> <td>Path to the Report in the Project Browser</td> </tr> <tr> <td>Timestamp</td> <td>The Gateway's current timestamp</td> </tr> </tbody> </table>	Key	Description	Gateway	Name of the Report's Ignition Gateway	Name	Name of the Report	Path	Path to the Report in the Project Browser	Timestamp	The Gateway's current timestamp
Key	Description										
Gateway	Name of the Report's Ignition Gateway										
Name	Name of the Report										
Path	Path to the Report in the Project Browser										
Timestamp	The Gateway's current timestamp										
Date	The current date/time										
Page	The current page										
PageMax	The total number of pages in the generated report										
Page of PageMax	Shows current page number and the total number of pages in the 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										
Row	Shows the current row number. Must be used in a table										

Show Calculations Property

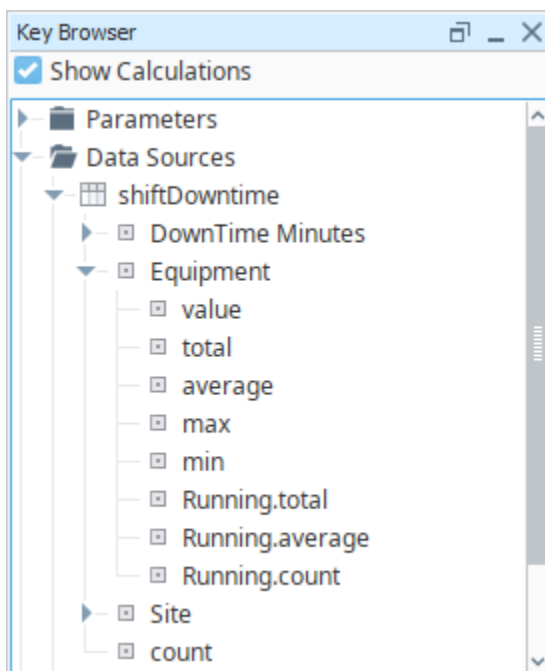
In the Key Browser, the **Show Calculations** property will add several aggregates to each key. These allow your reports to easily display things like the total of a key. These calculations are typically used in the [summary row](#) on the Table component.

Once Show Calculations is enabled, the Key Browser will refresh, and each key will be expandable. Expanding a key will show the available calculations. For example:

When **Show Calculations** is disabled, the key browser only lists the shiftDowntime, alarm_journal, and tag_calcs Data Source keys.



When **Show Calculations** is enabled, the shiftDowntime, alarm_journal, and tag_calcs Data Sources are expandable and show available calculations like DownTime Minutes, Equipment, and Site.




Calculation Keys work like any other key: they may be dragged onto the report, and utilized in [Keychain Expressions](#).

Dynamic Data Keys

Normally, Data Keys may only be used to display the value of a key, such as the Text property on a [Text Shape](#) component. However, they can not be used in the same manner to modify other properties on a report component. Instead, you can utilize Dynamic Data Keys.

Dynamic Data Keys allow you to use the value of a Data Key on a non-string property. With Dynamic Data Keys, you can modify properties on report components, such as the background color or width, based on the value of a key. This is very similar to the binding system used by components in the [Vision Module](#).

INDUCTIVE
UNIVERSITY

Use Dynamic Data Key

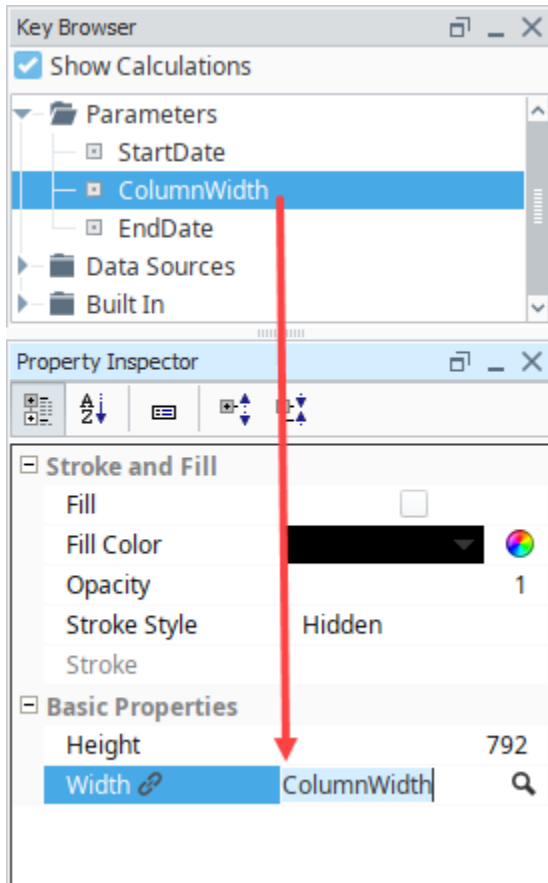
[Watch the Video](#)

Configuring a Dynamic Data Key

There are two ways to configure a dynamic data key. Note that the syntax of keys differs in Dynamic Data Keys: the "@" are omitted, as demonstrated below.

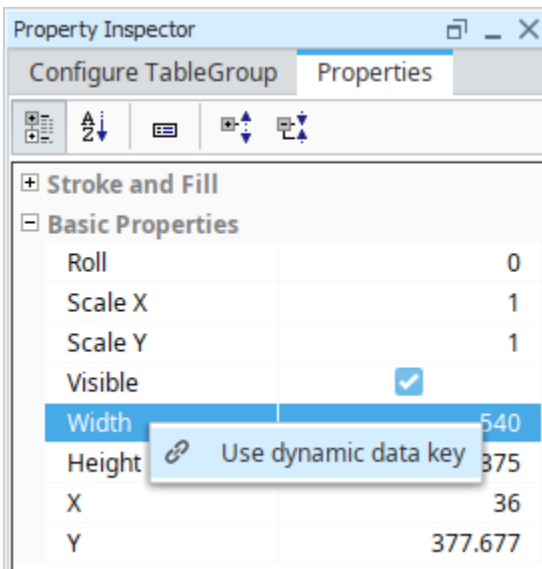
Drag-and-Drop

The easiest approach is by simply dragging a data key from the **Key Browser** directly to a property on a report component.



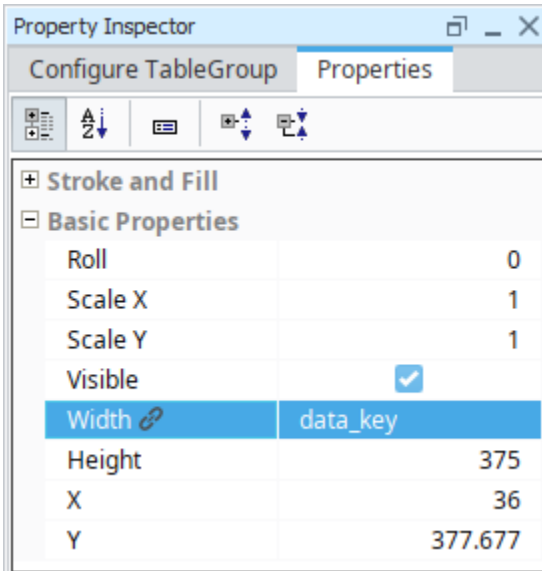
Right-Click

1. With a report component selected, look in the **Property Inspector**, and left click on the name of a property you wish to place the dynamic data key on.
2. Once selected, you can right-click on the property name and a menu will appear.
3. Click on the **Use dynamic data key** menu item.



4. This will place a dynamic data key on the property. An icon of a link (🔗) will appear next to the property name, and a default key will be applied to the property.

- Next we will want to override the default value with one of the keys from the Key Browser. Simply left click on the value field and a magnifying glass (🔍) icon will appear.



- Click on the (🔍) icon. From here a popup of available keys will appear. Select the key you wish to use, and then click OK.

Data Key Usage

Data Keys as Paths

Data Keys are *relative*, and use 'dot notation' to reference children. Meaning, if we have a nested data structure, we can use Data Key paths (also known as *Keychains*) to reference the nested data. In the key browser image below, we have a nested data source called `Downtime`. `Downtime` contains a number of columns, and then contains a reference to additional data called `runInfo`. If we wanted to access the highlighted `operator` data, we could use the keychain dot notation in the Designer - `@Downtime.runInfo.operator@`. Nested data sources are outside the scope of this page, but you can learn about Data source nesting in the [Nested Queries](#) section.

Array Index of Data

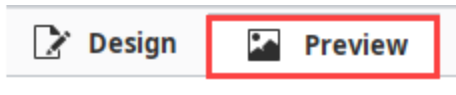
You can reference an individual object in a list using standard array indexing syntax (brackets) like this: `@dataSource[0].columnName@`, where "dataSource" is a data source that contains a child data key named `columnName`. Assuming a data source with the values listed below, we can retrieve the value of "Second Row" by specifying index 1 and the column `stringValue`: `@static_data[1].stringValue@`

static_data Example

```
indexColumn, stringValue
0, "First Row"
1, "Second Row"
2, "Third Row"
4, "Fourth Row"
```



```
@static_data[1].stringValue@
```



Second Row

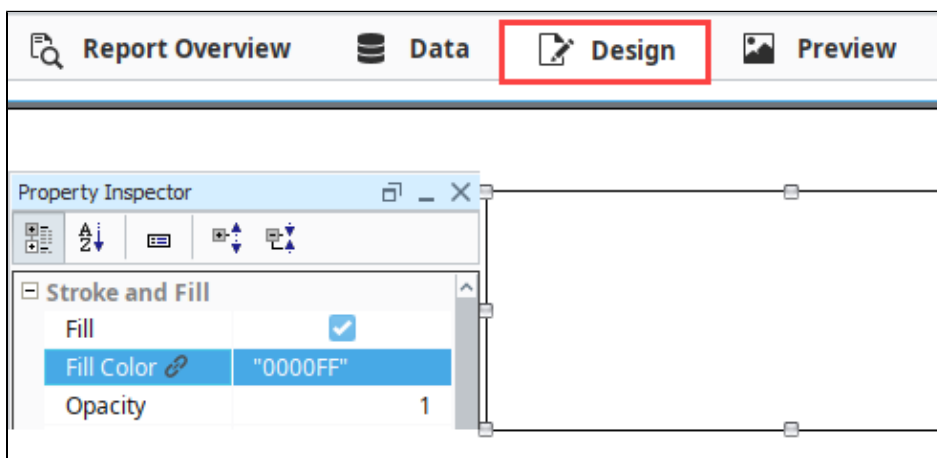
Colors in Expressions

Colors may be references in Keychain Expressions in several ways.

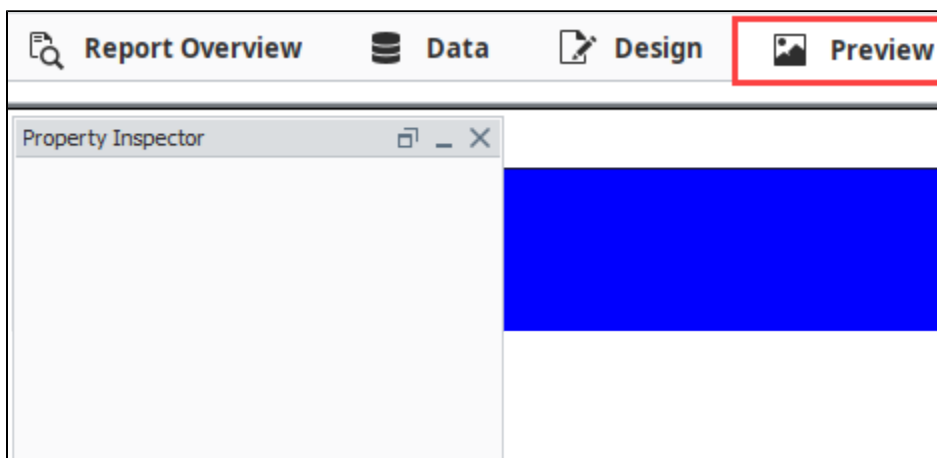
Colors in Hexadecimal

First, hexadecimal case-insensitive color codes may be used. The code must be wrapped in quotation marks to be evaluated correctly. Note that the color change will only appear when the report is executed. The easiest way to test the expression is to switch to the Preview Panel.

Below we see a Blue hexadecimal code of **"0000FF"** is used on the Fill Color of a Rectangle. The Fill Color on the Rectangle was originally set to White. Because the expression will not evaluate in the Design Panel, the Rectangle will appear as a White color.

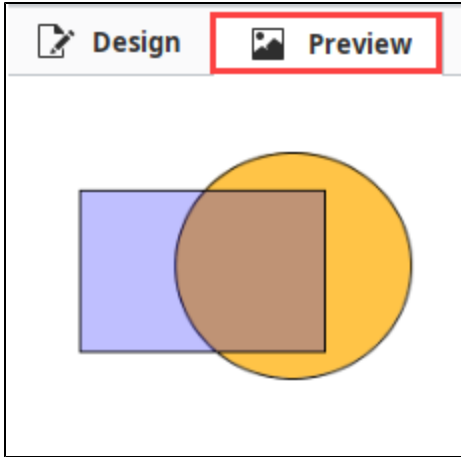


However, switching over to the Preview Panel will generate the report, and evaluate the expression. This in turn returns a Blue Fill Color.



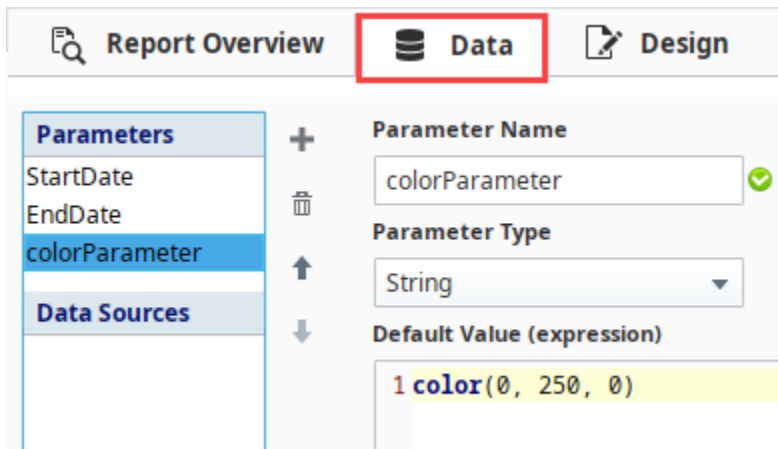
The 7th and 8th digits may be added to specify an alpha channel, or the opacity of the color: **00** is fully transparent, while **FF** is fully opaque.

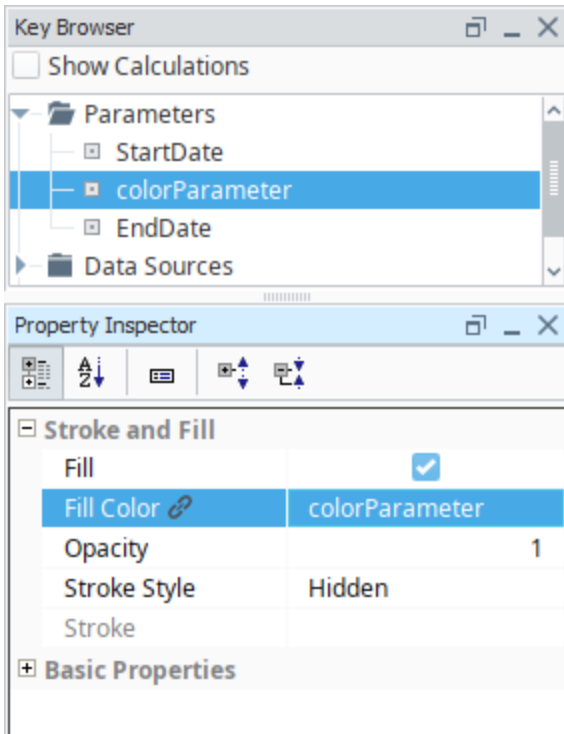
Below, we see a similar rectangle overlapping an ellipse, but with a code of **"0000FF40"**. This represents ~25% opacity, so objects behind the rectangle will be visible, and the fill color will only be slightly opaque.



Parameters as Colors

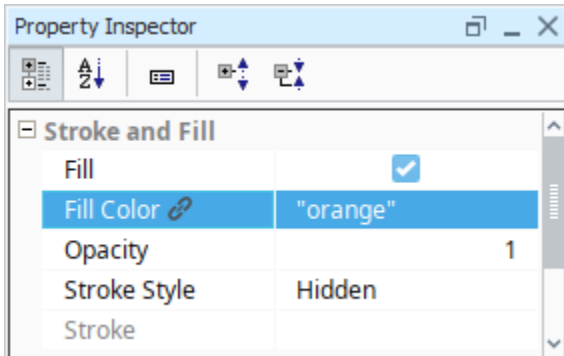
You may also leverage Report Parameters to specify colors. This typically involves creating a parameter with a string datatype, and using the `color` expression function. Once created, you can simply create a dynamic data key reference on the property by dragging the parameter from the **Key Browser** onto the property in the **Property Inspector**. This way, you can have several components use the same color, and modify the color in a single location.





Strings as Colors

Additionally, case-insensitive string color names may be used to return a color. Again, the value must be wrapped in quotation marks, "orange".



The following string values may be used:

String Value Color Reference									
Value	Example	Value	Example	Value	Example	Value	Example	Value	Example
"beige"		"gold"		"lavender"		"pink"		"tan"	
"black"		"goldenRod"		"lightGray"		"plum"		"teal"	
"blue"		"gray"		"lime"		"powderBlue"		"violet"	
"brown"		"green"		"magenta"		"purple"		"white"	
"crimson"		"hotPink"		"maroon"		"salmon"		"yellow"	

"cyan"		"indigo"		"navy"		"silver"		"clear"	Zero opacity. Similar to disabling the Fill property.
"darkGray"		"ivory"		"olive"		"skyBlue"			
"fuchsia"		"khaki"		"orange"		"red"			

In This Section ...

Keychain Expressions

It's possible to perform calculations on data keys. This section documents the various operators and functions that are available.

Keychain Expressions

Keychains have their own expression language that is largely similar in syntax to Java. To separate this language from others in Ignition, we refer to it as a Keychain Expression.

Keychain Expressions are configured by simply utilizing any operators or functions within the "@" characters. Assuming a key named **"myKey"** with a value of 10, we can multiply its value by 10 with the following expression:

```
//Expression
@myKey * 10@
```

```
//Output
100
```

Again, note that the "*" operator and multiplier are enclosed in the "@" characters. Of course, we can also use a different key as the multiplier. Assuming we have another key named **"myMultiplier"** that has a value of 5:

```
//Expression
@myKey * myMultiplier@
```

```
//Output
50
```

Any characters outside of the "@" characters are not part of the expression, so you can easily add prefixes and suffixes with static text.

```
//Expression
Total: @myKey * myMultiplier@ gal
```

```
//Output
Total: 50 gal
```

Additionally, you can utilize separate Keychain Expressions in the same TextShape. Note that **"myUnits"** (which has a string value of **"gal"**) is enclosed in a separate set of "@" characters, since it is a separate expression:

```
//Expression
Total: @myKey * myMultiplier@ @myUnits@
```

```
//Output
Total: 50 gal
```

Dynamic Data Key Expressions

Keychain Expressions may also be used with Dynamic Data Keys. The syntax and operators work exactly the same, except there is no need to type the "@" characters. Because of this, the entire field is treated as a single keychain expression.

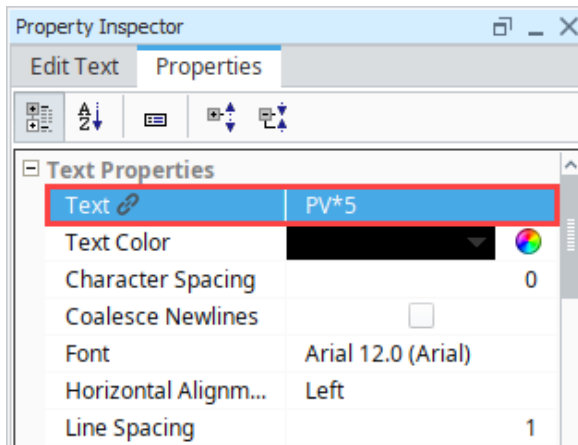
On this page ...

- [Keychain Expressions](#)
 - [Dynamic Data Key Expressions](#)
- [Conditional Keychain Example](#)
- [Operators and Functions](#)
 - [Operators](#)
 - [Math Functions](#)
 - [String Functions](#)



Key Calculations

[Watch the Video](#)



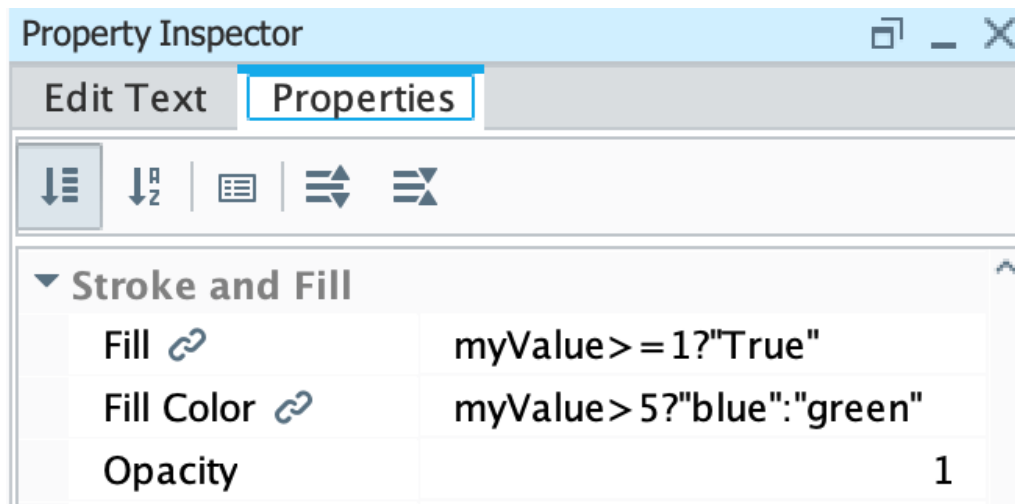
Conditional Keychain Example

The following example demonstrates an if-statement using a Dynamic Data Key. This allows us to highlight different values or ranges contextually, making important values stand out.

Assuming a key named "myValue" has been created, and contains a numerical value, we can use the following syntax:

```
#If the value of the "myValue" key is greater than 5, a blue color will be returned. Otherwise, a green
color will be used.
@myValue>5?"blue":"green"@
```

In the image below, the Fill property is also using a dynamic data key. Therefore, the Fill Color will be disabled if myValue is less than 1, Blue if myValue is between 1 and 4, and Green if greater or equal to 5. Note that in the Property inspector, the @ symbols are not needed.



To add more color-value pairs, we simply add more if statements to the end of the expression with a colon:

```
#If the value of "myValue" will determine one of multiple colors:
#Greater than 10 will return Red
#greater than 5 (but not greater than 10) will return Blue
#anything else will return Green.
@myValue>10?"red":myValue>5?"blue":"green"@
```

Operators and Functions

Operators

The following operators may be used in a Keychain expression.

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.

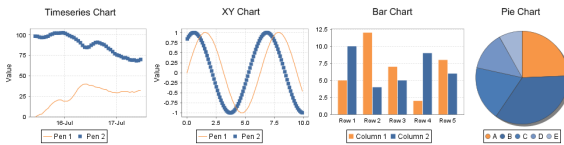
join(List aList, String aKeyChain, String aDelimiter)	Used to display an individual attribute of individual objects as a single String. Suppose you have a list of movies and want to show their titles in a comma separated list: @join(getMovies, "getTitle", ", ")@
substring(Object aString, int start, int 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)@

Report Charts

Report Charts

Report charts allow you to display your data in a graphical way, just like the charts in the rest of Ignition. Charts can be driven by any [data source](#) in a report, or even [embedded into table rows](#) using nested data sets. There are four types of charts so you can show the data any way you like.

- **Timeseries Chart:** a simple chart that plots values against a timestamp X axis. Great for showing historical trending.
- **XY Chart:** similar to the Timeseries chart, this chart plots lines on X-Y axes, but is configured to show numeric categories on the X axis.
- **Bar Chart:** a bar chart that uses text categories on the X axis. Can be configured as a Pareto chart.
- **Pie Chart:** a basic pie chart that uses text categories. Percentages are automatically calculated.



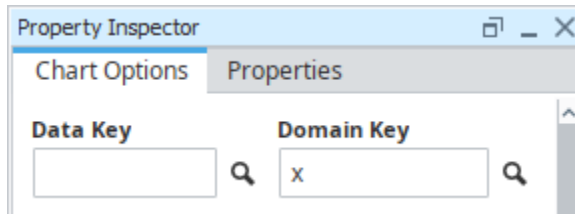
On this page ...

- [Report Charts](#)
- [The Data Key Property](#)
- [Timeseries Chart](#)
 - [Usage](#)
- [XY Chart](#)
 - [Usage](#)
- [Bar Chart](#)
 - [Usage](#)
- [Pie Chart](#)
 - [Usage](#)
- [Chart Scripting](#)

The Data Key Property

Similar to Tables in reporting, all charts must **first** be assigned a Data Key. This assignment configures the chart to look at specific keys in a data source, and prevents any name collisions with other data keys: if multiple data sources in your report return a column named "id", the chart wouldn't know which column you were referring to without this initial assignment. Like all keys in the Report manual, make sure your Domain and Range keys are just the column name. If your keys look like "query.column" then you will not see data in your chart.

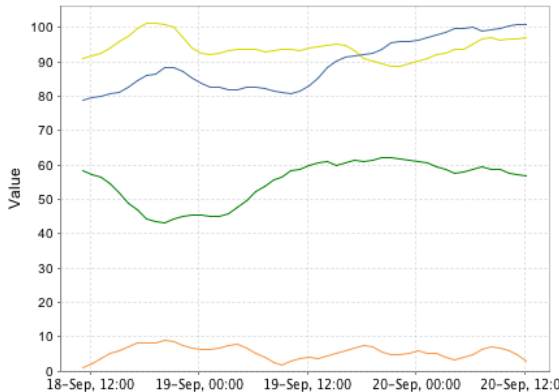
Assignment can be achieved by drag-and-dropping a key into the Data Key property, or by clicking the Key Search button.



Once assigned, you can the start adding pens or bars to your chart.

Timeseries Chart

The [Timeseries Chart](#) is a type of XY chart whose domain or X-Axis represents time series data and range can be one or more pens. The Timeseries Chart is a great way to display data visually from Tag History, or similar time related data sources.



Usage

Timeseries Chart

[Watch the Video](#)

To use the chart, drag the component from the Report Palette to your report. Type or drag a data key from the Key Browser into the **Data Key** field of the Chart Options tab in the Property Inspector. Select a time series domain (for example a `t_series` column of your query). The Y axis (value) can be modified in the Chart Options tab, but the X axis (time range) is based on the data in the 'Data Key' that powers the chart dataset.



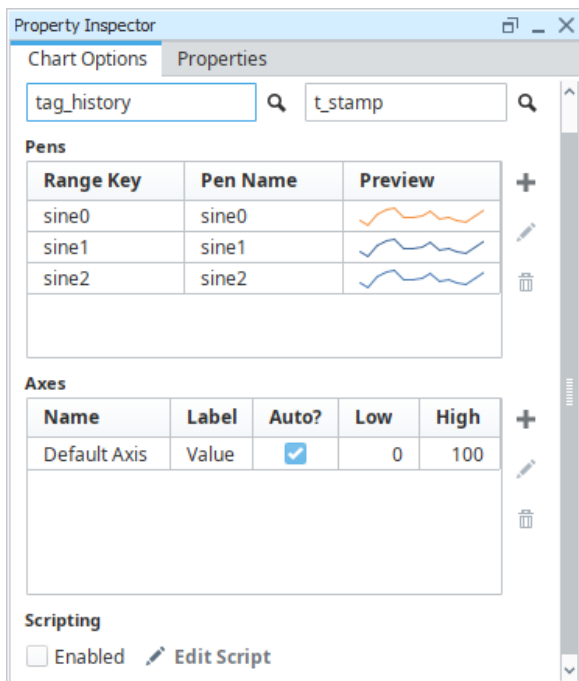
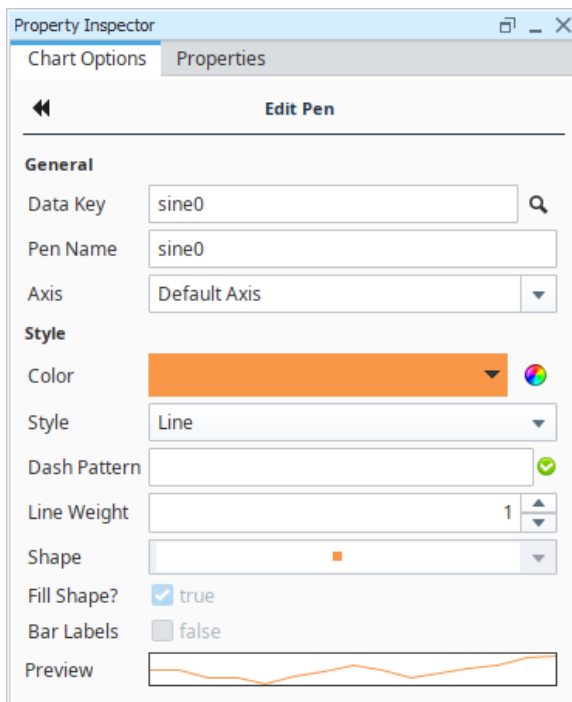
To add pens to your query, simply click the **+** button on the Chart Options tab next to the Pens table in the Property Inspector. You can either double click a pen, or select a pen and click the  button on the right side, to navigate to the Pen Configuration area. Click the  to return to the Chart Options tab.


Chart Options area

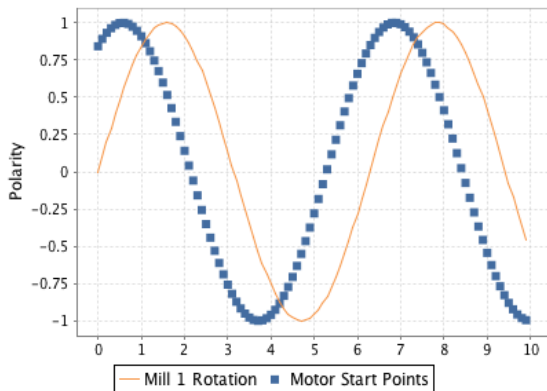


Pen Configuration area




XY Chart

The  **XY Chart** generates an X vs Y plot of your data. XY Charts can have multiple pens and axes per data source, and each is easily configurable in the Chart Options tab for the component.



Usage




To use the chart, drag the component from the Report Palette to your report. Type or drag a data key from the Key Browser into the **Data Key** field of the Chart Options tab. Select Domain Key to use for the X axis.




INDUCTIVE UNIVERSITY


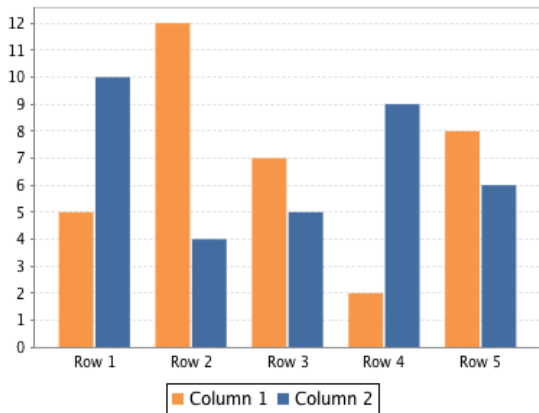
XY Chart

[Watch the Video](#)

To add pens to your query, simply click the **Add**  icon on the Chart Options tab next to the Pens table in the Property Inspector. You can either double click, or select a pen and click the **Edit**  icon on the right, to navigate to the Pen Configuration area. Click the **Back to chart options**  icon to return to the Chart Options tab. The setup and Configuration of this chart and its pens works similarly to the Timeseries Chart, the difference being that instead of using a date datatype for the domain, a different datatype is needed.

Bar Chart

The **Bar Chart**  component can be used to add bar charts to a report. It uses text categories on the X axis, and can also be configured as a Pareto chart.



Bar Chart

[Watch the Video](#)

Usage

To create a Bar Chart, simply drag the component from the Report Palette and drop it onto your report. Bar Charts are quite easy to use and have a large number of customization options. Configuring a Bar Chart requires a data source whose first column generally contains the name or identifier the bar represents, and the following one or more columns represent some numerical value to be plotted. A Bar Chart example can be found in the [Report Workflow Tutorial](#).

Property Inspector

Configure | **Properties**

Data Key

Extract Order Row Column


Options 3D Bars Pareto

Render Style Bars Stacked Layered

Segment Colors

Scripting Enabled

Pie Chart

The [Pie Chart](#)  **Pie Chart** adds a Pie Chart based on a simple two column data structure.





 Bananas  Grapes  Apples  Pineapples  Other

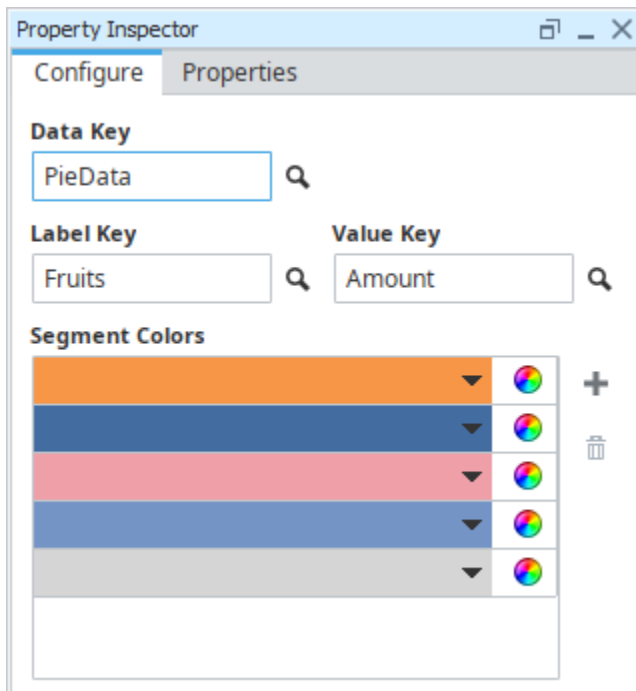


Pie Chart

[Watch the Video](#)

Usage

To create a new Pie Chart, drag the pie chart palette item from the Report Design Palette onto your Report Page or Parent Shape. A pie chart has a simple configuration consisting of label values (generally Strings) and numeric quantity values. Color segments can be added or removed by clicking the **Add**  or **Remove**  icons on the Configure tab.

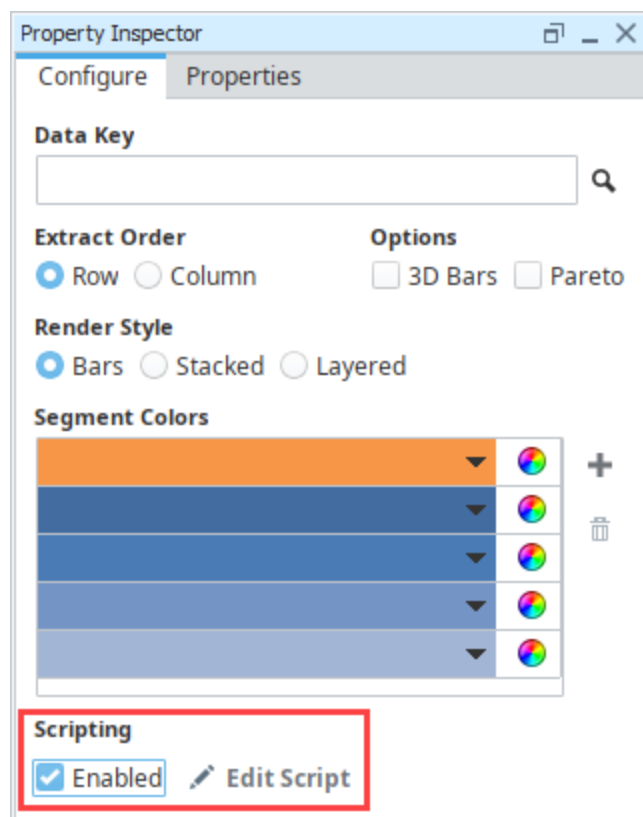


The Pie Chart in the images above was created using a simple data source which had just two columns, the first which represented our Label, and the second being a column of values.

```
Fruits, Amount
"Bananas", 52
"Grapes", 154
"Apples", 58
"Pineapples", 25
"Other", 265
```


Chart Scripting

The look and feel of the Timeseries, XY, and Bar charts may be modified through scripting. Scripting may be accessed on these charts by selecting them, and then clicking the **Edit Script** icon in the Property Inspector. Note that the **Enabled** property must be checked for the script to become active.



Clicking on **Edit Script** will provide access to `configureChart`, which allows you to make modifications to the chart right before the report is rendered. The charts are simply JFreeCharts, so [reading through the JFreeChart API](#) would be useful here.

Related Topics ...

- [Report Tables](#)
- [Tutorial: The Report Workflow](#)

In This Section ...

Getting Started with Report Charts

Report charts have a ton of functions and features that it's hard to know where to begin. You'll be amazed how simple a standard chart is to create. To start with, there are several chart types to select from, but the chart type you choose depends on the type of data you have and how that data might best be graphically displayed. Next, charts are driven by specific data keys in a data source. This page provides a quick start to getting you well on your way to creating report charts that are easy to configure and look stunning in your reports. Let's get started!

On this page ...

- [Creating a Report Chart](#)
 - [Creating a Data Source](#)
 - [Adding a Chart](#)
 - [Adding an Axis](#)

Creating a Report Chart

One of the most common charts used for historical trending is a [Timeseries Chart](#). The Timeseries Chart plots values against a timestamp X axis.

Adding a chart to a report involves a couple of steps. In order, they are:

1. The chart needs data, so we must create a data source.
2. Once a data source exists, we can create the chart and assign the data source.
3. Apply any additional chart configuration, such as adding an additional axis.

Now, let's create a Timeseries Chart using Tag History data.

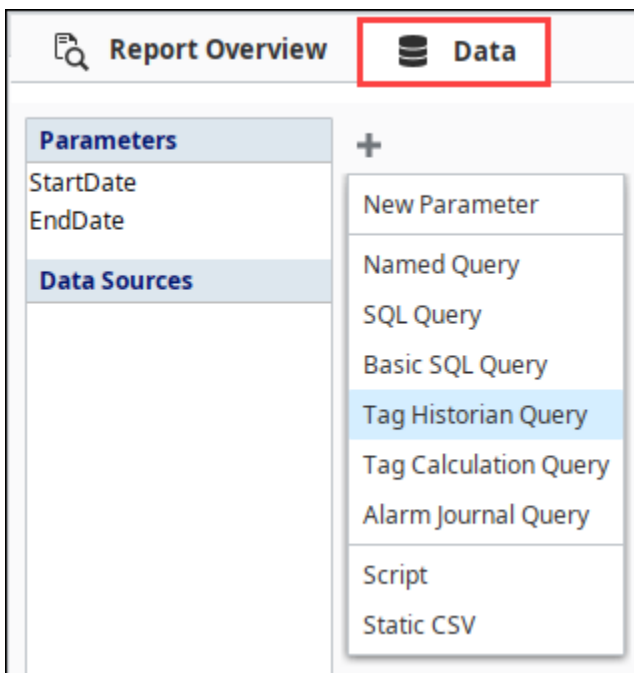
Before you start this example...

Ignition comes with one internal Tag Provider so you can get started right away, but first you need to connect to a device. The [Programmable Device Simulator](#) is a great starting point if your Gateway is not currently connected to a PLC. This example assumes you named your device 'generic' are using the Generic Program device tags, but any tags could be used instead.

You also need a [database connection](#) that the historian can use to store Historical Tag data. This example assumes [Tag History has been enabled](#) on the **Sine** tags.

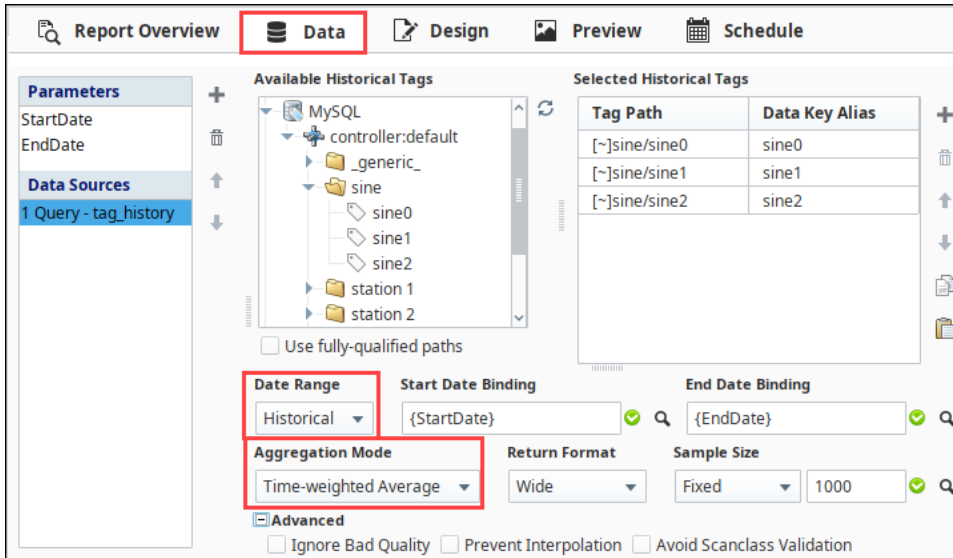
Creating a Data Source

1. In the **Data** tab, under Parameters and Data Sources, click the Add  icon to add a data source. Select **Tag Historian Query**.



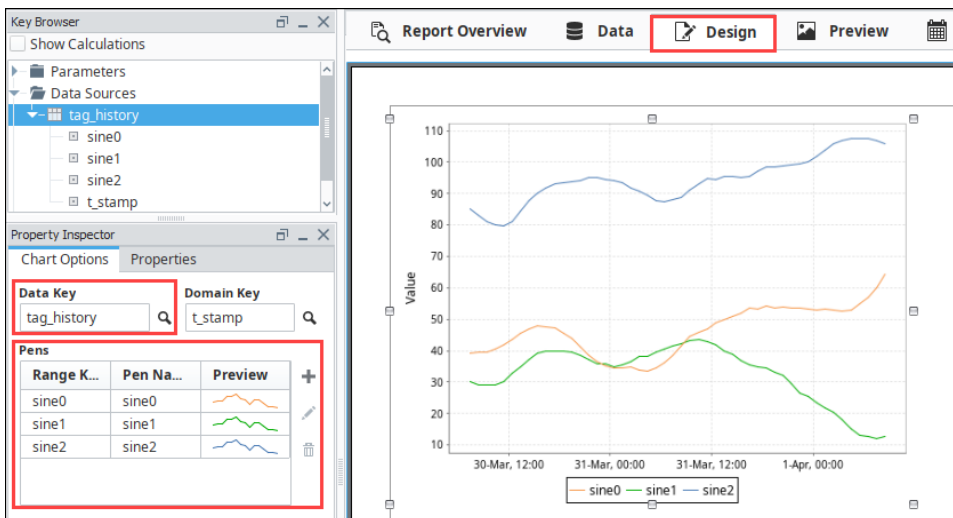
2. Under the **Available Historical Tags** column, open the folder structure until you locate your Tags. As mentioned above, we are using the Programmable Device Simulator and loaded the Generic program. so we will select the 'generic' folder in this example. Here you'll find a bunch of Tags you can use immediately. Expand the **Sine** folder.

3. Select a few of the Tags (i.e. sine0, sine1, sine2) and drag them to the **Selected Historical Tags** column under **Tag Path**. Take a look at the properties down at the bottom. Set the **Date Range** property to **Historical**, and the **Aggregation Mode** to **Time-weighted Average**.

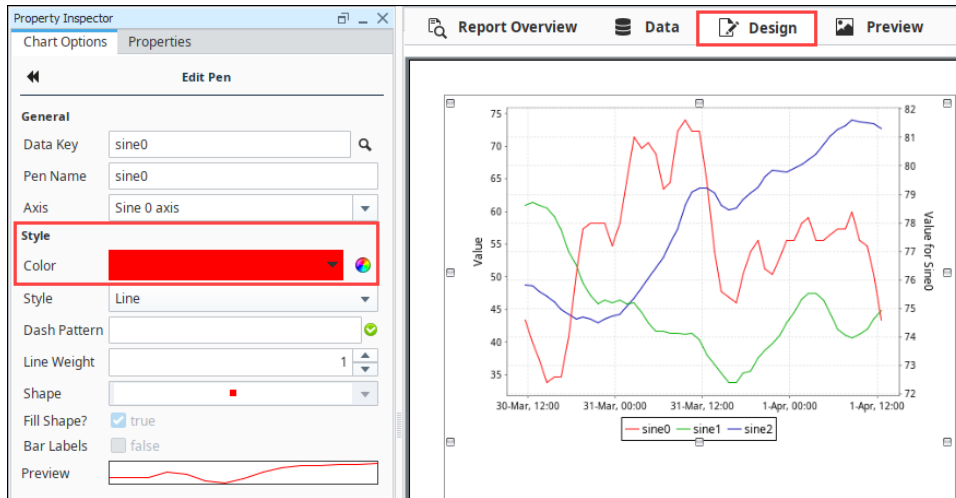


Adding a Chart

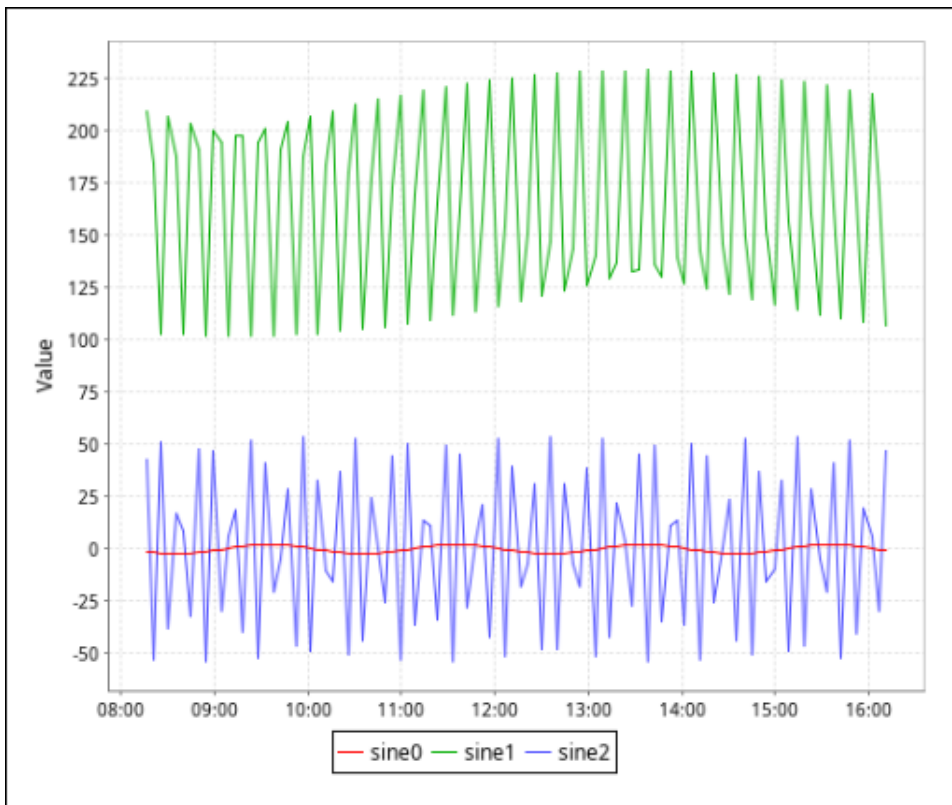
1. Now switch over to the **Design** panel. Drag a **Timeseries Chart** from the Report Design Palette into the report and expand it.
2. From the **Key Browser**, expand the **Datasources** folder, and drag your datasource (i.e., tag_history) down to the **Data Key** under the **Chart Options** tab of the Property Inspector.
3. Select **Test1** and **Test2** pens and delete them by selecting each pen individually, and then clicking the **Trash Can** icon.
4. Once the Data Key is correct, you can assign pens to your chart. Drag in your **Keys** (i.e., sine0, sine1, sine2) to the **Pens** property table. You'll immediately notice your chart is populated with fake data, this is just a preview and does not represent the data returned by your data source.



5. You can modify a selected pen by clicking the **Edit** icon or double clicking the pen to open the **Edit Pen** tab. Here you can change the name of the pen, change the pen style, color, and more. Let's change the color of the pens to make them really stand out.
 - a. Lets start with the sine0 pen.
 - i. Set the **Color** to be red.
 - ii. To return to the **Chart Options** tab, click the double arrow button.
 - b. Repeat the above steps for the remaining two pens, making sine1 green, and sine2 dark blue.






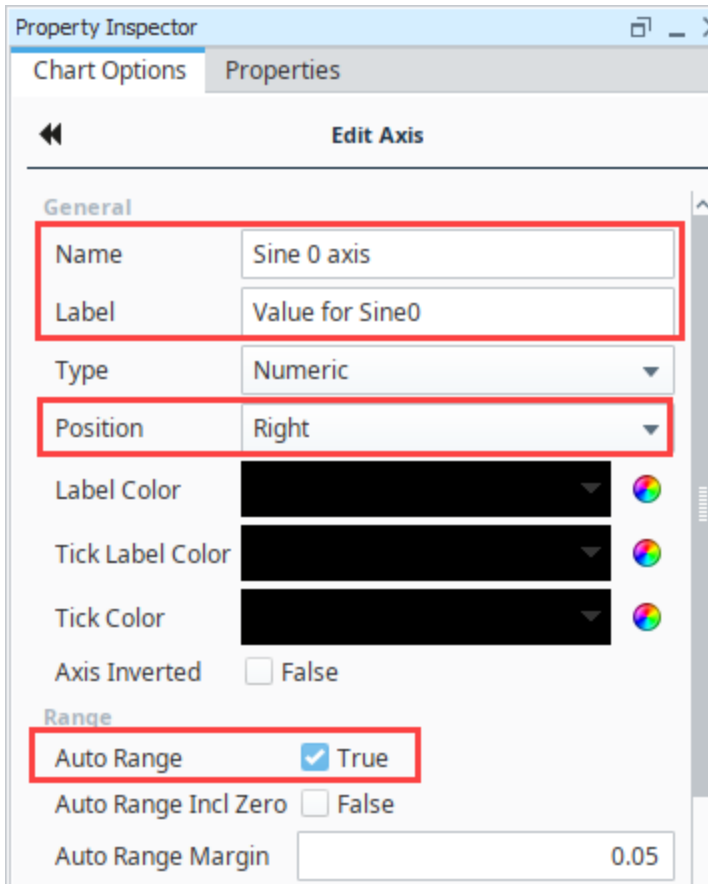
6. Go to the **Preview** tab to view the Timeseries Chart in the report.



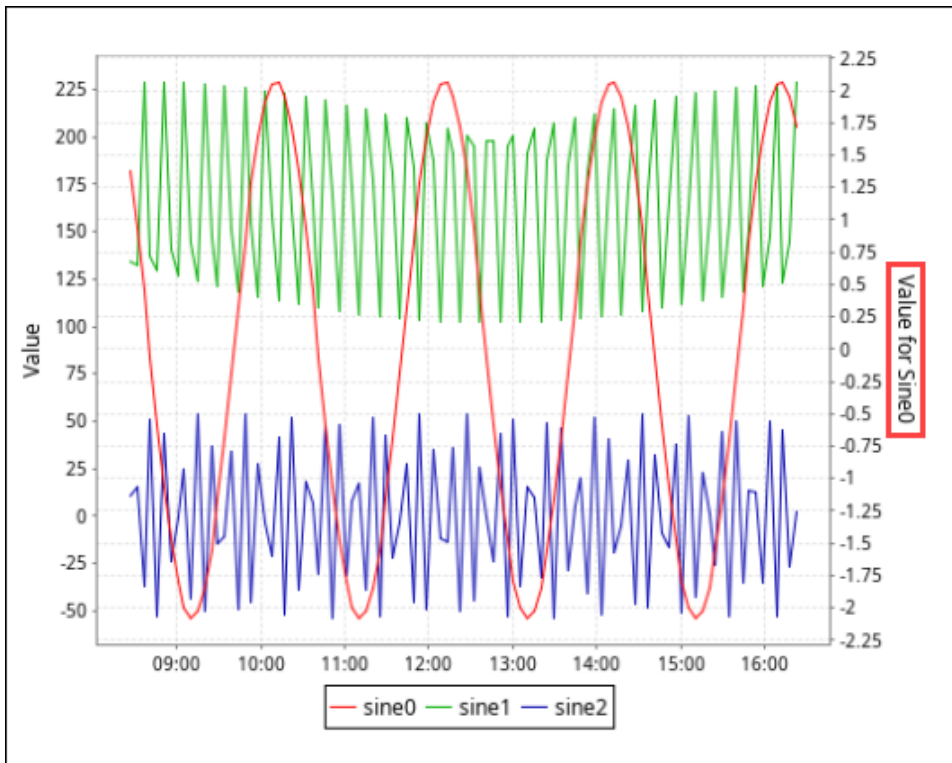
Adding an Axis

Unfortunately, it is a little difficult to see the small changes in our sine0 pen because its range is much smaller than the other pens. Let's fix this by adding a new axis.

1. Go back to the Design tab and select the chart. Click the Add  icon next to the Axes table to add a new axis, then click the Edit  icon below it to edit our new Axis.
 - a. Give it a name of **Sine 0 axis**
 - b. Give it a Label of **Value for Sine0**
 - c. Change the position to right so that it shows up on the right side of the chart.
 - d. We can leave **Auto Range** turned on, so that it will automatically pick the appropriate min and max for the sine0 pen.
 - e. Return to the **Chart Options** tab by clicking the double arrow  button.



2. We now need to tell our sine0 pen to use that new axis. Select the sine0 pen from the pens list and click the **Edit** button. Change the Axis of the pen to be the pen we just created.
3. Navigate to the **Preview** tab to view our finished chart.



Related Topics ...

- [Tag Properties](#)
- [Report Charts](#)

Report Tables

Tables are a major part of Ignition Reporting. Tables are objects that display data in a structured, repetitive format giving users the ability to flexibly layout and organize tabular data in a variety of ways. Their complexity can range from trivially simple to complicated, but fundamentally they do the same thing - take the data given to them and repeat it in a structured way for each row of the data provided. The Reporting engine will automatically create new pages to fit all data within the Table's boundaries while applying formatting preferences each step along the way such as font, size, layout and alignment. Combine that feature with powerful data manipulation and expressive layout tools, and you get an object that often forms the basis of your reports.


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

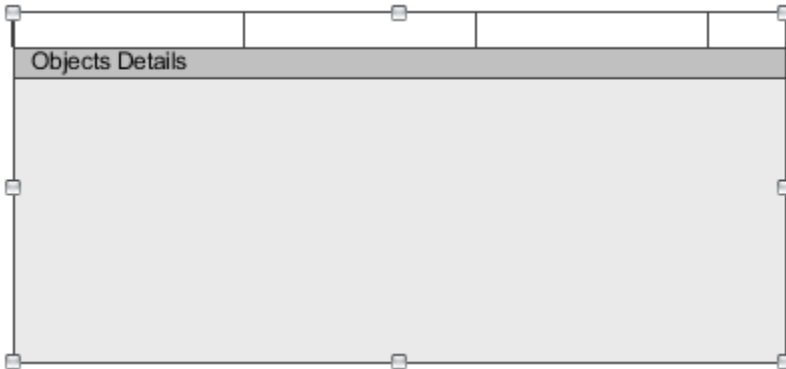
On this page ...

- [The Table Component](#)
 - [Anatomy of a Table](#)
 - [Table Configuration](#)
- [Report Table Features](#)
- [Other Table Components](#)

Let's start by looking at the various parts of the Table component and move into more features.

The Table Component

The Table  component can be created by dragging it from the Report Design Palette, or by simply dragging a Data Source from the Key Browser to a page in your report. A new Table with no bound Data Key is fairly simple looking, but hides a wealth of functionality. When you create a table, by default, you will see a basic table on the report page. The table header will display "Object Details" text until it is set up to use a Data Source.



Once you have your dataset, you're probably thinking about how to display that data in a report. To best understand how to use a table to illustrate your data in a report, you need to have a basic understanding of how the report table works.

Anatomy of a Table

A table has multiple sections, each with its own properties. By default, a Table consists of three sections:

- **Table Body** - Grey area at the bottom of the table template. You can stretch and shrink the table boundaries to position and size a table.

Downtime Data Header			
Downtime Data Details			
Downtime Data Summary			

Table Body Selected

- **Table Rows** - There are three types of table rows: Header, Details, and Summary. More information on these row types can be found on the [Table Rows](#) page.

Downtime Data Header				Standard ▾
Downtime Data Details				Standard ▾
Downtime Data Summary				Standard ▾

Table Row Selected

- **Text Shapes / Cells** - are available only in a structured table row. In Reporting, Text Shapes are commonly referred to as cells. You can select multiple cells with alt or shift.

Downtime Data Header				Standard ▾
Downtime Data Details				Standard ▾
Downtime Data Summary				Standard ▾

Text Shape / Cell Selected

i Text Shapes or Cells

If you're looking in the Project Browser tree, you'll notice a node called 'Text Shape.' In Reporting, the Text Shape is also called a 'cell.' For the sake of clarity, Text Shapes will be referred to as 'cells,' unless we are referring to the Text Shape component.

The following example shows the basic anatomy of a table with data populated in the Header, Details, and Summary rows. Once you're finished designing your report, go to the Preview panel and check your results. You can continue to navigate between the Design and the Preview panels

making changes until you get your report just right!

Order Number	Date Shipped	Sales Person	Amount
Sales Data Header			
@Order Number@	@Date Shipped@	@Sales Person@	@Subtotal@
Sales Data Details			Amount: \$@total.Subtotal@
Sales Data Summary			

Each section (i.e., Table, Rows, and Cells), when selected, has its own unique properties, along with some common basic properties that appear in the **Properties** tab of the Property Inspector. Refer to the [Table](#) page in the Appendix for a complete list of all Table properties.

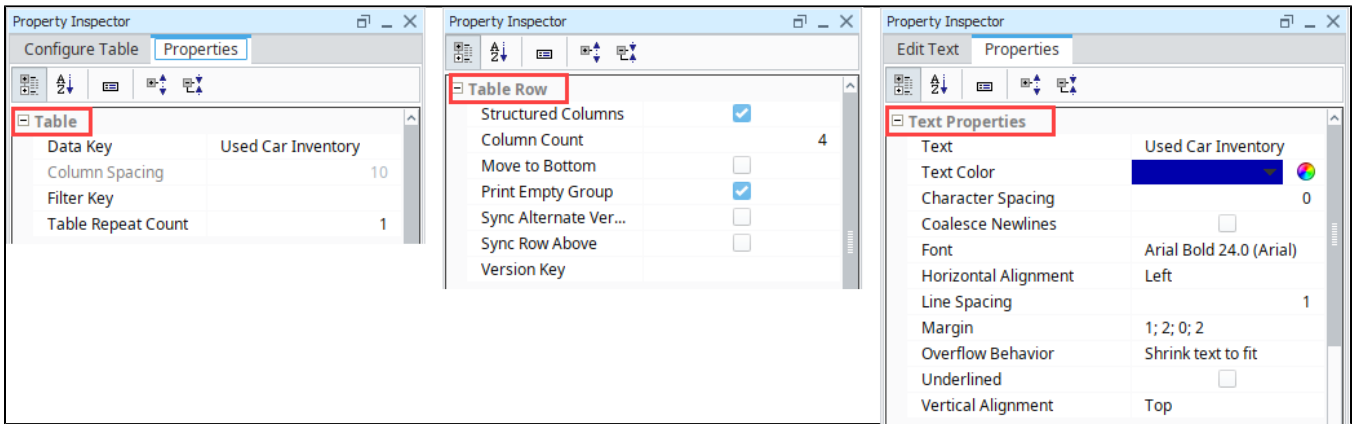
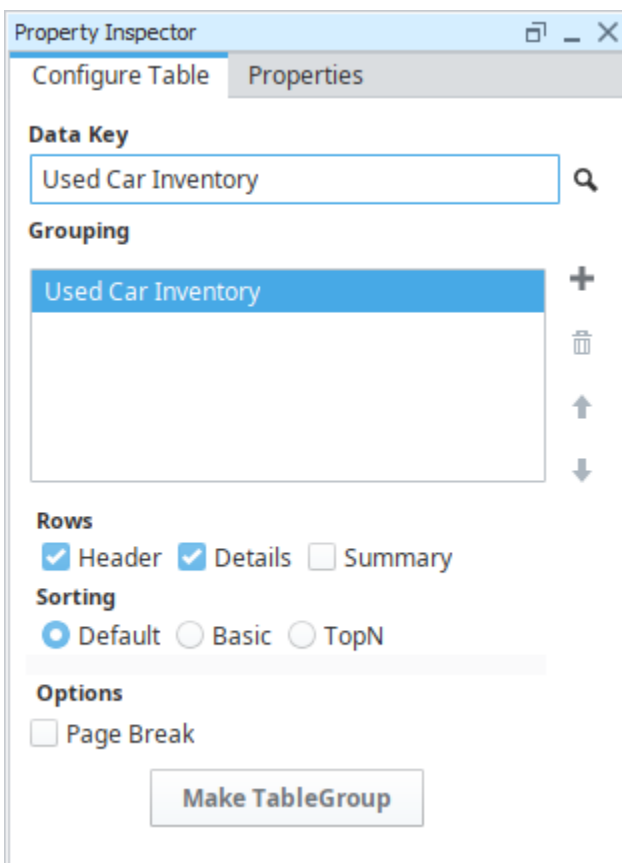


Table Configuration

When you first drag a table into the Design panel of your report, the **Configure Table** tab will appear in the Property Inspector. You will see that it has a default **Data Key** of "Objects", which matches the **Details** row in the table. The first step when configuring your table is to drag your **Datasource** from the Key Browser to the Data Key field in the Configure Table tab. Changing the Data Key will also change the label on the Details row making it easier to determine what you are looking at when editing multiple tables or **Groupings**. Depending on how you want to present your data, you may also want to add Header and Summary Rows.

Note: The **Data Key** field represents the data source that the table will pull records from. The Table can only have a single data source assigned to it. If the table needs to display values from multiple data sources, then a **Nested Query** should be used to collect all the values, and then assigned to the table's Data Key setting.



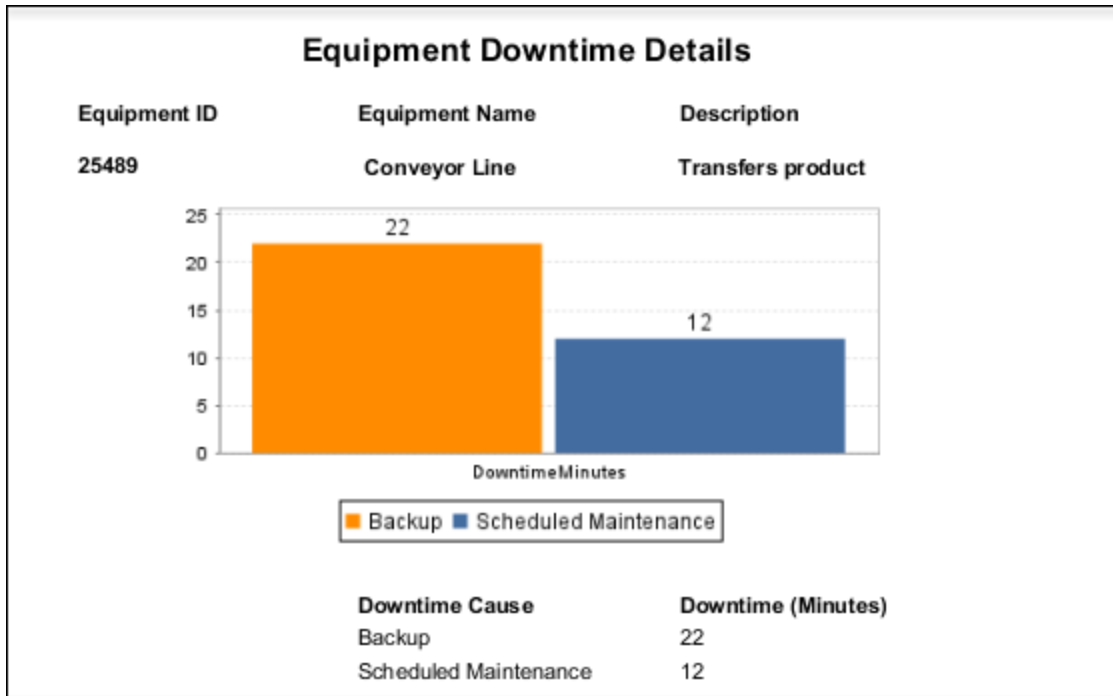
Once you add a Data Key to your table, you can start designing. There are many tools built into the Table interface that you can use to help you layout and organize the data in your table. Now all you need to do is decide how you want to organize and display the data in your report.

If you have a lot of tabular data, you probably want to use a Standard row. You might start by simply dragging your Data Keys from the Key Browser to columns in your Details row. Then, you might give each Header column a label. If you have a bunch of numbers that you want to add together, find out the 'min', 'max,' you can create a Summary row, enable the [Show Calculations](#) property, and choose from the list of calculation keys. Lastly, you might want to make some cosmetic changes to the font size, font color, or changing the Header row from [structured to unstructured](#) so you can add an image, chart, text shapes, etc.

Report Table Features

Tables have a lot of built-in features that give reporting users the ability to create simple tables and increasingly more complicated tables. The table features are quite powerful, not only do they allow you to organize your tabular data, they also provide sophisticated data manipulation and layout tools. There are three types of table rows: [Header, Details, and Summary rows](#) that make up the body of a table. You can break tables down by data keys that share a common value using Dataset Grouping. The different types of rows can be independently enabled for each level of grouping, each group having its own Header, Details, and Summary rows. Additionally, the keys from '[Show Calculations](#)' and other [keychain functions](#) are supported for any level of grouping.

By default, when you initially create a table all the row types are [structured](#). A structured row has a variable number of columns and allows you to organize your text based table data, whereby, an unstructured row is highly customizable allowing you to place images, text shapes, and charts within the table row. [Unstructured rows](#) are perfect for placing charts anywhere within the row of the table. An unstructured Details row typically works best when used in conjunction with [Grouping Data Inside of Tables](#) or [Nested Queries](#) so that each row has a chart with data from that group or query.



Another feature is [Table Row Versioning](#) which gives you the option of displaying rows with a different format to make them stand out, such as creating an alternate background row color, making the first row different, or even creating your own custom row version. You have a choice of using any of the Built-in row versioning options or creating a custom version.

Candy Bar Name	Inventory	Cost per Candy	Value of Candy Supply
Pay Day	22	\$ 2.10	\$ 46.20
Baby Ruth	44	\$ 2.40	\$ 105.60
Milky Way	16	\$ 1.90	\$ 30.40
KitKat	28	\$ 1.85	\$ 51.80
Butterfinger	7	\$ 2.30	\$ 16.10
Snickers	19	\$ 2.00	\$ 38.00
Twix	13	\$ 2.05	\$ 26.65

Report Tables also support [Table Grouping](#) which is an easy way to add multiple Data Sources to an existing table in a report using Peer Tables and Child Tables. Peer Table groups allow the second table to begin exactly where the first table ends. Child Table groups allow you to nest one table inside another. What's really nice about Table Groups and Nested Data Sources is that you can create Summary Tables for categories of items or drill-down charts all in one report .

Once you get familiar with all the report table features, you can easily design professional tables for your reports.

Other Table Components

There are a few [other table components](#) in the Report Module, Simple Table and CrossTab Table, and they both behave a little differently. Selecting the right table depends on the type of data you have and how you want to display it.

The [Simple Table](#) is similar to a basic table that dynamically creates new rows and columns for rows returned by the Data Keys on the component. With a Simple Table, you can very quickly add a table inside a report.

The [CrossTab Table](#) is commonly used to summarize the relationship between two groupings of data by showing summaries of cross sections of the datasource. The CrossTab Table has lots of repetitious data, a datasource that provides at least two columns of data which are repetitious compared to the number of rows, and one or more columns that represent a value that requires a calculation. This table will stretch both its height and width to accommodate the underlying dataset.

[Related Topics ...](#)

- [Report Design](#)
- [Report Design Tools](#)

- [Table](#)

In This Section ...

Getting Started with the Report Table

The [Report Tables](#) is another Reporting component, like report charts, that has a lot of features and functions that help you create meaningful reports. Tables are a major part of Reporting and simple to create. This page will give you a great head start for creating your own standard report tables using the built-in design tools.

Creating a Report Table

Creating a table is a simple process, but the order of how you create a table is important. The first thing you need before creating your report table is to create a [data source](#). Your data source can be a query, script, or CSV file. Next, add a table and configure the data in the table that you want to display. Lastly, you can use the built-in design tools to enhance your reports. You can add images to your reports, change row types, show mathematical calculations, and change the font style, size, or color of the text.


Let's get started!

This example creates a standard chart that tracks the downtime minutes for each production line. We'll configure the data in the table, do a calculation, add an image, and make a few property changes so you get a sampling of the power of report tables.

Adding a Datasource

Datasource

This example uses a CSV file as its datasource. You can use the one shown in the code block below, or create your own.

1. In the Data panel, click the **Add**  icon to add a datasource, and select **StaticCSV** from the resulting list of options. From the code block below, copy the text and paste it into the **Data** field. This is the data that we will use in this example.

Downtime Data

```
Production Line, Downtime
Line A, 75
Line B, 92
Line C, 43
Line D, 54
Line E, 66
Line F, 80
Line G, 40
Line H, 88
```

On this page ...

- [Creating a Report Table](#)
 - [Adding a Datasource](#)
 - [Table Configuration](#)
 - [Finishing Touches](#)
- [Preview a Report](#)

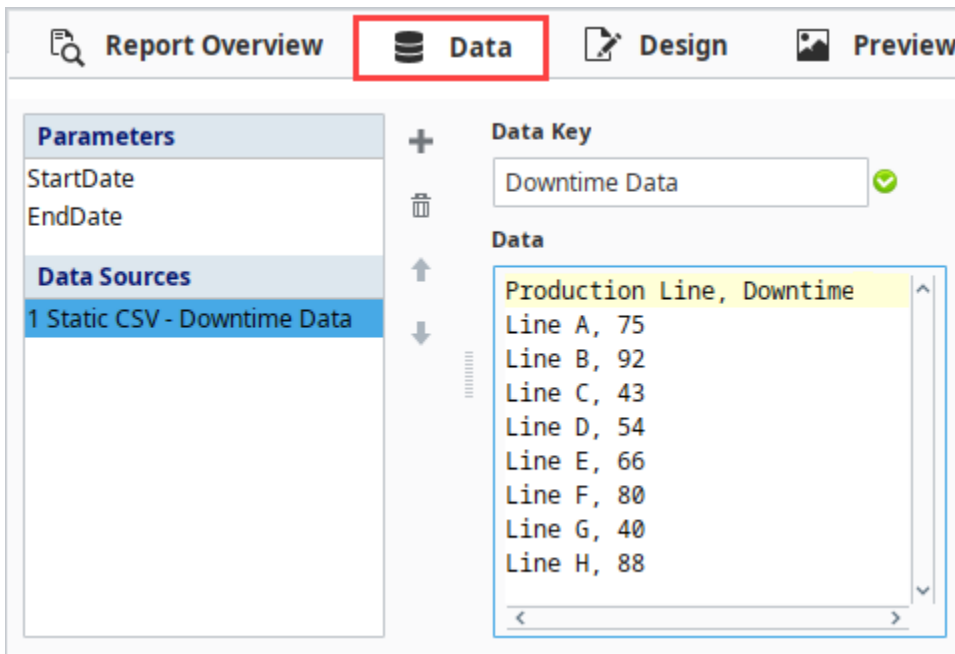
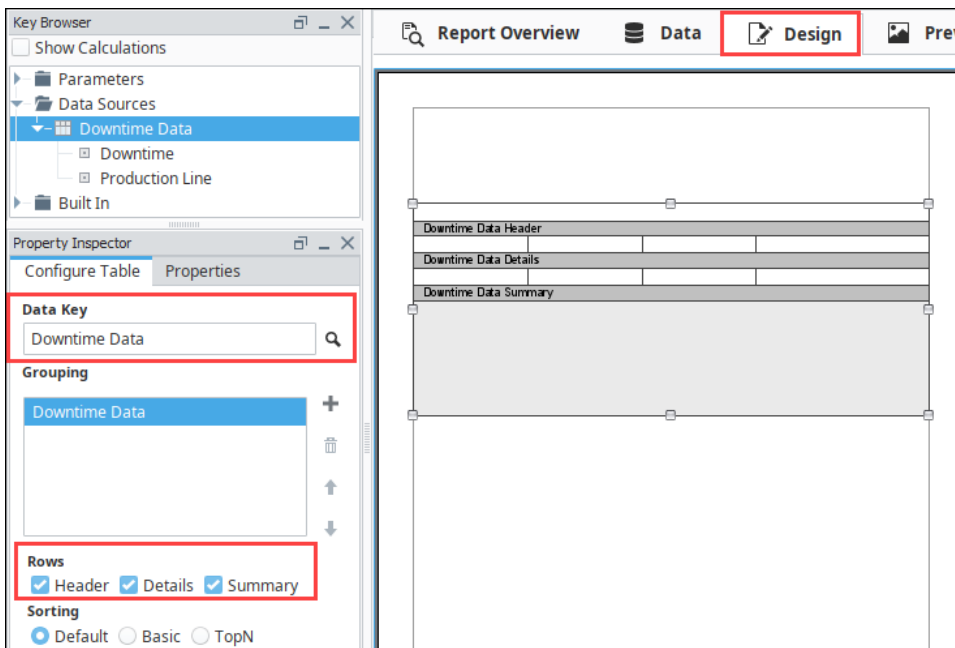
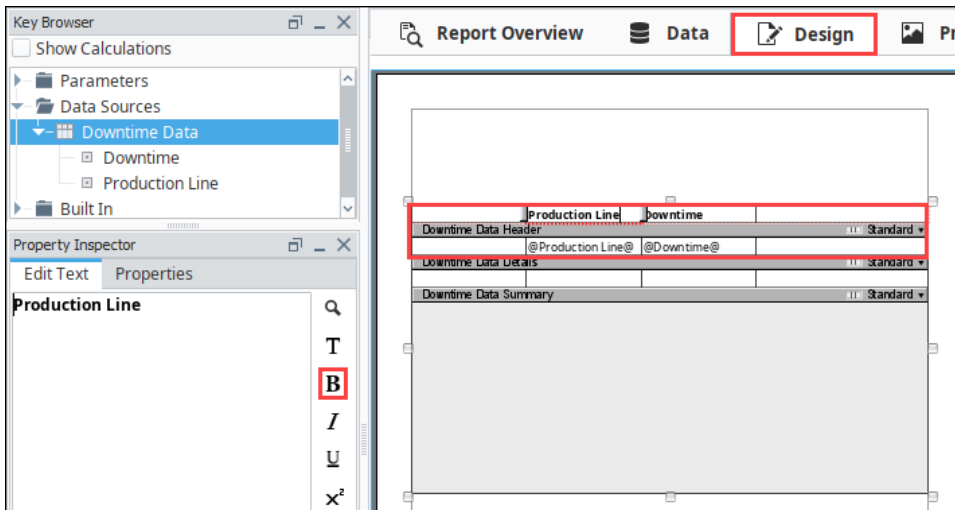


Table Configuration

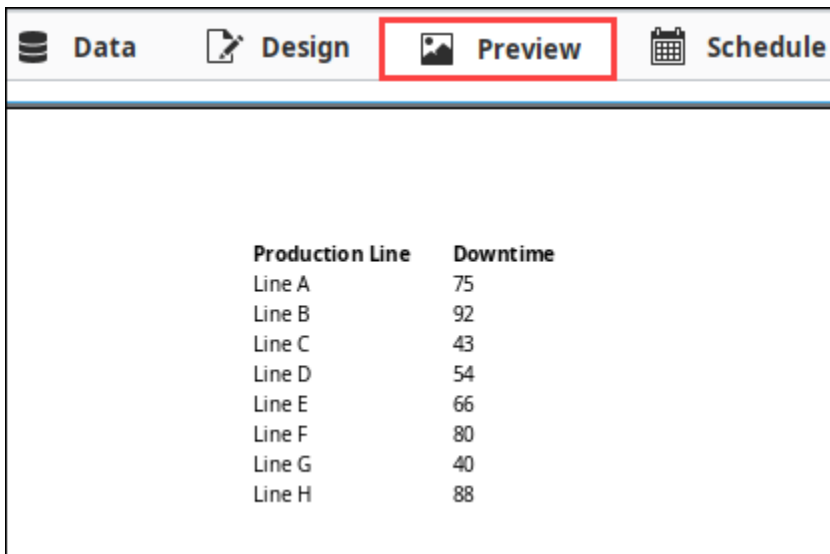
1. Go to the Design panel, and drag a Table component into the page.
2. From the Key Browser, expand the **Datasources** folder. Drag your data source (i.e., Downtime Data) into the Data Key field of the Configure Table tab of the Property Inspector.
3. Add **Header** and **Summary** rows by marking the checkboxes.



4. Now, let's configure the data in the Table. With the Table still selected, drag each of the data keys to a cell on the table above the Data Details row: (i.e., Production Line, and Downtime).
5. Next, let's add column headers by selecting a cell above one of our details cells, and enter the appropriate column header name (i.e., Production Line, and Downtime [Minutes]). You can change the font style, size, and color using the functions on the right side of the **Edit Text** tab. Bold the headers using the **B** button.



6. Go to the Preview panel to view your report with the data. It's very common to make formatting changes once you see how your data looks in the report.



Finishing Touches

1. Back in the Design panel, let's make a few changes. Select the "@Production Line@" in the Details row and click on the **Properties** tab to see all the cell properties. Let's change the Horizontal Alignment property from left to Center so the data falls nicely under the header. Repeat this for "@Downtime@."

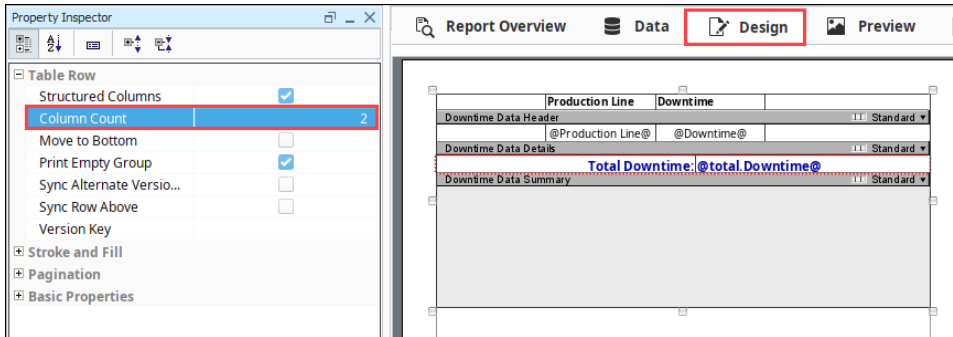


Checking your table in the report

At any point during the table configuration process you can go to the **Preview** panel and see how your table looks with all the data. Configuring the data and designing the table layout is an iterative process. You can go back and forth between the **Design** and **Preview** panels as many times as you want making and viewing changes until you get the results you want.

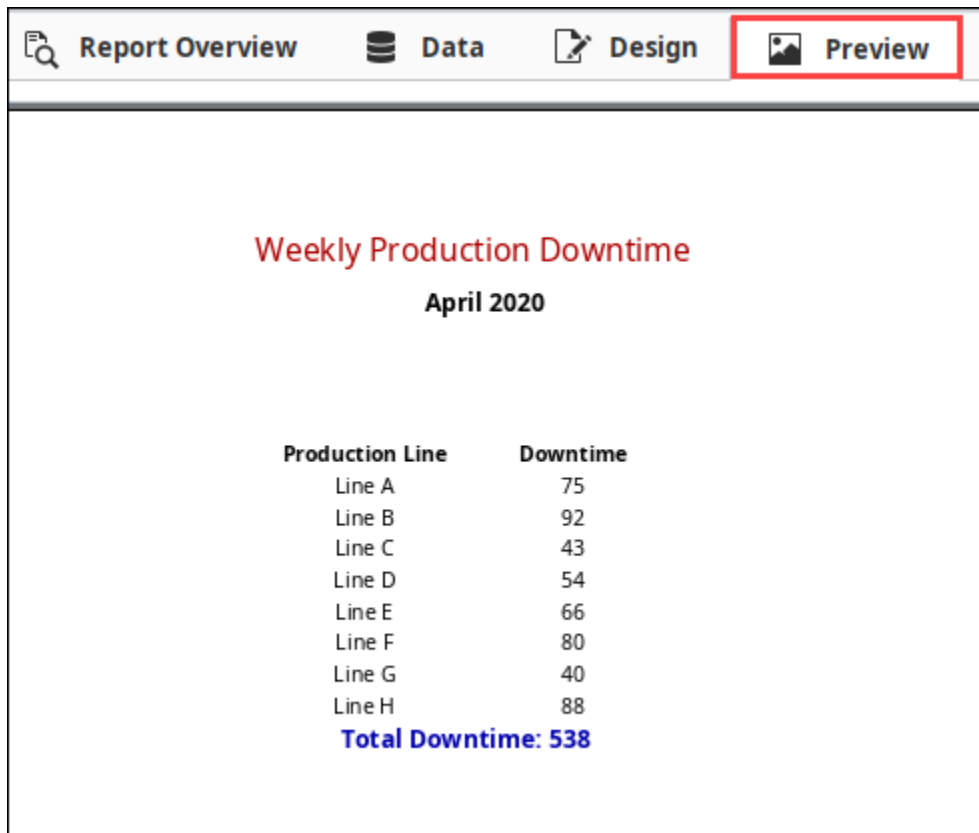
2. Let's add the total downtime by using the **'total'** calculation. In the Key Browser, set **Show Calculations** to **'true.'**
3. The Data Summary row is a good place to show the total number of downtime minutes. Expand the **Downtime** data key and you'll see a list of calculations you can use. Drag the **'total'** key to a cell in the Data Summary row.
4. You can even give the total number of downtime minutes a title, make it blue, and bold it (i.e., Total Downtime). If the cell is too small to show the title, you can make the cell larger by dragging the cell border either to the left or right. You can also select the Data Summary row, go to the Properties tab and change the **Column Count** (i.e., 2), as shown in the screenshot below.

5. Click the **Preview** tab to check your work.



Preview a Report

The Preview panel lets you validate that all your data is organized and formatted, and you're completely satisfied with how your data is presented in the report. Once you're satisfied, you can [schedule the report](#) to be run and delivered automatically.



Related Topics ...

- [CrossTab and Simple Tables](#)
- [Nested Queries](#)

Table Rows

Row Types

Rows are an important fundamental aspect of tables. There are three types of table rows: **Header**, **Details**, and **Summary**. The different types of rows can be independently enabled for each level of **Grouping**, and for each table in a **Table Group**. Adding or removing the Header, Details, or Summary rows is as simple as selecting the Row boxes in the Configure Table tab of the Property Inspector. A row can also be **structured or unstructured**. A structured row having a variable number of columns, and unstructured row having no columns so you can add images, charts, data, and text shapes anywhere in the row. Additionally, **Table Row Versioning** gives you the option of conditionally displaying rows with a different format to make them stand out.

To learn more about Tables, go to [Report Tables](#) and [Table](#) in the Appendix.

Header Row

The Header Row allows for a single row to be placed before the Details rows. This is commonly used to create a header for the Table. In many cases where one header is used, the other header could be used equivalently in its place. An interesting feature of the header row is the **Reprint When Wrapped** property which allows the header row to be reprinted on a new page when its data crosses a page boundary. To disable this feature, uncheck the **Print When Wrapped** property in the Properties tab.

Automatic Text Resizing

If text in a Header Row is so long that it will result in truncation or overflow, the size of the text will automatically resize to accommodate.

Detail Rows

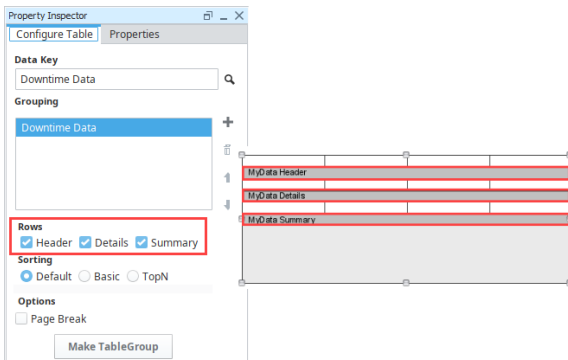
The Detail Rows typically represent the majority of the data on a table or commonly referred to as the "middle" rows. Once you configure the Detail rows with your datasource, the table displays your data in a structured, repetitious format. You can customize the layout and organize the data to determine how you want your data to look on the report. You can even disable Detail Rows in unusual situations such as only displaying aggregate summaries.

Summary Row

The Summary Row is like the Header row only it prints at the bottom of the table, and is typically used to display aggregates for keys. They are typically used in conjunction with some of the [Show Calculation keys](#), such as count, total, running total, etc.

Configuring Header, Details, and Summary Rows

Adding Header, Details, and Summary Rows is super easy. Once a Table component is in your Design Panel and your Datasource is populated in the Data Key field, check the **Header** and **Summary** boxes. The Header, Details, and Summary rows will be added to your report. The **Details** box will be checked by default. To remove the Header, Details, or Summary rows, uncheck the applicable rows.



Structured vs Unstructured Rows

Structured Rows behave like a row in a spreadsheet: a series of columns are horizontally adjacent to each other. In the case of a row, the columns are Text Shapes. A structured row can have a variable number of Text Shapes. Structured rows provide more control over the layout, wrapping, and

On this page ...


- [Row Types](#)
 - [Header Row](#)
 - [Detail Rows](#)
 - [Summary Row](#)
 - [Configuring Header, Details, and Summary Rows](#)
- [Structured vs Unstructured Rows](#)
- [Report Data Configuration](#)
 - [Sorting](#)
 - [Filtering](#)



Table Rows

[Watch the Video](#)



organization of text based table data. Since structured rows only allow for [Text Shapes](#), they can not contain components like images, barcodes, charts, and text shapes.

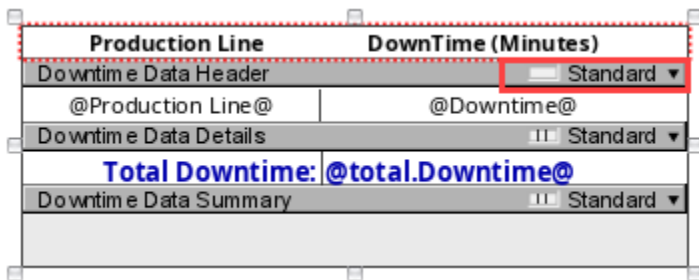
To view the properties of the Details row (or any row), super-select the row by clicking on the Details row (dark bar in the table), or clicking the **Details** node in the Project Browser tree. You can confirm that a row is structured by looking at the **Structured Columns** property in the Properties Tab, or by looking at the **Row Structure** icon  next to the **Standard** row version label. You can also change the number of columns in a row, using the **Column Count** property.




You can tell when a row (any row) is selected by looking at the Project Browser tree, or by looking at the red outline of the rows in a table. A row's content is above its respective dark row bar.

Unstructured Rows are functionally very similar to structured rows, but they do not offer the same column based text constraints. Instead, Unstructured Rows allow you to place any other report component inside of the row. Unstructured rows are highly customizable rows allowing you to place data, text shapes, images, or charts anywhere you want them within the row.

There are two ways to make a row unstructured once your row is selected:

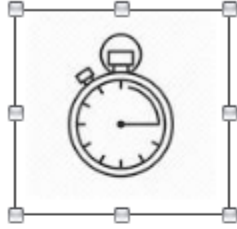
- In the Property Tab of the Property Inspector, set the **Structured Columns** property to **'false.'**
- In your Table, click on the **Row Structure**  icon on the right side of the row to make the row unstructured . You can toggle the Row Structure icon to switch between a structured row and unstructured row.



Production Line	DownTime (Minutes)
Downtime Data Header  Standard ▼	
@Production Line@	@Downtime@
Downtime Data Details  Standard ▼	
Total Downtime: @total.Downtime@	
Downtime Data Summary  Standard ▼	

Now that the Header row is unstructured, drag the Header row down the page making room to add any components from the Report Design Palette, and place it anywhere in the row. For example, you can add a **Text Shape** to the report title and a **Line Shape** to separate the image from the report data.

Production Line Downtime Report



Production Line

Downtime (Minutes)

Downtime Data Header Standard ▾

@Production Line@

@Downtime@

Downtime Data Details Standard ▾


Total Downtime: @total.Downtime@

Downtime Data Summary Standard ▾

--

Always be sure to verify your report appearance using the **Preview** panel.

Production Line Downtime Report



Production Line	Downtime (Minutes)
Line A	75
Line B	92
Line C	43
Line D	54
Line E	66
Line F	80
Line G	40
Line H	88
Total Downtime: 538	



Report Data Configuration

In addition to configuring rows, you can also configure how you want your data to appear in a report using the **Sort** and **Filter** functions in the Table component.

Sorting

Sorting orders your data by a single data key or list of data keys. There are three types of Sorting in Tables.

- **Default** - data is sorted based on the order in which it is retrieved.
- **Basic** - takes a list of data keys and sorts by the first one. If the sort results in a tie, the tie will be resolved by the next data key in the list, and so on.
- **TopN** - uses a single key path, with a **Count** value that allows a limit to the number of rows that are processed.

Basic and **TopN** sorts can be configured for either ascending () or descending () sorts. They can also utilize aggregate (calculation) keys.

The **TopN** sort option, **Include Others** **Include "Others"** if selected, will include all values outside of the specified **Count** range by compressing them into a single row.

Sort and Filter Examples

The CSV dataset containing all the data and the table used for the following Sort and Filter examples are shown below.

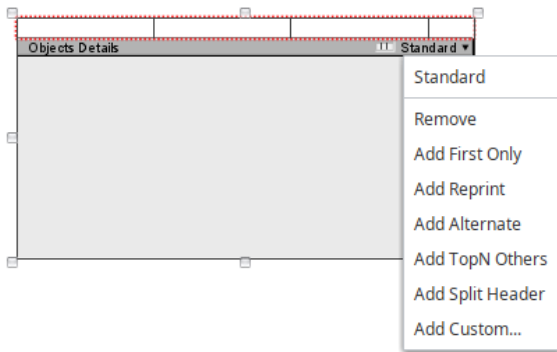
Dataset	Table in the Design Panel

Table Row Versioning

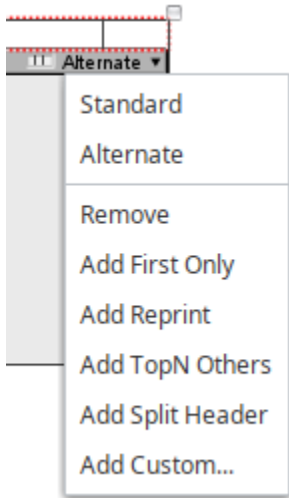
Table Row Versioning

Table Row Versioning allows you to conditionally display rows of data in different formats. It is used to make certain data stand out or to make a report more legible. It allows you to do things like create alternate background row colors, or make the first row different, or show negative valued rows differently. You will see a list of your existing row versions (only one named "Standard" appears first in the list) followed by options to add new row versions. When creating new row versions, the currently selected row will be duplicated as the new row version. From here, you can modify your row in any way. You can make it similar to the Standard row or completely different. Once you added a few row versions, you can swap between them by selecting one from this list.

To use Row Versioning, click on the word "**Standard**" on the far right of a row. Row versions are either **Built-in** or **Custom**, and may be specified with a version key expression. They are applicable to the [Header and Summary Rows](#), but are most often added to the **Detail Row**.



Once a Row Version has been added, it becomes selectable in the dropdown. The word "**Standard**" also gets replaced with whatever Row Version has currently selected to edit. In this case, an Alternate Row Version was added, and it is currently selected.



Note: It is important to always be aware of what Row Version you are currently working on by checking the name of the Row Version on the right of the Table row. After navigating away from the Design panel, the Table will automatically display the standard Row Version on return to the Design panel.

Here is a list of the Row Version types and their descriptions.

Row Versions	Description
Standard	Default row version. When adding a new Row Version, it will start off as a copy of the Standard Row Version.

On this page ...

- [Table Row Versioning](#)
 - [Using Row Versions](#)
 - [Sync Alternate Versions](#)
 - [Custom Row Version](#)




Table Row Versioning

[Watch the Video](#)

Remove	Removes the currently selected Row Version. The Standard row cannot be removed.
First Only	Applies only to the first instance of the row. Good for showing header information without using an upper level Detail row. This is not the same as a Header row.
Reprint	Applies to every page after the first. Good for one time headers or (continued) indications to save space.
Alternate	Applies to every other row starting with the second row. Good for changing the background color on alternate rows.
TopN Others	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 have been split due to excessive height. Good for providing "Continued" type indicators.
Custom	Create your own Row Version. When this option is selected, you must enter a label for the new row version. Instead of being used automatically like with the other Row Versions, all custom Row Versions are driven by the Version Key Property.

Using Row Versions

Once a new Row Version has been added to the table row, it is easy to configure each row version to appear visually distinct.

When using a Table to display a large number of rows, an Alternate Row Version can be added to alternate between two fill colors, which in turn improves readability. We start off by adding a table to the report, assign a data source to the table and add some data keys to the text shapes in the Details row. In this example, we will alternate between white and orange, so we will leave the standard row's **Fill** property disabled to use the default color of our paper.

The screenshot shows a report table with the following structure:

Candy Bar Name	Inventory	Cost per Candy	Value of Candy Supply
CandyData Header			
@Candy Bar@	@Inventory@	@Cost@	@Cost*Inventory@
CandyData Details			

The 'Stroke and Fill' property window for the 'CandyData Details' row is open, showing the following settings:

- Fill:
- Fill Color: [Black color swatch]
- Opacity: 1
- Stroke Style: Hidden
- Stroke: [Empty field]

An Alternate row was added to the details row of the table. On the Alternate row, we can enable the **Fill** property and set **Fill Color** to an orange background.

The screenshot shows the same report table as above, but with the 'CandyData Details' row highlighted in orange. The 'Stroke and Fill' property window for this row is open, showing the following settings:

- Fill:
- Fill Color: [Orange color swatch]
- Opacity: 1
- Stroke Style: Hidden
- Stroke: [Empty field]

Taking a look at a Preview of the report, we can see that the Alternate row with the orange color is automatically used on every other row starting with the second row.

Candy Bar Name	Inventory	Cost per Candy	Value of Candy Supply
Pay Day	22	\$ 2.10	\$ 46.20
Baby Ruth	44	\$ 2.40	\$ 105.60
Milky Way	16	\$ 1.90	\$ 30.40
KitKat	28	\$ 1.85	\$ 51.80
Butterfinger	7	\$ 2.30	\$ 16.10
Snickers	19	\$ 2.00	\$ 38.00
Twix	13	\$ 2.05	\$ 26.65

Sync Alternate Versions

Once a new row version has been created, it is not linked to the standard row in any way, so configuration changes made to one row will not be automatically applied to other row versions.

The common case where this behavior becomes a problem is when the width of one or more Text Shapes in a row are modified. Here we see a row with an Alternate version using a different **Fill Color**. After the Alternate row version was created, the second and third Text Shapes on the Standard version has their **Width** increased, causing the text inside to shift. However, the Text Shapes on the Alternate version will not automatically resize themselves by default, so the Preview Panel shows offset columns on the Alternate version.

Motor	Site A	15
Motor	Site A	23
Conveyor Line	Site B	148
Pallet Wrapper	Site A	58
Motor	Site C	96

Instead of manually adjusting the Text Shapes on the Alternate version, we can toggle the **Sync Alternate Versions** property on the Standard row. Assuming both columns have the same value for **Column Count**, enabling **Sync Alternate Versions** will automatically resize the widths of Text Shapes on the alternate row version to match the widths of the standard version Text Shapes.

Table Row

Structured Columns	<input checked="" type="checkbox"/>
Column Count	4
Move to Bottom	<input type="checkbox"/>
Print Empty Group	<input checked="" type="checkbox"/>
Sync Alternate Versions	<input type="checkbox"/>
Sync Row Above	<input type="checkbox"/>
Version Key	

Heading back to the Preview Panel, we see that the Text Shapes on both rows are now synchronized.

Motor	Site A	15
Motor	Site A	23
Conveyor Line	Site B	148
Pallet Wrapper	Site A	58
Motor	Site C	96

Custom Row Version

Custom row versions are ideal when the built-in Row Versions don't fit your needs. Custom versions are identified by a string-based name, and will be used when the Version Key property is a string that matches the Row Version name. If that string equals the name of a Row Version, that Row

Version will be used. An invalid string will default back to normal built-in Row Version behavior. The Version Key property also accepts Data Keys used in expressions, using the same syntax that [Keychain Expressions](#) use. For example, if you wanted to highlight red in all of the rows where the Inventory is less than 20, you can use a simple expression of:

```
Inventory<20?"Row3"
```

Where "Row3" is the name of my custom Row Version enclosed in quotes. This can help me keep track of low inventory, excessive downtime, or anything that may be important to highlight.

Candy Bar Name	Inventory	Cost per Candy	Value of Candy Supply
Pay Day	22	\$ 2.10	\$ 46.20
Baby Ruth	44	\$ 2.40	\$ 105.60
Milky Way	16	\$ 1.90	\$ 30.40
KitKat	28	\$ 1.85	\$ 51.80
Butterfinger	7	\$ 2.30	\$ 16.10
Snickers	19	\$ 2.00	\$ 38.00
Twix	13	\$ 2.05	\$ 26.65

Finally, it is actually possible to use many different Row Versions in conjunction with each other. You can set up multiple custom rows and have a more complex expression that helps decide when a particular Row Version gets used. You can also use multiple built-in rows, and even a combination of built-in rows and custom rows. In the event of a conflict between the custom row and a built in row, the custom row will take precedence. For example, when combining my custom Row3 with the alternate rows, you can see that the row Snickers would be the next alternate row in the table, but since it has less than 20 inventory, it uses the custom Row3 instead.

Candy Bar Name	Inventory	Cost per Candy	Value of Candy Supply
Pay Day	22	\$ 2.10	\$ 46.20
Baby Ruth	44	\$ 2.40	\$ 105.60
Milky Way	16	\$ 1.90	\$ 30.40
KitKat	28	\$ 1.85	\$ 51.80
Butterfinger	7	\$ 2.30	\$ 16.10
Snickers	19	\$ 2.00	\$ 38.00
Twix	13	\$ 2.05	\$ 26.65

You can also reference multiple row versions in the same conditional expression:

```
Inventory<20?"Row3" : Inventory<40?"Row4" : "Row5"
```

Related Topics ...

- [Table](#)
- [Grouping Data Inside of Tables](#)
- [Table Groups](#)
- [Report Tables](#)

Charts Inside of Tables

Adding Charts Inside of Tables

Adding charts inside of tables provides a lot of flexibility designing reports as well as organizing and displaying data in a report. The most common way of adding charts inside tables is using an unstructured row and placing the chart inside the row. Unstructured rows are highly customizable allowing you to place charts anywhere within the row of the table.

There are two common locations to place a chart inside a table:

- Using a Header followed by a chart component.
- Using a Details Row followed by a chart component.

It is very common to add a chart in an unstructured Header row because then the chart will be at the top of the first page of a report. An unstructured Details row typically works best when used in conjunction with [Grouping Data Inside of Tables](#) or [Nested Queries](#) so that each row has a chart with data from that group or query.

On this page ...

- [Adding Charts Inside of Tables](#)
 - [Adding a Chart Inside a Header Row](#)
 - [Adding a Chart Inside a Details Row](#)
- [Using a Chart in a Table with Nested Queries](#)
 - [The Tables](#)
 - [Queries](#)
 - [Configuring the Chart](#)



Charts Inside of Tables

[Watch the Video](#)

Adding a Chart Inside a Header Row

This example shows several pieces of hardware including how many pieces were produced, and how many pieces were shipped. A Bar Chart was added after the Header row in the table.

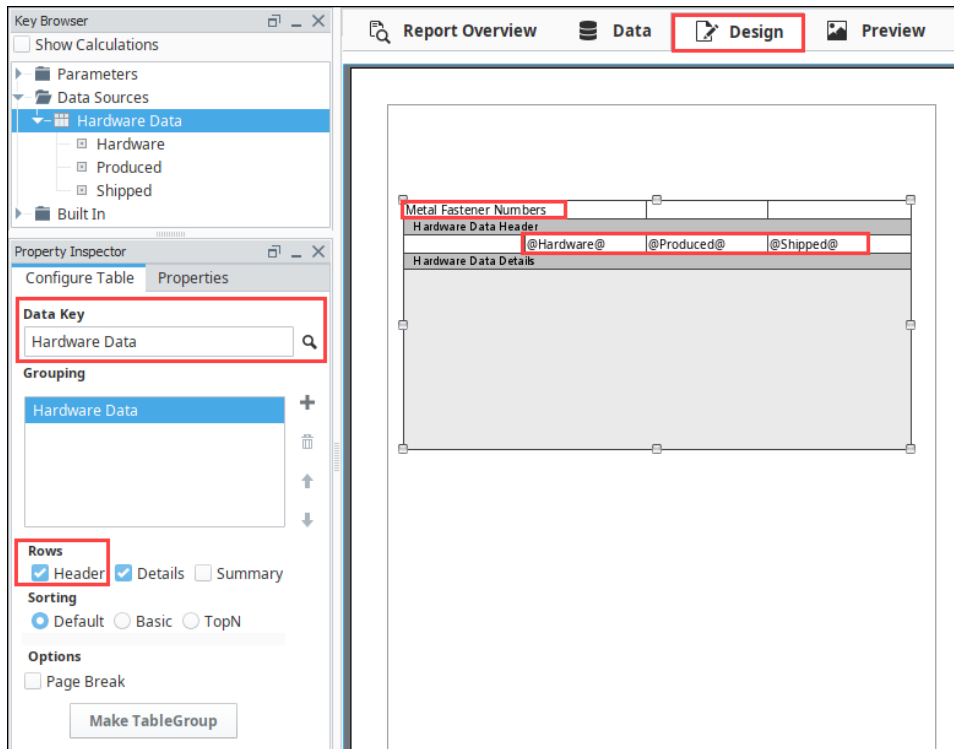
1. In the **Data** panel, create a Static CSV data source (i.e., Hardware Data). The dataset is shown below.

Hardware Dataset

```
Hardware, Produced, Shipped
Nails, 95, 85
Screws, 80, 60
Bolts, 50, 47
```

2. In the **Design** panel, drag a Table component on to your report. Drag your **Datasource** (i.e., Hardware Data) from the **Key Browser** to the **Data Key** field, and click the **Header** box in the **Configure Tab**.
3. Next, drag each **Data Key** (i.e., Hardware, Produced, and Shipped) to a column in your table in Data Details row.

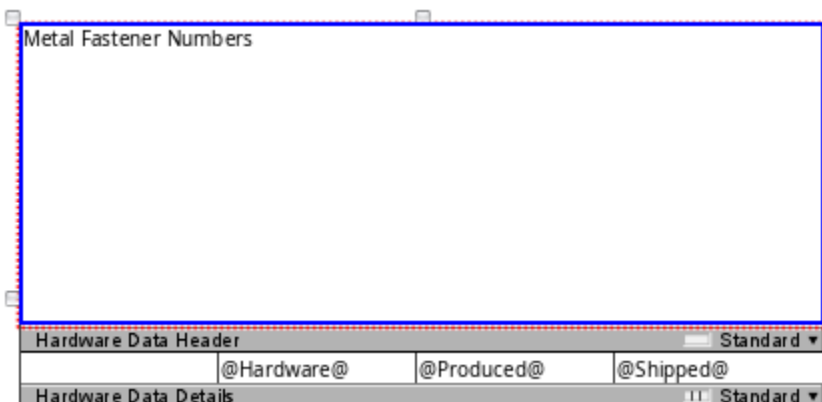
- Enter the Header name (i.e., Metal Fastener Numbers Report) in the left column of the Data Header row.



- Select the Header row and click on the **Row Structure** icon  to make the row unstructured  so you can add a chart.

Note: You can toggle between a structured and unstructured row by simply clicking the Structure Row icon.

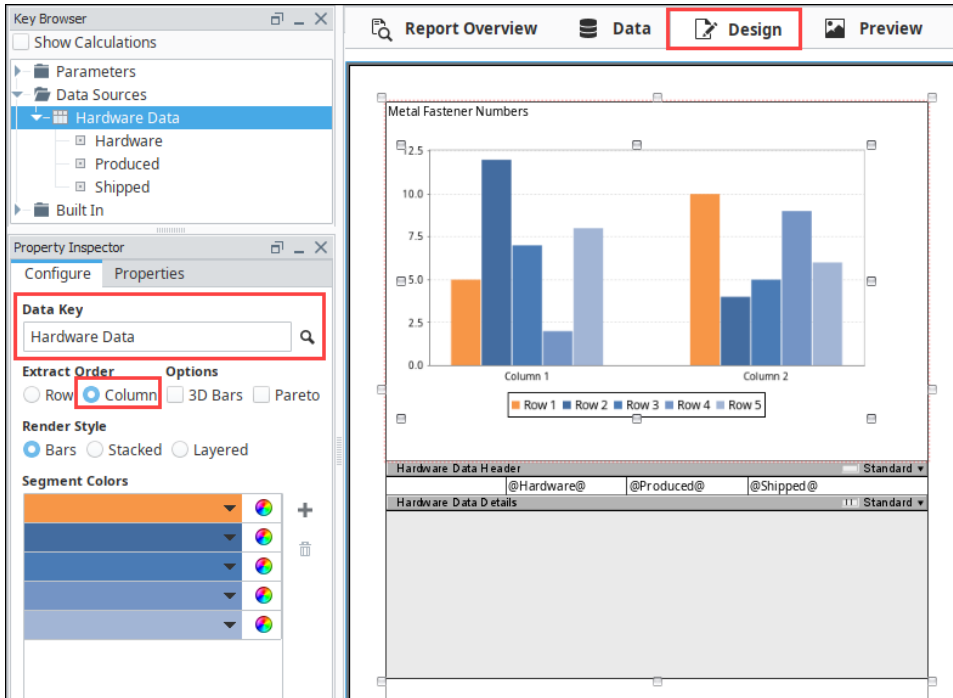
- With the Header row selected, drag the row down the page to make room for the chart: click and drag on the dark gray bar titled "Hardware Data Header".
- Drag in a Bar Chart from the Report Design Component Palette into the unstructured Header row and expand it. While dragging the chart into the header, you'll notice a blue outline around the Header as you hold the mouse button down while the cursor is in front of the header. This signifies that the chart will be placed directly into the header row, as opposed to in front of the table.



- With the chart is selected, drag a **Datasource** (i.e., Hardware Data) to the **Data Key** of the **Configure Tab**. Set the **Extract Order** as 'Column

Extract Order

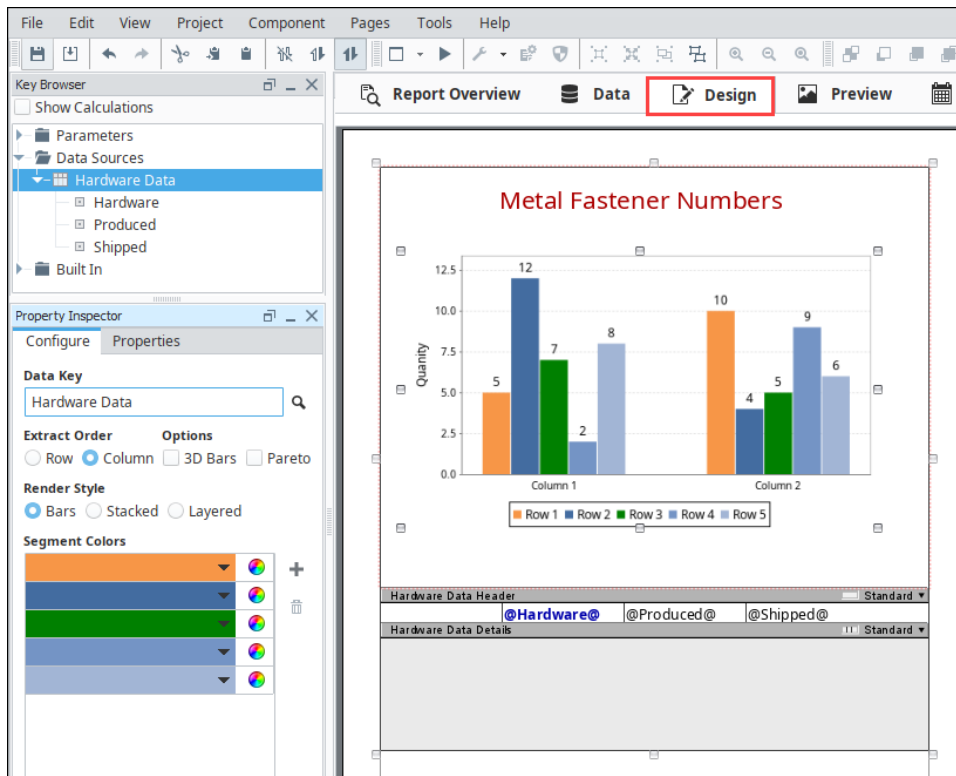
For a Bar Chart, Extract Order is the order in which the data is extracted from the data key, by Row or Column. By default, Row is selected. Use Row when columns in the data key define the series of data. Use Columns when rows in the data key define the series of data; each column is a new value for the same series.



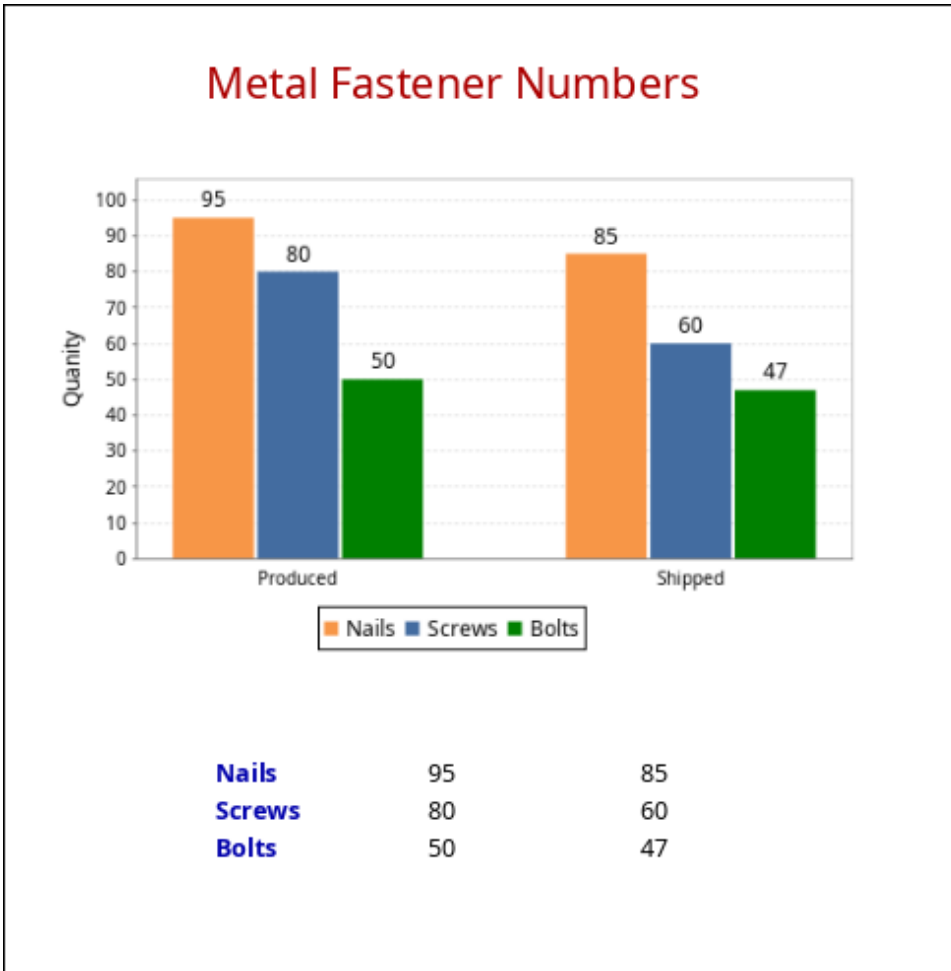
9. Now, let's put your creativity to work and make your report stand out by changing properties. This example changed the following properties:

- In the **Configure Tab** - change the third bar color to green.
- Select **Metal Fastener Numbers Report** header - change the Font Size to 24 pixels, the Text Color to red, and the Horizontal Alignment to Center in the Properties tab.
- Select the **Bar Chart** - enter a name for the Axis Label (i.e., Quantity), and set the Bar Labels to 'true' in the Properties Tab.
- Select each **Data Details cells** (i.e., Hardware, Produced, and Shipped), change the Font Size to 14 pixels, and the Hardware Text to blue in the Properties tab.

The Design panel will update the displayed sample data after changing the properties above.



10. Go to the **Preview** panel to view your report.



Nails	95	85
Screws	80	60
Bolts	50	47

Adding a Chart Inside a Details Row

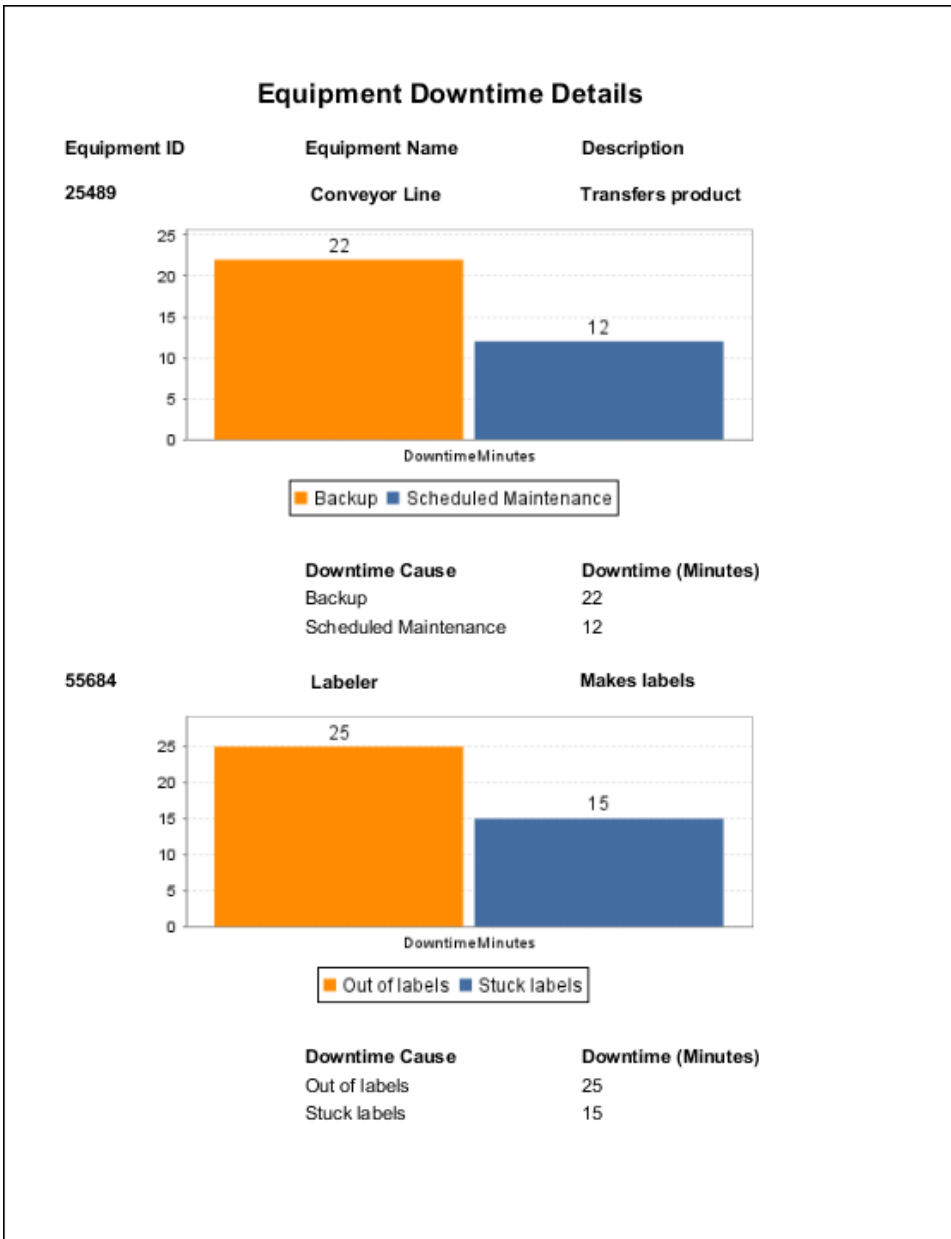
Adding a chart inside a Details row, duplicates the chart for the number of Detail rows you have in your table, and displays the respective data for each row in the chart. This example shows the Equipment Downtime Details for each piece of equipment: Cause and Downtime in Minutes. It also adds a Bar Chart after the Details row using an unstructured row, and a nested query so that each row has a chart with data from the query. Go to the [Nested Queries](#) page and complete the Equipment Downtime example at the bottom if you want to use the same datasets for this example.

1. Once you have your Data set up from the above example, click on the **Design** tab and add a table component to your report.
2. With the table created, add a Header row to the table. Drag all of your Equipment details into the table, and add header titles for each column.
3. Select the **Details** row, make the row **unstructured** , and drag the row down the page to make room for a chart.
4. Drag a chart from the Report Design Component Palette into the unstructured row and expand it.
5. With the chart selected, drag a **Datasource** (i.e., EquipDowntime) from the **Key Browser** to the **Data Key** field in the **Configure Tab**.

- If you are using a Bar Chart, set the **Extract Order** to **Column**.

The screenshot shows a report design tool interface. On the left, the 'Project Browser' shows a table named 'Equipment Header (Standard)'. Below it, the 'Table Browser' shows a hierarchy: Parameters, Data Sources, Equipment, EquipmentDescription, EquipmentID, EquipmentName, EquipmentDowntime, DowntimeCause, and DowntimeMinutes. The 'Property Inspector' is open for 'EquipmentDowntime', showing 'Data Key' as 'EquipmentDowntime', 'Grouping' as 'EquipmentDowntime', and 'Rows' checked for 'Header', 'Details', and 'Summary'. The 'Design' panel on the right shows a bar chart titled 'Equipment Downtime Details'. The chart has two columns, 'Column 1' and 'Column 2', and five rows of data. The y-axis represents 'Downtime (Minutes)' from 0.0 to 12.5. The legend indicates five rows: Row 1 (orange), Row 2 (dark blue), Row 3 (medium blue), Row 4 (light blue), and Row 5 (very light blue). Below the chart, the report structure is visible, including 'EquipmentDowntime Header', 'EquipmentDowntime Details', 'Equipment Details', and 'Equipment Summary'.

- Go to the **Preview** panel to view the report. If you want to make any changes to the chart like changing Segment Colors, adding Bar Labels, or adding a page break, etc., go back to the **Design** panel to modify any of the chart properties. When you're finished, return to the **Preview** panel. You'll notice that for each piece of equipment there is a chart and data. You can make each piece of equipment have its own page by setting the Page Break option on the Table under the Configure Table tab.



Using a Chart in a Table with Nested Queries

When working with data from nested queries, representing the sub-query data on a chart in a table row is fairly similar to using two unrelated data sources.

In this example, we used content from two different tables, as showing below

The Tables

Equipment

This table contains identifiers and descriptions for multiple pieces of equipment. A query named "equipment_list" is used by the report to retrieve this information.

equipment_id	equipment_description	equipment_name
1	North Tank	Tank 101
2	South Tank	Tank 202

Equipment_Downtime

This table identifies different downtime events, and the id for the equipment that went down. A query named "downtime_list" is used by the report to retrieve this information.

downtime_id	equipment_id	downtime_cause	downtime_minutes
1	1	Tank full	30
2	1	Pump failure	14
3	2	Changeover	60
4	2	Pump failure	34
5	2	Tank full	10

Queries

The queries used by this example are in the expand panel below.

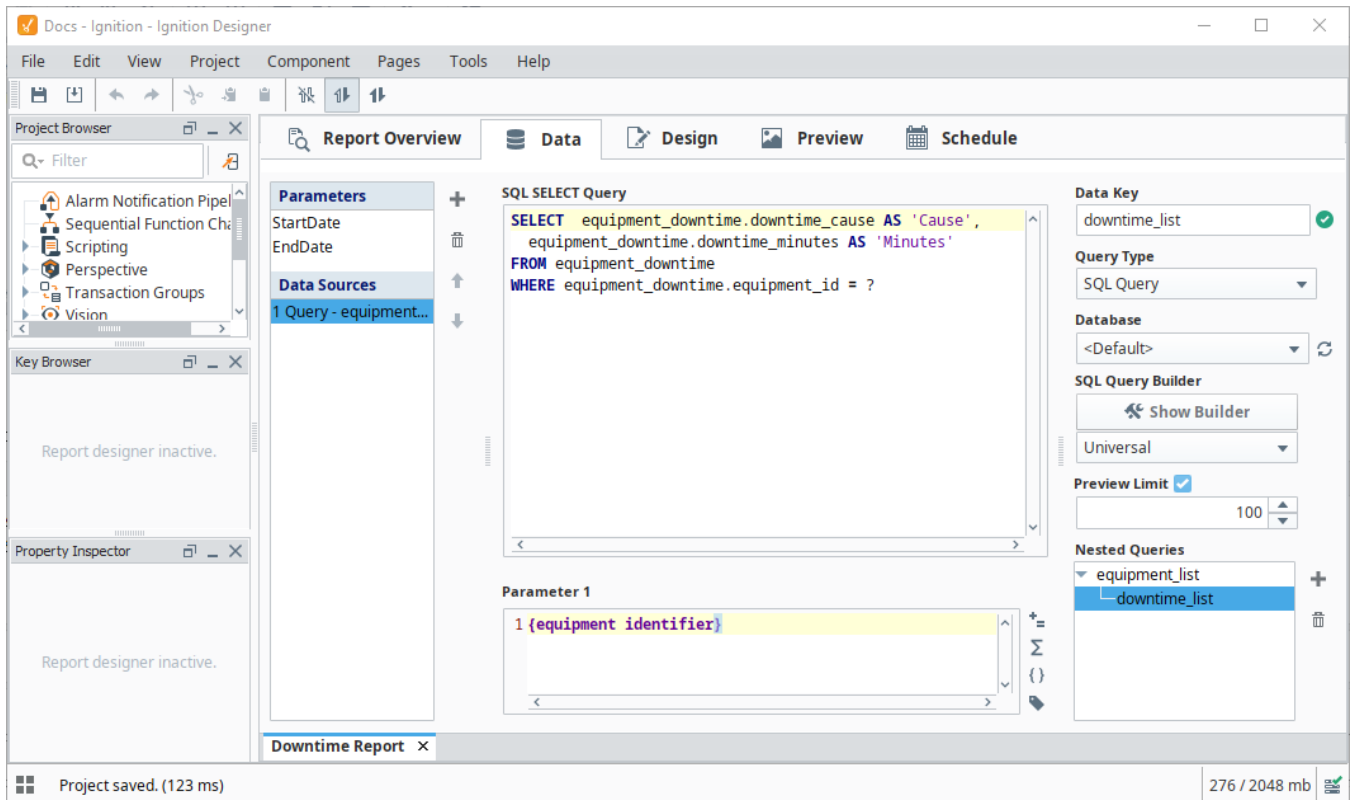
The query for the 'equipment' data source (an "SQL Query" type query) looks like the following:

```
SELECT equipment.equipment_id as 'equipment identifier',
       equipment.equipment_description,
       equipment.equipment_name
FROM equipment
```

The screenshot shows the Ignition Designer interface. The main window is titled "Docs - Ignition - Ignition Designer". The menu bar includes File, Edit, View, Project, Component, Pages, Tools, and Help. The toolbar contains various icons for file operations and navigation. The Project Browser on the left shows a tree view of the project structure, including Alarm Notification Pipelines, Sequential Function Charts, Scripting, Perspectives, Transaction Groups, and Visions. The Key Browser and Property Inspector are also visible on the left, both showing "Report designer inactive." The main workspace is divided into several panes. The "Report Overview" pane shows a list of "Parameters" (StartDate, EndDate) and "Data Sources" (1 Query - equipment...). The "Data" pane displays the SQL SELECT query for the "equipment" data source. The "Design" pane is currently empty. The "Preview" and "Schedule" panes are also visible. The right-hand pane contains configuration options for the query, including "Data Key" (equipment_list), "Query Type" (SQL Query), "Database" (<Default>), "SQL Query Builder" (Show Builder), "Preview Limit" (100), and "Nested Queries" (equipment_list, downtime_list). The bottom status bar shows "Project saved. (123 ms)" and "238 / 2048 mb".

The query for the "equipment_downtime" table (also an "SQL Query" type query) looks like the following:

```
SELECT equipment_downtime.downtime_cause AS 'Cause',
       equipment_downtime.downtime_minutes AS 'Minutes'
FROM equipment_downtime
WHERE equipment_downtime.equipment_id = ?
```

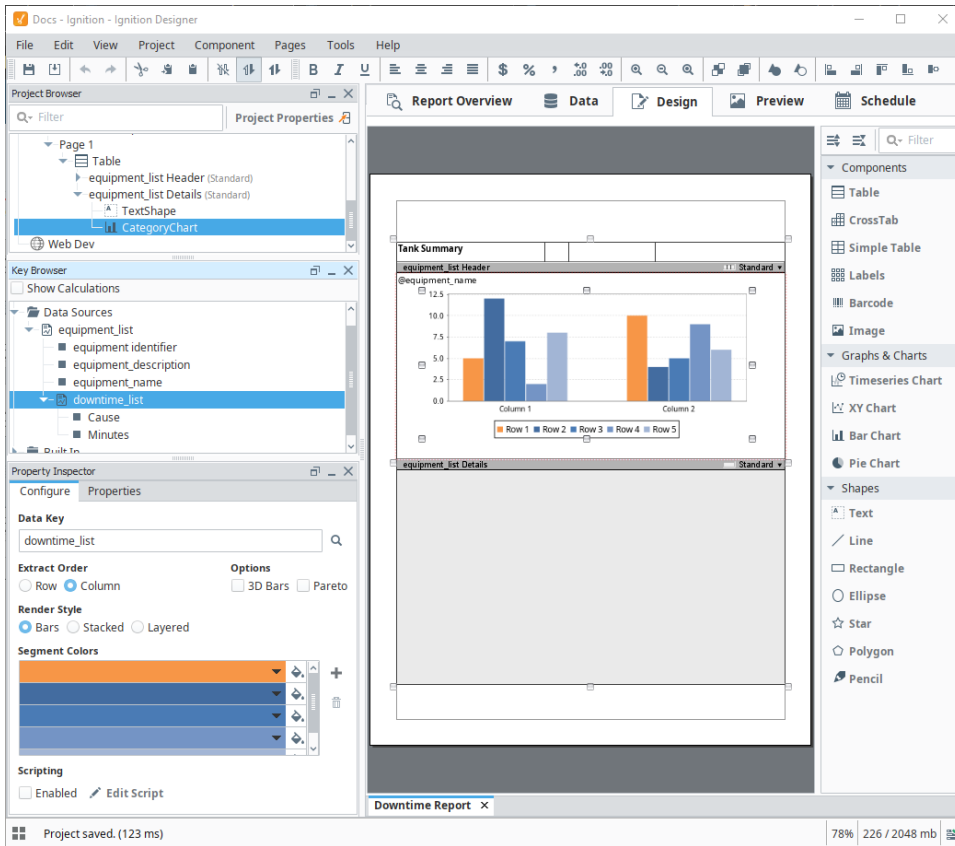



Configuring the Chart

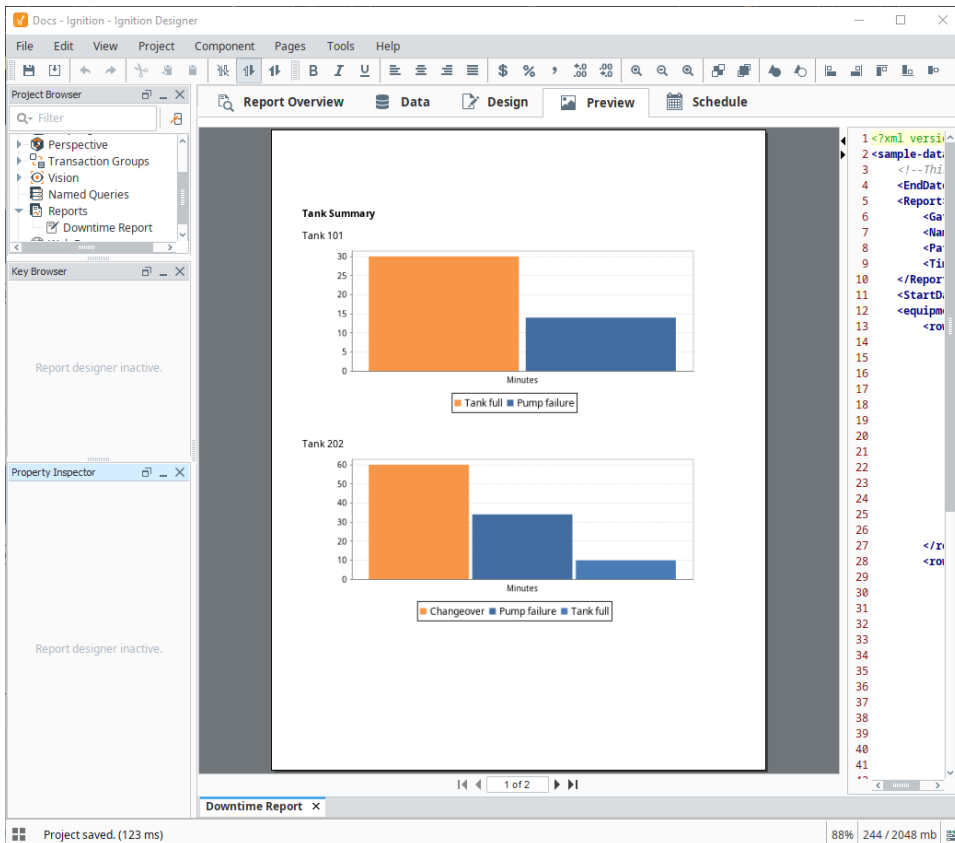
In this section, we are going to have a Category Chart component show the results of our downtime_list query. Because we're using a table, and a nested query, each row on the table will represent a different piece of equipment from our equipment table.

For this to work, the table component must be assigned the results of the parent query (which would be equipment_list in this case), and then assign downtime_list to our chart.

1. Create a new table component, and assign "equipment_list" as the **Data Key** for the table.
2. Make the Details row **unstructured**.
3. Add the equipment_name key to the details row, so we know which piece of equipment is represented by the row.
4. Add a **Category Chart** component to the details row of the table.
5. Set the **Data Key** on the chart to "downtime_list."
6. Set the **Extract Order** of the chart to "Column."



7. Switch to **Preview** panel, you should see each both pieces of equipment, represented as different charts, each showing their totaled downtime reasons.



Related Topics ...

- [Grouping Data Inside of Tables](#)
- [Nested Queries](#)
- [Table Rows](#)

Grouping Data Inside of Tables

Another important feature of tables is the ability to separate a single dataset into different categories or groups. When using a table in a report, you can group data in the table by a specific column. With Dataset Grouping, you can break tables down by data keys that share a common value (i.e., if you have a table that shows addresses, you can group the rows by the city, state, zip code, or any combination of the columns). This is done by dragging and dropping any of your data keys from the Key Browser to the Grouping list under the Configure Table tab.

When you add a data key to the Grouping list, a corresponding Details row will be added to your table component. Using dataset grouping allows you use data keys to organize and arrange your data into different categories, organizing the results based on the values of a key. Each group can have its own [Header, Details, and Summary rows](#). Additionally, the keys from [Show Calculations](#) and other [keychain functions](#) are supported for any level of grouping.

Note: Table Groups and Grouping Data in Tables are two completely different things despite having similar names. Table Grouping involves using multiple datasets in the same Table component while Grouping Data Inside of Tables (this page) sorts the rows inside a single dataset.

Demonstration

Assuming an initial table that looks like the following:

Type	Count
Type 1	100
Type 2	45
Type 2	450
Type 4	123
Type 3	50
Type 1	250
Type 3	871
Type 2	984


Type	Count
Type 1	100
Type 2	45
Type 2	450
Type 4	123
Type 3	50
Type 1	250
Type 3	871
Type 2	984

We could utilize Dataset Grouping to group the results in the table by unique "Type" values. By adding a grouping on the Type column, and some additional formatting, we can produce a table that looks like the following:

On this page ...

- [Demonstration](#)

[Grouping Data Inside of a Table Example](#)
[Separating Groupings using Page Breaks](#)



Dataset Grouping

[Watch the Video](#)

Type 1	Count
	100
	250
Type 2	Count
	45
	450
	984
Type 3	Count
	50
	871
Type 4	Count
	123

Type 1	Count		
	100 250	Type 2	Count
	45 450 984	Type 3	Count
Type 4	Count		50 871
	123		

Notice that we're no longer listing each type individually. Instead, the type acts as a sub-header for each group of data. See the example below for a how-to.

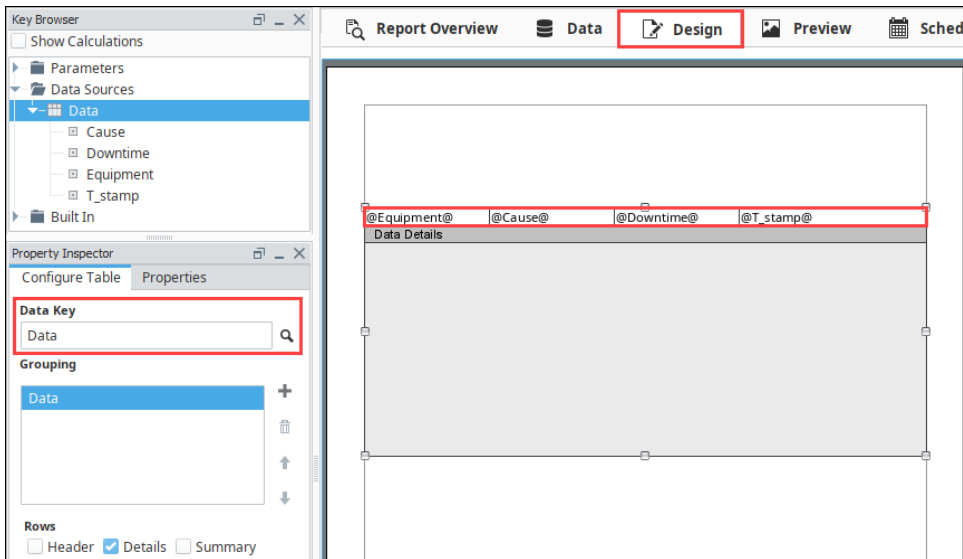
Grouping Data Inside of a Table Example

This example begins with a table similar to the one created in the [Report Workflow Tutorial](#). This example will demonstrate how to group the existing downtime report by equipment, collect downtime totals, and introduce some formatting techniques.

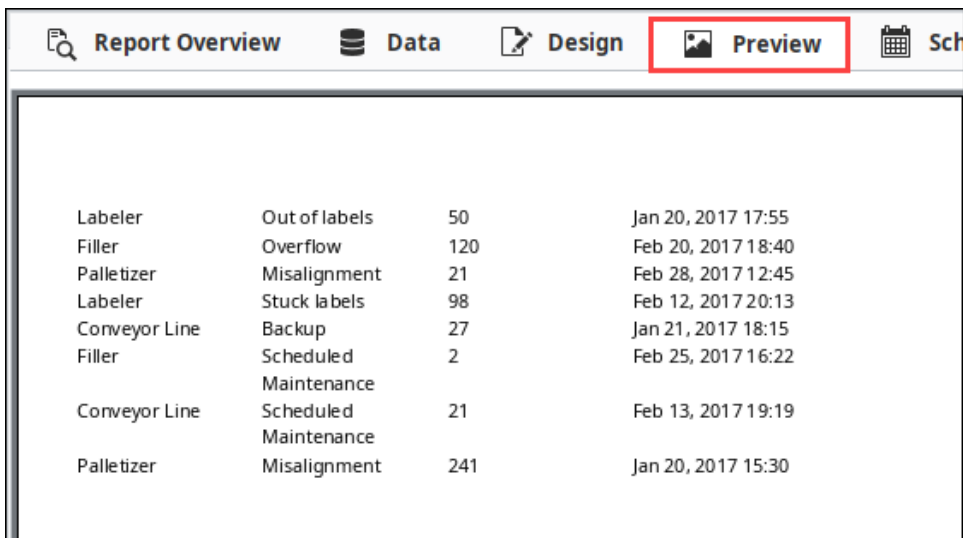
1. In the **Data** panel, create a Data Source that has a Timestamp, Equipment, Downtime, and Cause. Shown below is the text needed for a **Static CSV** datasource used for this example.


Data Source for Equipment Downtime Report
<pre>T_stamp, Equipment, Downtime, Cause "Jan 20, 2017 17:55", "Labeler", 50, "Out of labels" "Feb 20, 2017 18:40", "Filler", 120, "Overflow" "Feb 28, 2017 12:45", "Palletizer", 21, "Misalignment" "Feb 12, 2017 20:13", "Labeler", 98, "Stuck labels" "Jan 21, 2017 18:15", "Conveyor Line", 27, "Backup" "Feb 25, 2017 16:22", "Filler", 2, "Scheduled Maintenance" "Feb 13, 2017 19:19", "Conveyor Line", 21, "Scheduled Maintenance" "Jan 20, 2017 15:30", "Palletizer", 241, "Misalignment"</pre>

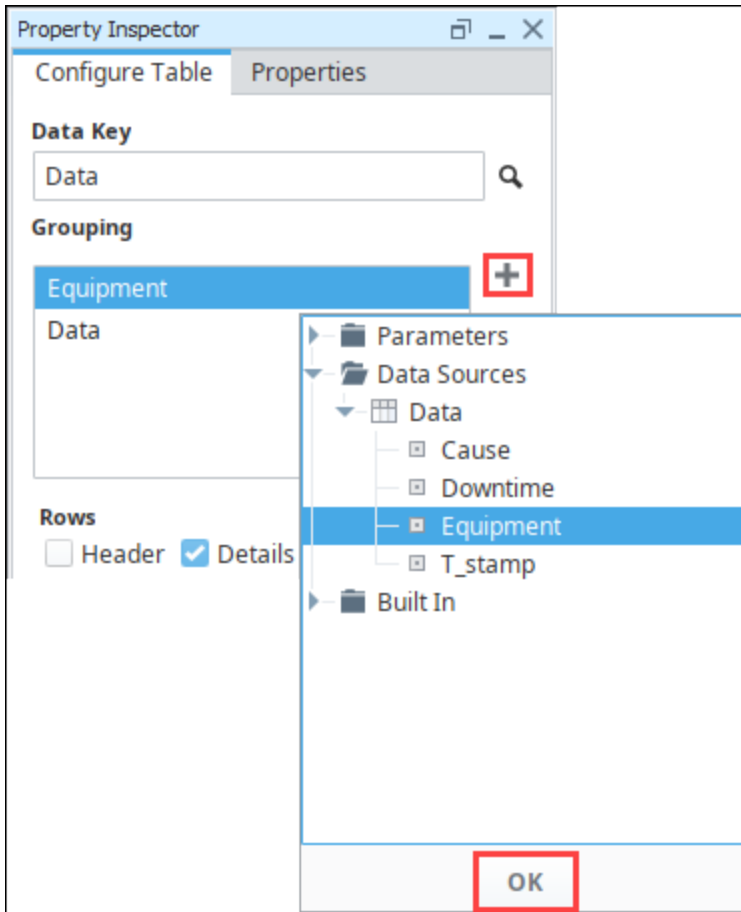
2. In the **Design** panel, drag a **Table** component to your report.
3. With the Table selected, drag the **Data** data source to the **Data Key** under the **Configure Table** tab of the Property Inspector.
4. Drag the each of the **data keys** (i.e., Equipment, Cause, Downtime, and T-Stamp) to any of the columns in the table row.



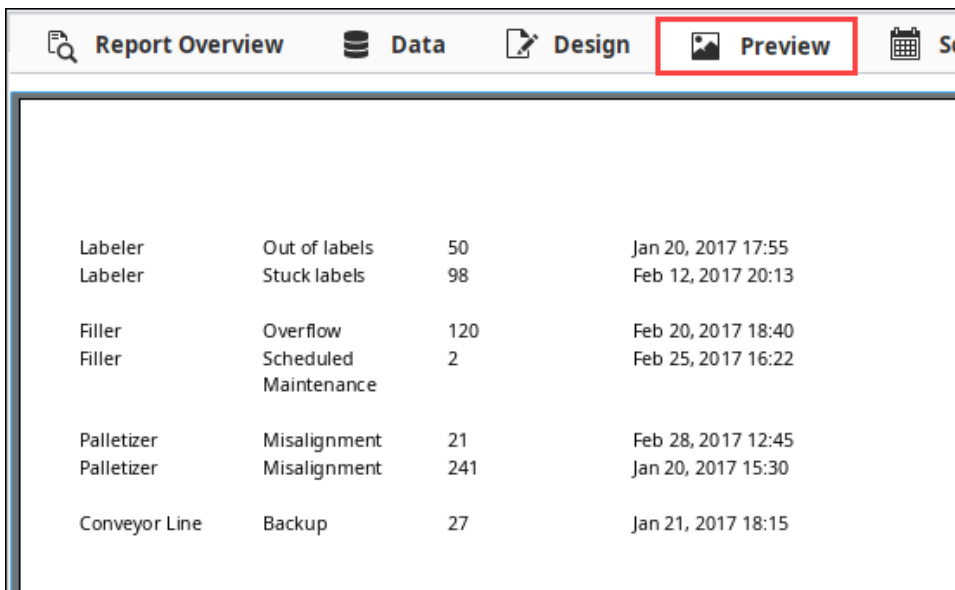
- Click on the **Preview** panel to see what the report looks like. You'll notice that all your data is there, but it's a little hard to read because it's not organized.



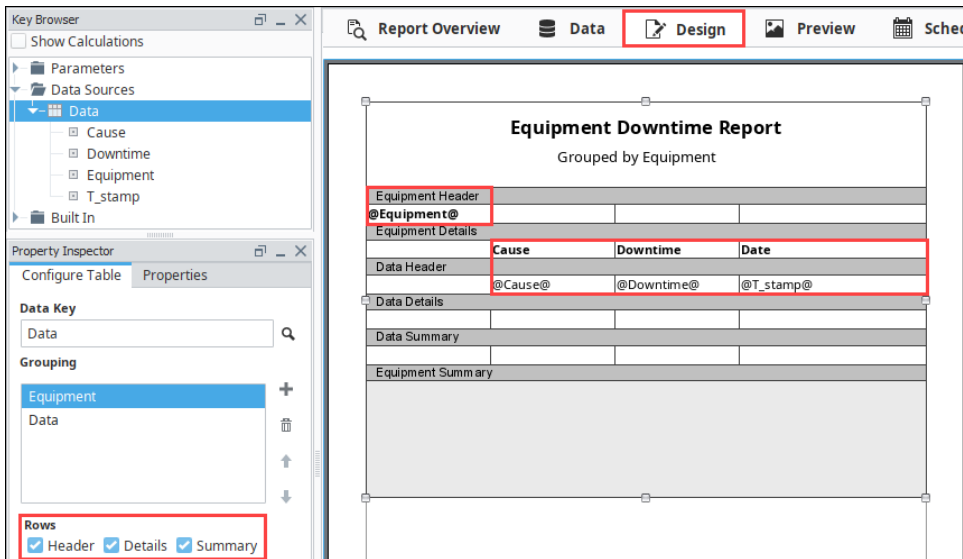
- Let's sort each row in the table by equipment and group all the equipment together. Go to the Property Editor, and in the **Configure Table**, click the **Add**  icon next to **Grouping**, and a window will open with a list of data keys. Select the **'Equipment'** row, and click **OK**. You'll notice 'Equipment' was added to the Grouping list, and an Equipment Details row was immediately added to the table.



7. Go to the **Preview** panel to see that the report is now sorting by equipment name.



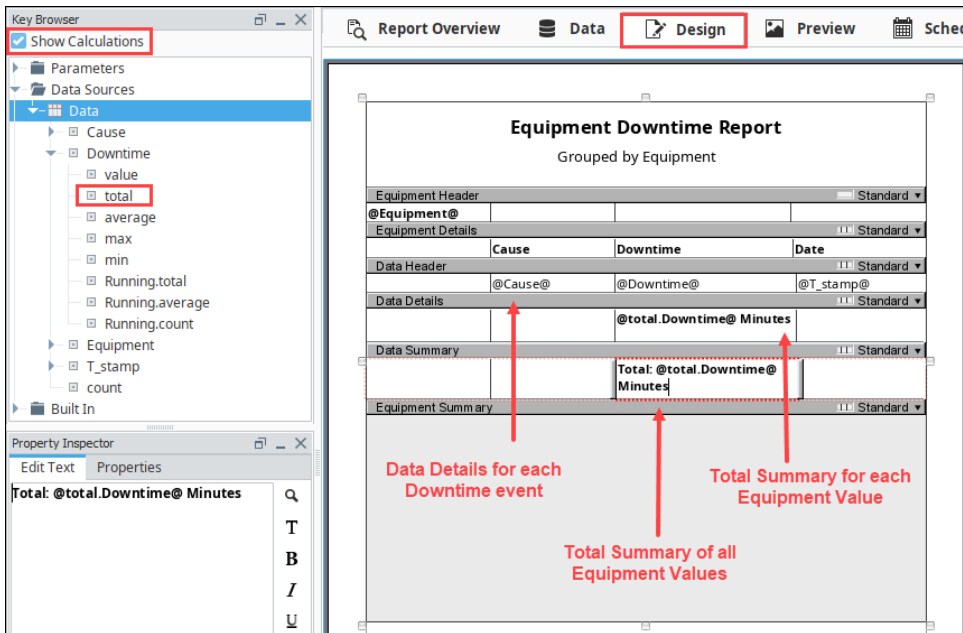
8. Let's remove the Equipment Name from each row in the table and add a Header. Go to the **Design** panel and cut '@Equipment@' from the Data Details row and paste it in the **Equipment Details** row. You can bold it to make it stand out.
9. While the table component is selected, go to the **Configure Table** tab in the Property Inspector and select the **Equipment** item in the grouping list and check both the **Header** and **Summary** boxes. Then select the **Data** group and select the **Header** and **Summary** checkboxes for it.
10. Next, make the Equipment header an **unstructured row** and add text as a title for your report using the **Text Shape** in the component palette. Unstructuring the row allows you to easily center the title of your report.
11. Now add header text for each of the Data Details columns (Cause, Downtime, and Date) by typing into the **Data Header** row.



12. Set the **Show Calculations** checkbox at the top of the **Key Browser**. Drill down into the Downtime column and drag the '@total.downtime@' to both the **Data Summary** and **Equipment Summary** rows. You can also add any text outside of the @ symbols. In this example, we added 'minutes' after the total downtime in our cell to '@total.downtime@ minutes'.
13. Lastly, do the same in the Equipment Summary row. We also added the word "Total" to the beginning of the cell: **Total: @total.downtime@ minutes**.



In any **Summary** row, '@total.Downtime@' is a sum of all downtime at that level of grouping; (i.e., in the **Data Summary** row it is the total downtime grouped by equipment. In the **Equipment Summary** row, '@total.Downtime@' is the sum of all downtime for all equipment groupings).



14. Click over to the **Preview** panel to check out your report. If you want to make any changes, go back to the **Design** panel to update your report. Notice that the **'total'** data key respects both groupings.

Equipment Downtime Report

Grouped by Equipment

Labeler

Cause	Downtime	Date
Out of labels	50	Jan 20, 2017 17:55
Stuck labels	98	Feb 12, 2017 20:13
	148 Minutes	

Filler

Cause	Downtime	Date
Overflow	120	Feb 20, 2017 18:40
Scheduled Maintenance	2	Feb 25, 2017 16:22
	122 Minutes	

Palletizer

Cause	Downtime	Date
Misalignment	21	Feb 28, 2017 12:45
Misalignment	241	Jan 20, 2017 15:30
	262 Minutes	

Conveyor Line

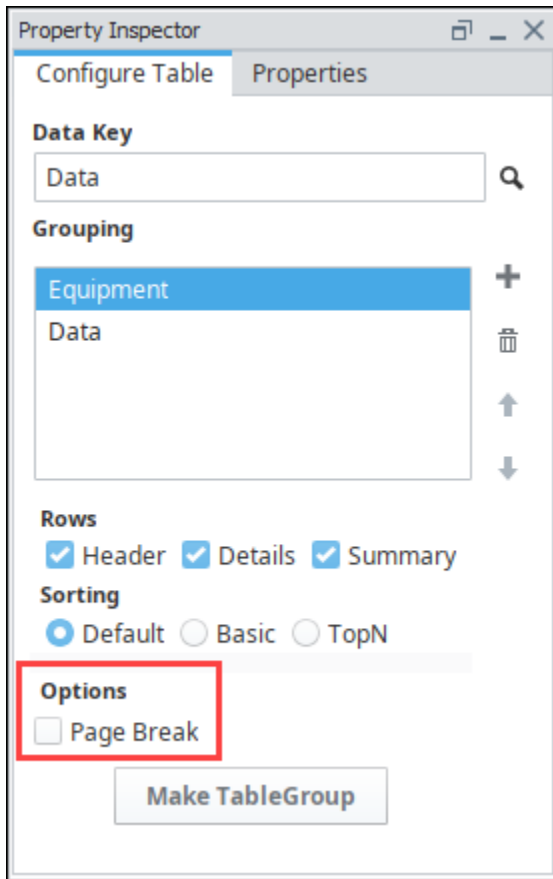
Cause	Downtime	Date
Backup	27	Jan 21, 2017 18:15
Scheduled Maintenance	21	Feb 13, 2017 19:19
	48 Minutes	

Total: 580 Minutes

Separating Groupings using Page Breaks

In the **Configure Table** tab of the **Property Inspector**, there is **Page Break** option that can be set to create breaks between each Grouping. Each new instance of that level of grouping creates a new page in the report. In the example above, we could add a page break in-between each grouping of equipment type, which would further delineate each grouping of data. This is especially useful if you are [adding charts](#) or other images at the

beginning of each group.



Related Topics ...

- [Table Groups](#)

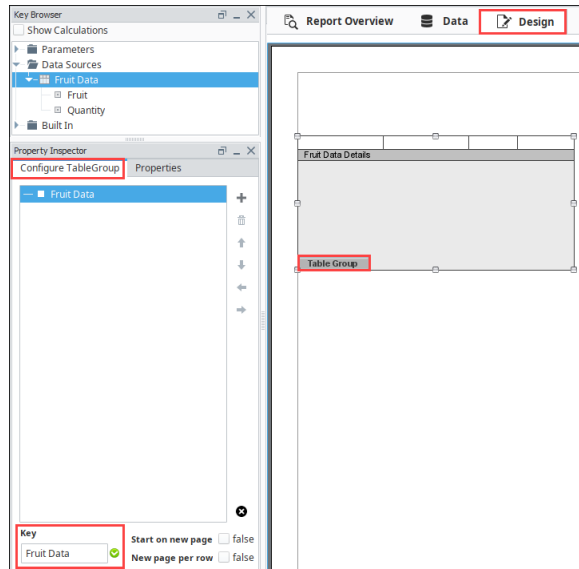
Table Groups

Table groups are a single component that contains many tables, each with its own unique data. There are two types of Table Groups you can create within a Table component: [Peer Tables](#) and [Child Tables](#). Peer Table Groups allow the second table to begin exactly where the first table ends. If multiple tables are used, they can be configured as multiple page templates, providing a page break between tables. Child Table Groups allow you to nest one table inside another, and work especially well with any [Nested Queries](#). What's really nice about Table Groups and Nested Data Sources is that you can create Summary Tables for categories of items or drill-down charts all in one report.

Note: Table Groups and Grouping Data in Tables are two completely different things despite having similar names. Table Grouping (this page) involves using multiple datasets in the same Table component while Grouping Data Inside of Tables sorts the rows inside a single dataset.

Making a Table Group

To make a Table Group, you simply need to click the **Make TableGroup** button in the Configure Table tab of your table. You will be able to tell that it worked when you notice a **Table Group** icon displayed in the lower left corner of the table component, and your **Configure Table** tab has now changed into a **Configure TableGroup** tab.



We can see our original table listed in the **Configure TableGroup** tab, and we have the option to add additional tables by clicking the plus **+** icon. Adding a child table will add a table that is a child to the table that is currently selected in the Table Group list. Adding a peer table will add a table that is a peer to the currently selected table. We can add as many of each of these as we need.

For example, in the image below we added two child tables to the original Fruit Data table. We then have two peer tables to Fruit Data, one of them is a standalone table, while one has a child table with that child table also having a child table. More information on [child and peer tables](#) is included below.

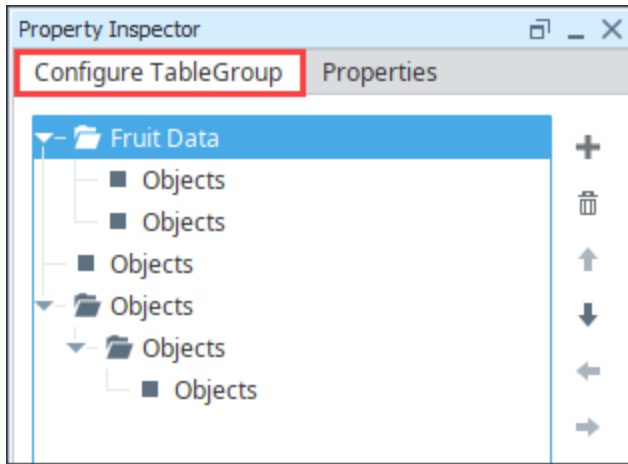
On this page ...

- [Making a Table Group](#)
 - [Navigating Within Table Groups](#)
 - [Child versus Peer](#)
- [Peer Tables](#)
- [Child Tables](#)
 - [Using Child Tables - An Example](#)



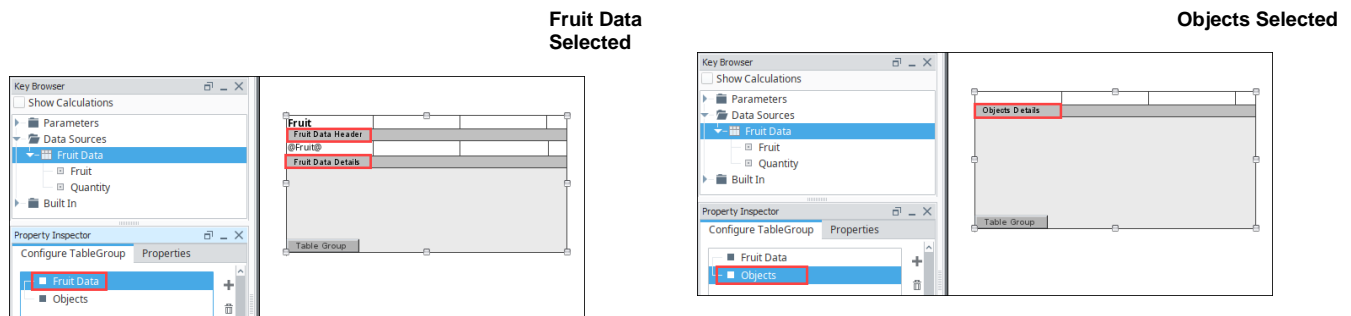
Table Groups


[Watch the Video](#)



As you can see from the image above, the Table Group hierarchy can get quite confusing very quickly. It is important to understand how each part of the Table Group works, and to always be aware of which table you are actually looking at inside the Table Group. To view a different table within the Table Group, simply select it from the hierarchy. The table displayed in the design area will switch depending on which table you selected.


Navigating Within Table Groups



This allows you to get a quick preview of how each of the tables within the Table Group are set up. To go in and configure an individual table within the Table Group, select it from the hierarchy and then click on the table object in the design area. The Configure TableGroup tab in the Property Inspector should change to the Configure Table tab, and will display properties relevant to the table you had selected. This allows you to add a Data Key, set grouping, configure relevant row styles, and format the table in any way that you want. After configuring the table, we can navigate back to the Table Group configuration by clicking on the **Table Group**  icon in the bottom left corner of the table.

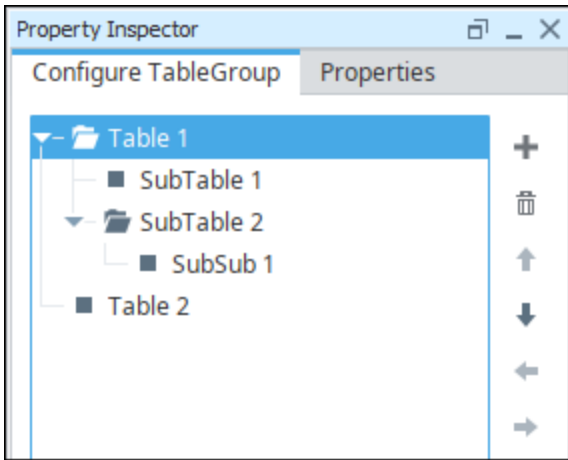
At the bottom of the Configure TableGroup tab, there is an option called **Start on new page**. With this option enabled, we can force the next table in the list to start on the next page instead of immediately after the previous table. This will only apply to the table that is currently selected.

In the Configure TableGroup tab, we can modify the hierarchy of the Table Group using the **Arrow**  icons to the right of the table list. This allows us to change the order that the tables appear in, change a peer table into a child table, or change a child table into a peer table. If at some point, you realize you have added too many tables, you can also use the **Delete**  icon to remove it. Be careful, as removing a table that has children will also remove the child tables.

Finally, if you ever want to cancel the Table Group click the **Delete**  icon to do so.

Child versus Peer

When adding tables to a Table Group, they can be added as either a "Child" or "Peer" table. These terms are always relative to adjacent tables in the group's hierarchy. Thus a table that is a child to one table may have some peers, as well as its own child tables. You can always tell the relationship between two tables by the indentation of each table.



Both Table 1 and Table 2 are peers. This is denoted by the matching indentation. In regard to the image above, the following statements are true:

- **SubTable 1** and **SubTable 2** are both children to **Table 1**, because they are indented over from their parent.
- **SubTable 1** and **SubTable 2** are also peers to each other, because they have a matching indentation.
- **SubSub 1** is a child to **SubTable 2**, and does not have any peer or children tables.

When the report is generated, the first row of **Table 1** will appear first. Each row of **Table 1** will generate an instance of **SubTable 1** and **SubTable 2**. Furthermore, each row of **SubTable 2** will generate an instance of **SubSub 1**. After all of **Table 1** has been represented on the report, **Table 2** will appear on the report.

The behavior of Peer and Child tables are further described below.

Peer Tables

Peer Tables are inserted one after another in the parent table. One table could potentially show details for each piece of equipment, with the peer table showing something completely unrelated starting at the end of the table before it. This helps keep the flow between two different datasources. Typically, each datasource would get its own table component. The issue with that is where to place them with how the tables automatically grow with more data. You could potentially have large amounts of blank space between the two tables because the first table only has a few records. This allows all the data to flow from one datasource to the next, even with unrelated datasources. You can see in the example below, the first image shows what a single individual table looks like, while the second image is a Table Group of two completely unrelated datasources stacked together. Each table inside the Table Group is also unique, with their own formatting, style, and even row versions.

Individual Datasource in Singular Table

Fruit	Quantity
Apples	60
Oranges	45
Melons	25

Table Group with Two Different Datasources

Fruit	Quantity
Apples	60
Oranges	45
Melons	25
Customer	Order Quantity
John	47
Mary	82
Tom	14
Peter	31

Child Tables

Child table groupings allow you to nest one table inside of another table. One table could potentially show details for each piece of equipment, but with a child table under each row, you can show all of the relevant downtime occurrences for that piece of equipment. These types of tables work well with [Nested Queries](#).

Using Child Tables - An Example

Creating child tables requires having a complex dataset. Go to the [Nested Queries](#) page and complete the Equipment Downtime example at the bottom if you want to use the same datasets for this example.

1. Once you have your Data set up, click on the Design tab and add a **Table** component to your report.
2. With the table created, add a **Header** row to the table. Drag all of your Equipment details into the table, and add header titles for each column. Equipment table column headers for this example:
Equipment ID
Equipment Name

Description

The screenshot shows a report design tool interface. The main window is titled 'Report Overview' and has tabs for 'Data', 'Design', and 'Preview'. The 'Design' tab is active. The main area displays a table titled 'Equipment Downtime Details' with the following structure:

Equipment ID	Equipment Name	Description
@EquipmentID@	@EquipmentName@	@EquipmentDescription@

The 'Equipment ID' column is highlighted with a red box. On the left, the 'Key Browser' shows a tree view with 'Equipment' selected, and the 'Property Inspector' shows the 'Equipment' table selected. The 'Table Group' button is visible in the lower left of the table component.

3. Now that we have a base table, click the **Make TableGroup** button at the bottom of the Property Inspector.
4. Once you make your TableGroup, you're ready to create a Child Table. In the **ConfigureTable Group** tab, make sure the Equipment table is selected and click the plus icon **+** and select '**Add child table.**' The table you create becomes a child of the Equipment table. By default, the child table will be called **Objects**.
5. A Child Table has its own datasource and uses a nested query to pull data from the database. Double click on the child table to select it and see the details for that table. **Note:** You can switch back to the Configure TableGroup tab by clicking on the Table Group button **Table Group** in the lower left of the Table component.
6. Drag the child datasource (i.e., EquipDowntime) from the Key Browser to the **Key** field in the Configure TableGroup tab.
7. Next, add a header row with column name text:
Downtime Cause
Downtime (Minutes)
8. Drag your columns from the Key Browser to the appropriate columns in your table.

Key Browser

- Show Calculations
- Parameters
- Data Sources
 - Equipment
 - EquipmentDescription
 - EquipmentID
 - EquipmentName
 - EquipDowntime
 - DowntimeCause
 - DowntimeMinutes
 - Built In

Property Inspector

Configure TableGroup Properties

- Equipment
 - EquipDowntime

Key

EquipDowntime

Start on new page false

New page per row false

Report Overview Data Design Preview Sch

Equipment DD Equipment DD2 x

9. Go to the **Preview** panel to view the report. You'll notice the Child Table is now embedded in the parent table.

Equipment Downtime Details

Equipment ID	Equipment Name	Description
25489	Conveyor Line	Transfers product
	Downtime Cause	Downtime (Minutes)
	Backup	22
	Scheduled Maintenance	12
55684	Labeler	Makes labels
	Downtime Cause	Downtime (Minutes)
	Out of labels	25
	Stuck labels	15
88456	Palletizer	Makes pallets
	Downtime Cause	Downtime (Minutes)
	Misalignment	38
	Misalignment	55
626145	Filler	Fills tanks
	Downtime Cause	Downtime (Minutes)
	Overflow	50
	Scheduled Maintenance	40

Related Topics ...

- [CrossTab and Simple Tables](#)
- [Nested Queries](#)

Images in Reports

Images in Reports

You can spice up your reports by adding images, whether it's your corporate logo, or embedding icons in tables to help users find data quickly. Images can be added to reports in several different ways:

- Drag and drop an image from your desktop/computer to the report - No data key is required.
- Drag an image from the Image Management Tool to a report - No data key is required.
- Drag an Image component from the Report Design Palette to a report. This option requires that you enter a path from wherever the image is located (i.e., local drive, shared drive, or webpage) to the image in the Key property. You can either manually type in the path or reference a parameter, thus, making it static or dynamic.

[Data Keys](#) can help you make the images in reports more dynamic. By dragging a parameter or a datasource column from the Key Browser into the Key property of the Property Inspector, a data Key can resolve to a byte array, such as an image retrieved from a database, or a URL that points to an image on a webpage, or an image file stored on a shared drive. Reports are generated on the Gateway so the image source needs to be accessible from the Gateway computer.

[Keychain Expressions](#) can not be used in the data Key, but the Key can be a parameter or Data Source that dynamically constructs a path to an image. This allows you to easily change images in reports without changing the actual image component.

Caution: Whether using a parameter or typing in a path name in the Key property, always use double quotes around the path name.

Before we dive-in to show you how to configure the paths for your images, please read the Notes contained in this section carefully because path names require a specific format. There are also a few helpful hints about configuring images in reports.

Image Formats

The Report Module supports the following image file formats:

- .gif
- .tiff
- .jpeg
- .jpg
- .png
- .bmp
- .pdf

Drag and Drop Images

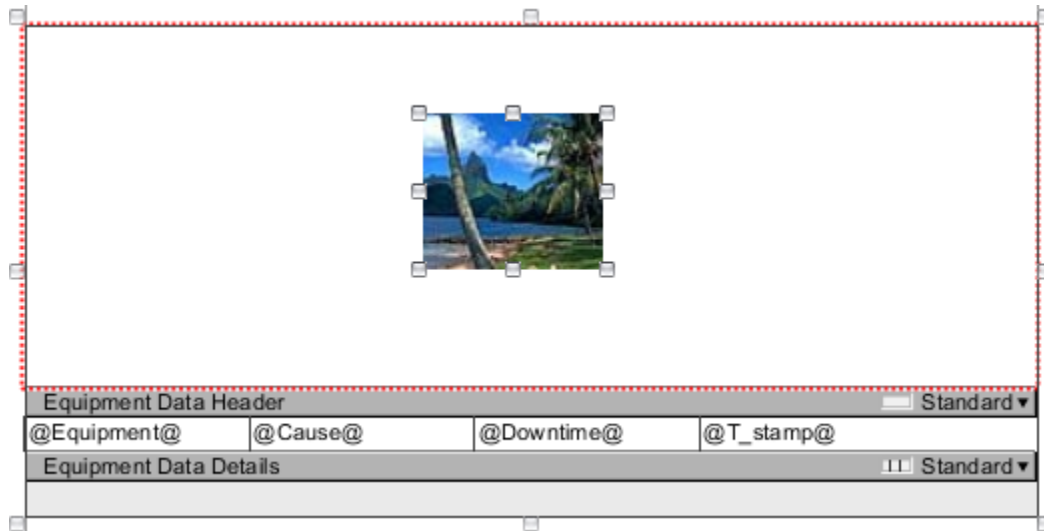
You can use the drag and drop feature to drag images from a local computer, shared drive, webpage, or the Image Management Tool to add images on a report. The report will display blue highlights around your report when you drop your image onto the report. No data key is required, and the dropped image will be used. Below, you can see a table header highlighted to drop an image in.

Equipment Data Header			
@Equipment@	@Cause@	@Downtime@	@T_stamp@
Equipment Data Details			

On this page ...

- [Images in Reports](#)
 - [Image Formats](#)
- [Drag and Drop Images](#)
 - [Drag and Drop an Image from a Desktop](#)
 - [The Image Management Tool](#)
- [The Key Property](#)
 - [Static File Path](#)
 - [Dynamic Key Property](#)

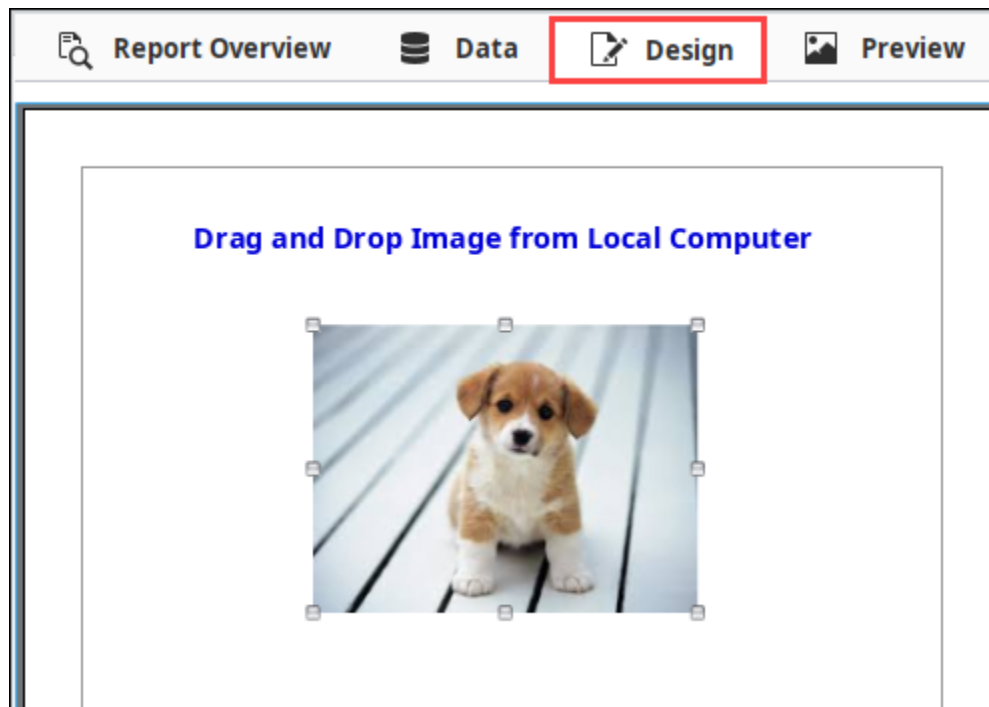
If you associate a key with an image component, the original placeholder image (as shown below) will still be seen in the Design panel, but will be replaced with the image associated with the data Key in the Preview panel. If the key does not link to a valid image, no image will be shown in the Preview panel.



Drag and Drop an Image from a Desktop

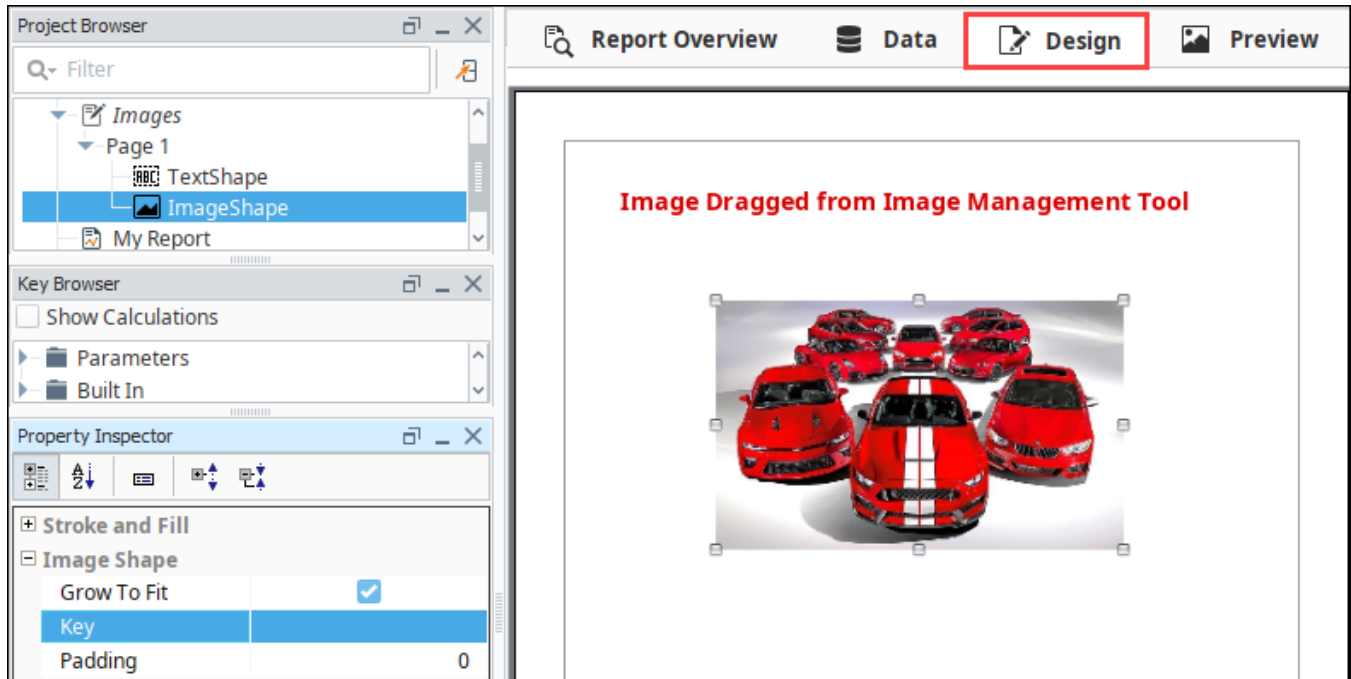
If you have a collection of images stored on your local computer, you can easily drag and drop any image into a report. No Key is required. You'll notice the **Key** property is blank. This doesn't mean you can't associate a Key with an image component. You most definitely can! The image will be replaced with the image associated with the Key when the report is generated. If you're going to use images from a local computer, it's a good idea to import those images to the Image Management Tool to make them visible and available to all project users.

To drag and drop images into a report from your desktop, head to the **Design Panel**. Find an image on your local computer, and drag the image directly onto the report. A new Image component will be created, and will be filled with your selected image. Note that the resulting Image component's Key property will be blank. This is expected behavior, as the image is embedded directly on the page, and not dependent on a key or file path.



The Image Management Tool

Images may also be dragged and dropped directly from the [Image Management Tool](#). Go to your report and open the **Design** panel. Drag an image from the Image Management tool to your report. You'll notice that the Key property is blank. Your image will be displayed directly in both the Design and the Preview panels.



The Key Property

Using the Image component is one way of adding an image in a report. You can make the image path static or dynamic by entering the path name manually, or referencing a parameter. The following sections describe how to set up the image path to use both static and dynamic data keys.

Note:

- Python is used to handle expressions when creating new parameters. Because the backslash has a special meaning in Python, formatting a path name for local drives and shared drives is a little different. Each backslash must be represented by two backslashes, for example, place four backslashes at the beginning of a network path and two backslashes between folders in the path. Here are a few examples;
 - Accessing a local drive - "\\C:\\Images\\veggies.jpg"
 - Accessing a shared drive - "\\ComputerName\Folder\Images\image.png"
- URLs from a webpage, use a forward slash, so use the link as is - "<http://i.stack.imgur.com/WCveg.jpg>"

Static File Path

A static file path is created by dragging an Image component from the Report Design Palette into the Design panel, and manually typing the image path name into the Key property. The image path can point to a drive on a user's local machine, shared drive, or webpage. In this example, the image path is pointing to a user's local machine.

Note: To paste a link into the Key property, use the Ctrl V command. Right clicking with your mouse to paste a link in the Key property is not an available function.

In the **Key** property, enter the path name where the image is located. You can type it in manually, or paste (**Ctrl-V**) the path name in the Key property making sure you have double quotes around the path name. **Note:** The path name uses double backslashes separating the drive and folders (i.e., "**C:\\Images\\Images\\veggies.jpg**").

The placeholder image will be seen in the Design panel, and the image associated with the Key property will be displayed in the Preview panel or when the report is executed.

Key Browser

- Show Calculations
- Parameters
- Data Sources
- Built In

Property Inspector

Stroke and Fill

Fill

Fill Color

Opacity 1

Stroke Style Hidden

Stroke

Image Shape

Grow To Fit

Key "C:\Images\veggies.jpg"

Padding 0

Page Index 0


Preserve Aspect Ratio

Rectangle Shape

Basic Properties

Report Overview Data Design Preview Sched


Placeholder Image



	Vegetable	Qty	
Vegetables Header			
	@Vegetable	@QTY@	
Vegetables Details			

Go to the **Preview** panel and you'll see the placeholder image was replaced by the image associated with the Key property.

Image Associated with Key Property



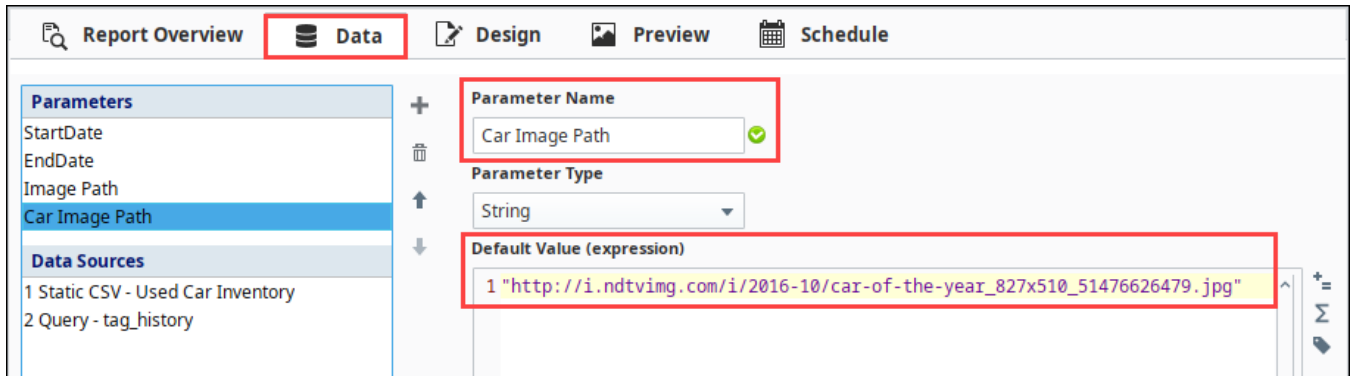
Vegetable	Qty
Squash	10
Zucchini	18
Carrots	25
Potatoes	8
Beans	33

Dynamic Key Property

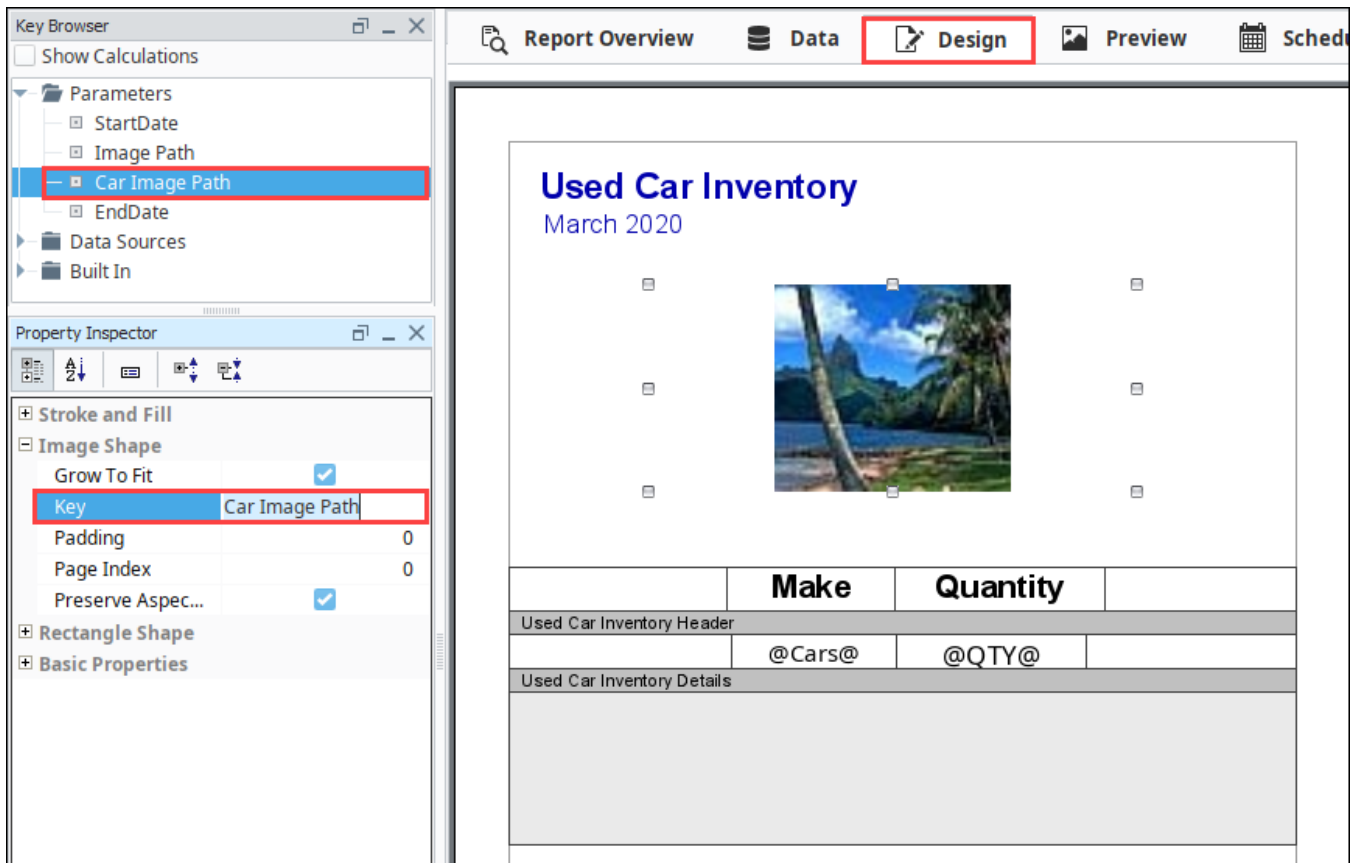
The Key property can be a parameter that dynamically constructs a path to an image which allows you to easily change images in reports without changing the actual image component. You can make an image path dynamic by dragging an Image component from the Report Design Palette into the Design panel, and creating a parameter for the image path. The expression for the Image Path parameter can be a path to a local drive, shared drive, or webpage.

Note, that reports **always** execute on the Gateway, so the file path is always relative to the Gateway.

To create a parameter, go to the **Data** panel, and click the plus **+** icon to add a parameter. In the expression block, enter in a link from a webpage using double quotes around the path name.



Go to the **Design** panel, click on the placeholder image, and expand the **Parameters** folder in the **Key Browser**. Drag your new parameter (i.e., Car Image Path) into the **Key** property of the **Property Inspector**. The image associated with the parameter in the Key property will be displayed in the **Preview** panel or when the report is executed.



Go to the **Preview Panel**, and you'll see the placeholder image was replaced by the image associated with the parameter in the Key property.

Used Car Inventory

March 2020



Make	Quantity
Ford	13
Toyota	22
Mercedes	2
Buick	6
Lexus	7
Jeep	11

Related Topics ...

- [Data Keys](#)
- [Keychain Expressions](#)

CrossTab and Simple Tables

In the other sections of the Report Table, we talked about creating tables using the Table component, commonly referred to as Standard Tables. Standard Tables focus on a dynamic number of rows with static columns, however, there are other table types that behave a little differently. These other table types are the Simple Table and CrossTab Table.

The CrossTab Table, although similar to standard tables and charts, are commonly used to summarize the relationship between two categories of data by showing summaries of cross sections of the datasource. The Simple Table allows you to easily create a grid-like table structure with the rows and columns being dynamic. Both table types are described below.



Working with Simple and CrossTab Tables

The examples on this page assume you already have datasources created. If not, you can copy the data from the Code Blocks in each example to create your own datasources to be used with the following Simple Table and CrossTab Table examples.

On this page ...

- [CrossTab Table](#)
- [Simple Table](#)

CrossTab Table

The Cross-Tabulation or CrossTab component is a tabular data element like the [Table](#) and [Charts](#). Also known as *Contingency Tables*, they are commonly used to summarize the relationship between two categories of data by showing summaries of cross sections of the data source. To be useful, crosstab data should have the following:

- Lots of repetitious data. Sums, Averages, and other Aggregating functions are well represented by CrossTab Tables.
- A data source that provides at least two columns of data which are repetitious compared to the number of rows.
- One or more columns that represent a value that requires calculation. Examples are: summing money, displaying average response times, counting occurrences, etc. These calculations may be provided as columns or calculations of the data source, or as [Keychain Expressions](#).

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

The objective of this CrossTab example is to show the number downtime time events by site and equipment. Let's give it a try!

1. In the **Data** panel, create a CSV data source named **Downtime_by_Site**. You can copy the data from the Code Block below to create your own CSV Data Source.

Downtime_By_Site

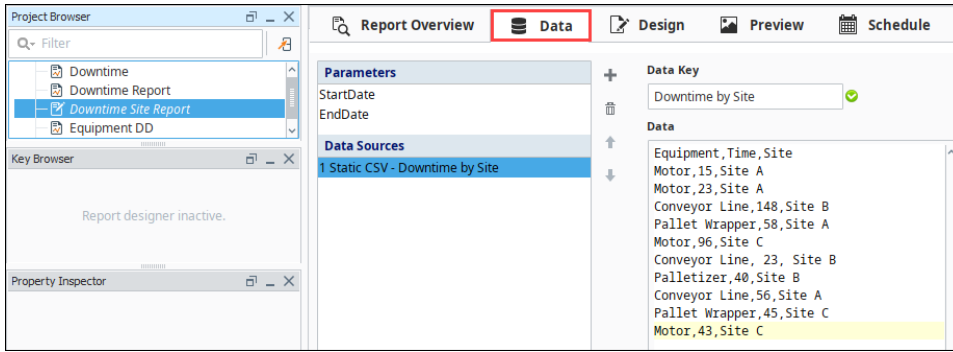
```
Equipment,Time,Site
Motor,15,Site A
Motor,23,Site A
Conveyor Line,148,Site B
Pallet Wrapper,58,Site A
Motor,96,Site C
Conveyor Line, 23, Site B
Palletizer,40,Site B
Conveyor Line,56,Site A
Pallet Wrapper,45,Site C
Motor,43,Site C
```


CSV Data Source

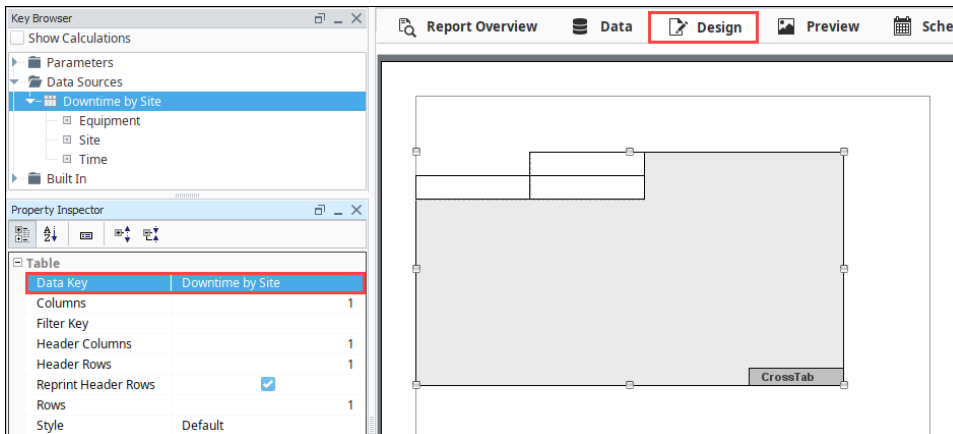


CrossTab Table

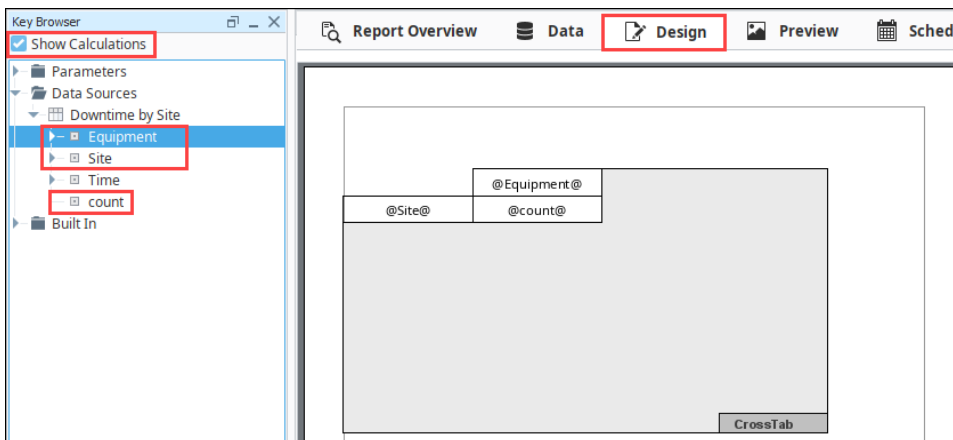
[Watch the Video](#)



- In the **Design** panel, drag a **CrossTab** component  to your report.
- With the CrossTab component still selected, drag-and-drop your Data Key (i.e., Downtime by Site) from the **Key Browser** to the **Data Key** property in the **Property Inspector**.



- In the **Key Browser**, set the **Show Calculations** property to **true**.
- Next, add some data keys to the cells in the CrossTab Table component. From the **Key Browser**, drag the '@Equipment@' to the top cell, '@Site@' to the leftmost cell, and '@count@' to the remaining cell. This will show the downtime count of each piece of equipment by site.



- Switch to the **Preview** panel and your CrossTab Table will display the results.

Report Overview Data Design **Preview** Schedule


CrossTab Table

	Motor	Conveyor Line	Pallet Wrapper	Palletizer
Site A	2	1	1	0
Site B	0	2	0	1
Site C	2	0	1	0

Simple Table

The Simple Table is a grid-like table structure that dynamically creates new rows and columns for rows returned by the Data Keys on the component. With the Simple Table you can very quickly add a table inside a report.

The objective of this Simple Table example is to show each piece of equipment including the total amount of downtime, occurrences, and the average duration of each occurrence, broken down by each piece of equipment.



**INDUCTIVE
UNIVERSITY**

Simple Table

[Watch the Video](#)

1. In the **Data** panel, create a CSV data source named **equipment_downtime**. You can copy the data from the Code Block below to create your own CSV Data Source.

```
equipment_downtime

equipment,downtime
conveyor line,78
filler, 68
labeler,84
palletizer,27
```

CSV Data Source

Report Overview **Data** Design Preview Schedule

Parameters

- StartDate
- EndDate

Data Sources


- 1 Static CSV - equipment_downtime

Data Key

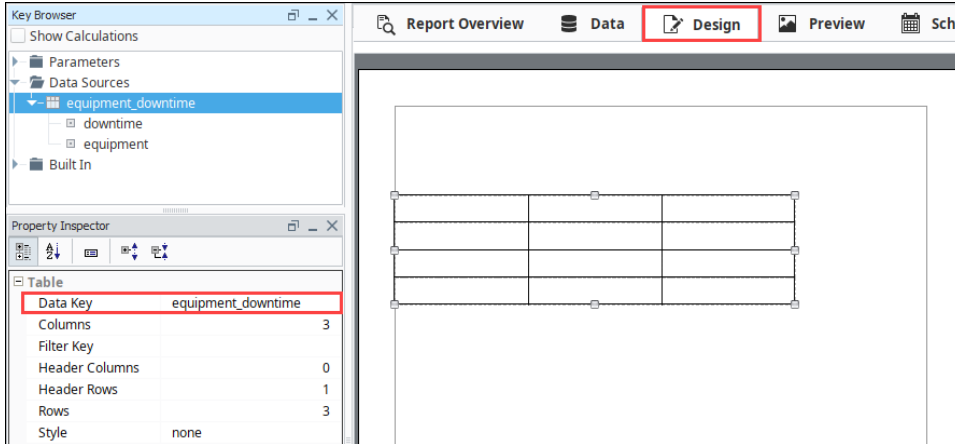
equipment_downtime ✓

Data

```
equipment,downtime
conveyor line,78
filler, 68
labeler,84
palletizer,27
```

2. In the **Design** panel, drag a **Simple Table** component  to your report.

- With the Simple Table component selected, drag and drop your Data key (i.e, equipment_downtime) from the **Key Browser** to the **Data Key** property of the **Property Inspector**.

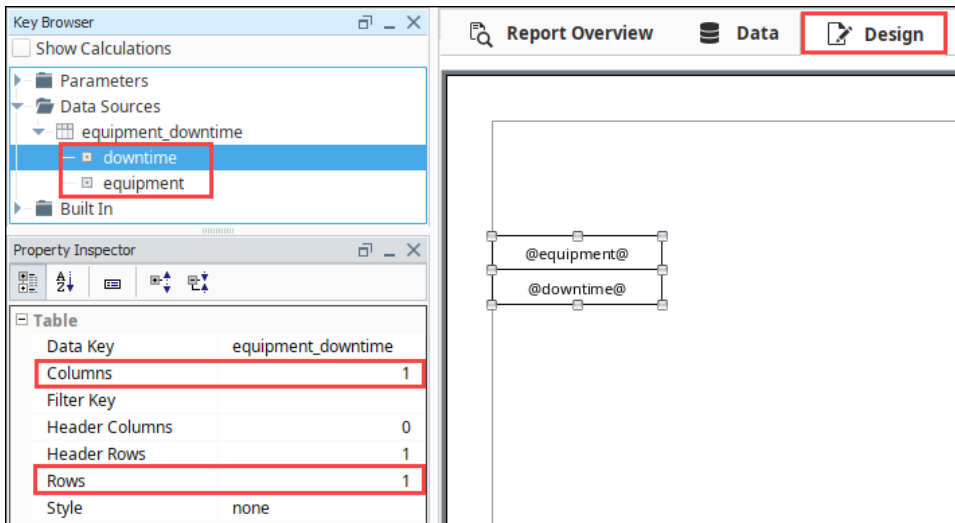


- From the **Key Browser**, drag the '@equipment@' key to the top row of the Simple Table, and the '@downtime@' key to the second row.

i Header Row

It's important to note that the top row of the Simple Table is a **Header Row**. In the Property Inspector, there is one Header Row, but you can always add more rows by changing the value, or even setting it to '0' to remove the Header Row.

- In the **Property Inspector**, change the number of **Rows** and **Columns** to 1. If you need to add or remove columns or rows, you can always change the Column and Row property values. Because the data keys determine how many columns and rows get created in your table, the Simple Table will add a new column for each value in the 'equipment' key, which will be visible on the **Preview** panel.



- Next, switch to the **Preview** panel to check out your report. You can always go back to the **Design** panel to edit your table and update your report format.



Report Overview



Data



Design



Preview



Schedule

Simple Table

conveyor line	filler	labeler	palletizer
78	68	84	27

Report Schedules

Perhaps one of the most compelling features of the Reporting module is the Scheduling system. Once a report has been designed, it can be run and delivered automatically without using a client. Reports run on the Gateway, giving you total flexibility when reports should run, and how they should be delivered.

Scheduling a Report

For any Scheduled Report, there are three settings that need to be configured:

1. When the Scheduled report executes (i.e., date, time, and frequency). There can be multiple schedules on a report, and each schedule can be refined down to a minute resolution.
2. What Parameters will be used to determine the report data. Can use the default parameter values or have new values passed in.
3. The Actions that occur following report generation. There can be multiple actions per schedule.



Schedules executed on the Gateway

It is important to note that schedules execute on the Ignition Gateway, not in the client. This means that schedules should be created to Gateway local time, not client or Designer time. Reports can also be run in the client using the [Report Viewer component](#).

On this page ...

- [Scheduling a Report](#)
- [Schedule Table](#)
 - [Schedule Tab](#)
 - [Parameters Tab](#)
 - [Actions Tab](#)





INDUCTIVE
UNIVERSITY

Report Schedule Tab

[Watch the Video](#)

Schedule Table

The Schedule Table is at the top of the Schedule tab and will display a list of all currently configured schedules and actions. A single report can have multiple scheduled times and actions configured, allowing it to be saved with different parameters, at different times, and with different actions.

Creating a scheduled report is easy. To add a new schedule click on the **Add**  icon on the top-right corner of the Schedule panel. In doing so, you've created a new Schedule which can now be configured. To remove any rows, simply select the row in the table and click the **Delete**  icon on the right side of the panel. To configure the schedule, select it, and fill in the tabs below the table: **Schedule**, **Parameters**, and **Actions**.

Schedule Tab

The **Schedule** tab is where the scheduled time is set for the **report** to run. Schedules are driven via [Crontab formatted strings](#), a popular scheduling format used in computing. The intuitive user interface allows you to set schedules easily, even if you are not familiar with Cron. The Schedule GUI provides some convenient pre-made **Common Settings**. If there isn't a setting for you in the Common Settings combo box, choose one that is close and then simply customize it using any of the selection boxes below.

You can also enable or disable the schedule by checking the **Enabled** option in the lower left side of the panel.

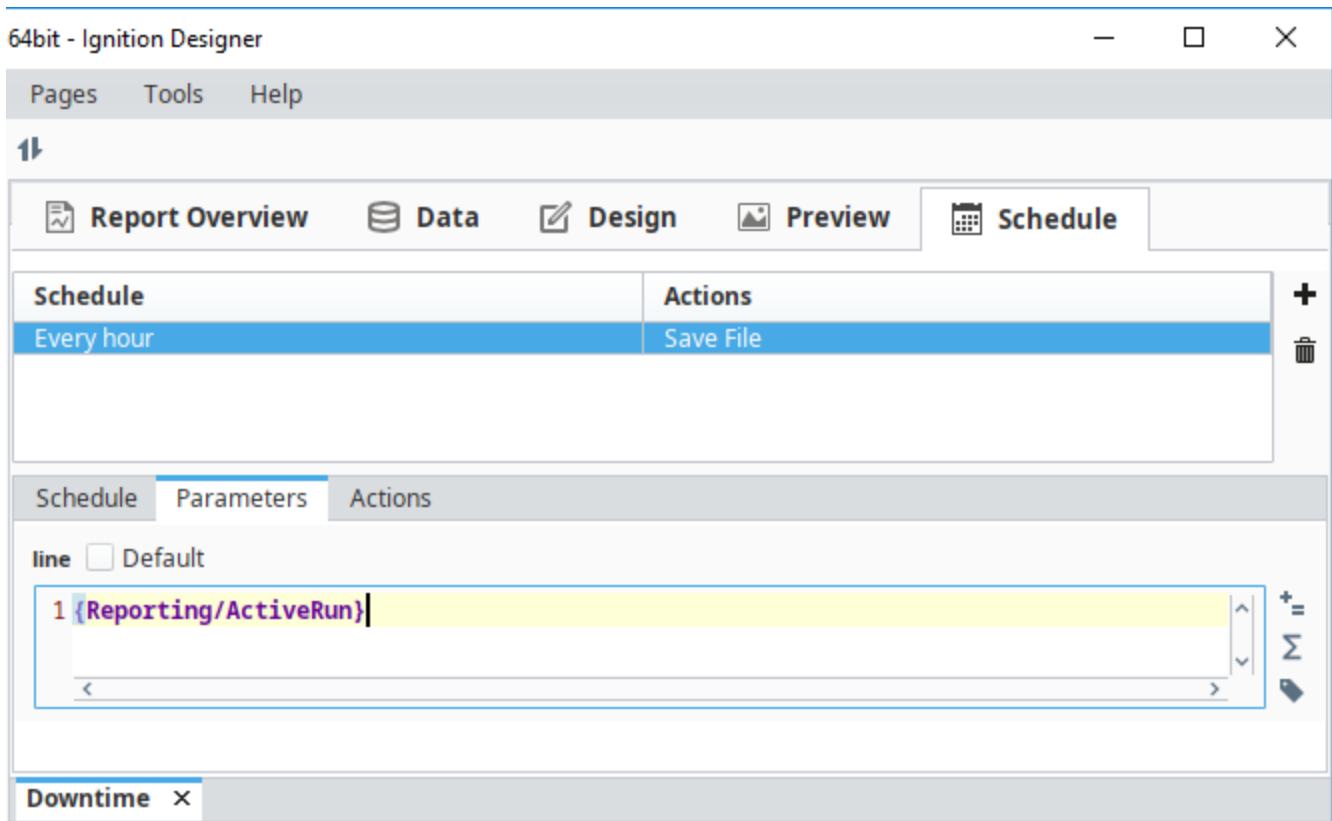
The screenshot shows the 'Schedule' configuration interface. At the top, there are tabs for 'Report Overview', 'Data', 'Design', 'Preview', and 'Schedule'. Below these is a table with two columns: 'Schedule' and 'Actions'. The first row in the table is 'At 12:00 am, on day 1 and 15 of the month (disabled)' with the action 'Save File'. Below the table are three sub-tabs: 'Schedule', 'Parameters', and 'Actions'. The 'Schedule' sub-tab is active, showing various settings. Under 'Common Settings', there is a dropdown menu set to 'On the 1st and 15th of Every Month (0 0 1,15 * *)'. Below this are five rows of settings, each with a text input, a green checkmark, and a dropdown menu: 'Minutes' (0, :00 (0)), 'Hours' (0, Midnight (0)), 'Days' (1,15, On the 1st and 15th of the Month (1,15)), 'Months' (*, Every Month (*)), and 'Weekdays' (*, Every Day (*)). At the bottom left, there is an 'Options' section with a checkbox labeled 'Enabled', which is currently unchecked and highlighted with a red box. Below the 'Options' section is a 'Downtime' section with a close button (X).

Parameters Tab

The **Parameters** tab is where you can override the default values of your parameters when the schedule runs. You can alter these default values to tailor scheduled reports without having to change the Report's design or configuration. For instance, imagine a report which summarizes how many widgets a factory produced during a given shift. Rather than create a separate report for each shift, you can create multiple schedules (one for each shift) and simply alter a shift parameter. Using Parameters and Schedules, users can avoid creating multiple reports while keeping projects more maintainable.

Each parameter of the report will be listed. They can either be set to their default parameter value by selecting the checkbox, or they can be customized by deselecting the checkbox and specifying a parameter value to pass in at the time the scheduled action executes.

For example, we can deselect the Default checkbox for a parameter and override it to the value of a Reporting/ActiveRun Tag by typing **{Reporting /ActiveRun}** in the field below.




Actions Tab

The **Actions** tab is where you can set up actions that will run following the generation of a scheduled report. There can be any number of Actions associated with a Schedule, covering virtually any requirement for automatically storing, distributing, or notifying upon completion of the report. Each action has its own custom configuration interface to make adding and editing Actions simple. To learn more about how to configure Actions, refer to [Scheduling Actions](#).

The Actions you can perform following report generation are the following:

- **Print File** - Configure print settings which execute when the report is generated. This is often used to create hard copies of reports automatically.
- **FTP** - Send your report to a file server for backups or storage. Automatically backs up your reports to a service.
- **Save File** - An easy way to save a report to a location on your local Gateway computer or shared network drive.
- **Email** - Email your report to a list of email addresses or users with specific roles. You can configure the Subject, Filename, and Body of the email.
- **Run Script** - A Run Script Action provides the ability to fully customize how a finished report is handled. The Run Script Action provides the report's data as well as the bytes generated according to the **Format** option in the configuration panel.

With any of these scheduling actions, you also have the option of running the report immediately, by clicking the **Double Arrow**  icon next to the action.

Report Overview Data Design Preview Schedule

Schedule	Actions
At 12:00 am	Save File, Email

Schedule Parameters Actions

Save File
Email

From Address

Subject

Attachment Filename

Body

Mail Server

Format

Retries

Address Source

[Create new server](#)

Recipient and ReplyTo Emails

Address	Method
user.name2@mail.com	To



In This Section ...

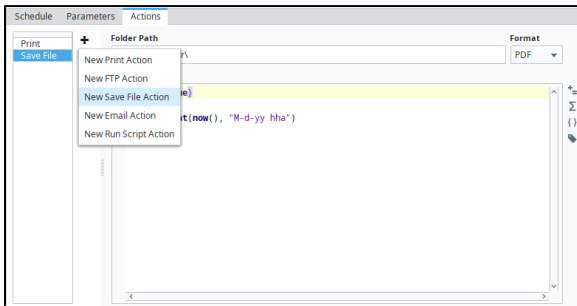
Scheduling Actions

Scheduling Actions

Actions can be configured to run once the report has generated at the scheduled time. Each action has its own custom configuration interface to make adding and editing Actions simple. The Actions you can perform following report generation are Print, FTP, Save, Email, and Run Script. You can even have multiple actions on the same schedule. So you can save the report to the hard drive, as well as email it out to multiple users.

Before creating any Scheduling Actions, you must first [create a schedule](#).

To create an Action, click the **Actions** tab, click the plus icon **+** and select an **Action**. Actions can be deleted using the trash  icon, and executed immediately using the **Double Arrow**  button.



On this page ...

- [Scheduling Actions](#)
 - [Format Options](#)
- [Print Action](#)
- [FTP Action](#)
- [Save File Action](#)
- [Email Action](#)
 - [Configuring an Email Action](#)
- [Run Script Action](#)
 - [Arguments](#)
 - [The dataMap Argument](#)

Format Options

Many of the actions will create a file out of the report. The following file formats are available.

- PDF (recommended)
- HTML
- CSV
- RTF
- JPEG
- PNG
- XML
- XLS
- XLSX

A Note on xls and xlsx Formats

The XLS and XLSX format options may return less than pixel perfect results. This is due to how many spreadsheet programs interpret the resulting file. As a result, the PDF format is recommended in most cases.

Print Action

The Print Action is used to send a report to a printer that is accessible from a computer Ignition is installed on. Here are a list of property descriptions for the **Print Action**.

Property Name	Description				
Primary Printer	The primary method of printing the report.				
Backup Printer	A backup method of printing the report. Will print using this option if the Primary Printer fails. [Optional]				
Print Mode	The mode to print the report in. Can be either Vector or Raster.				
	<table border="1"> <thead> <tr> <th>Print Mode</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> </tr> </tbody> </table>	Print Mode	Description		
Print Mode	Description				

	Vector	Uses math to draw shapes using points, lines, and curves. The most common types of vector graphics are fonts and logos. PDF is a popular vector type. Vector based graphics like SVG image files show images with no pixelation when the size is changed.
	Raster	Are composed of thousands of pixels or dots. Set the dpi (dots per inch). Common raster file format extensions are jpg, jpeg, png, tiff, bmp, and gif.
Copies	The number of copies of the report that will print.	
Print on both sides	Will attempt to print on both sides of a sheet of paper, if supported by the printer.	
Collate	Orders the pages so that a complete report prints before the next copy prints, if applicable.	
Use AutoLandscape Mode	Evaluates the page dimensions and determines portrait or landscape orientation.	
Page Orientation	The orientation of the page. Can either be Portrait or Landscape.	

Schedule
Parameters
Actions

Print

Save File

+ **Primary Printer**

Default Printer
↕
↻

▶▶ **Backup Printer**

Adobe PDF
↕
↻

🗑️

Print Mode
 Vector
 Raster 300 dpi

Copies

2
↕

Options
 Print on both sides
 Collate
 Use AutoLandscape Mode

Page Orientation

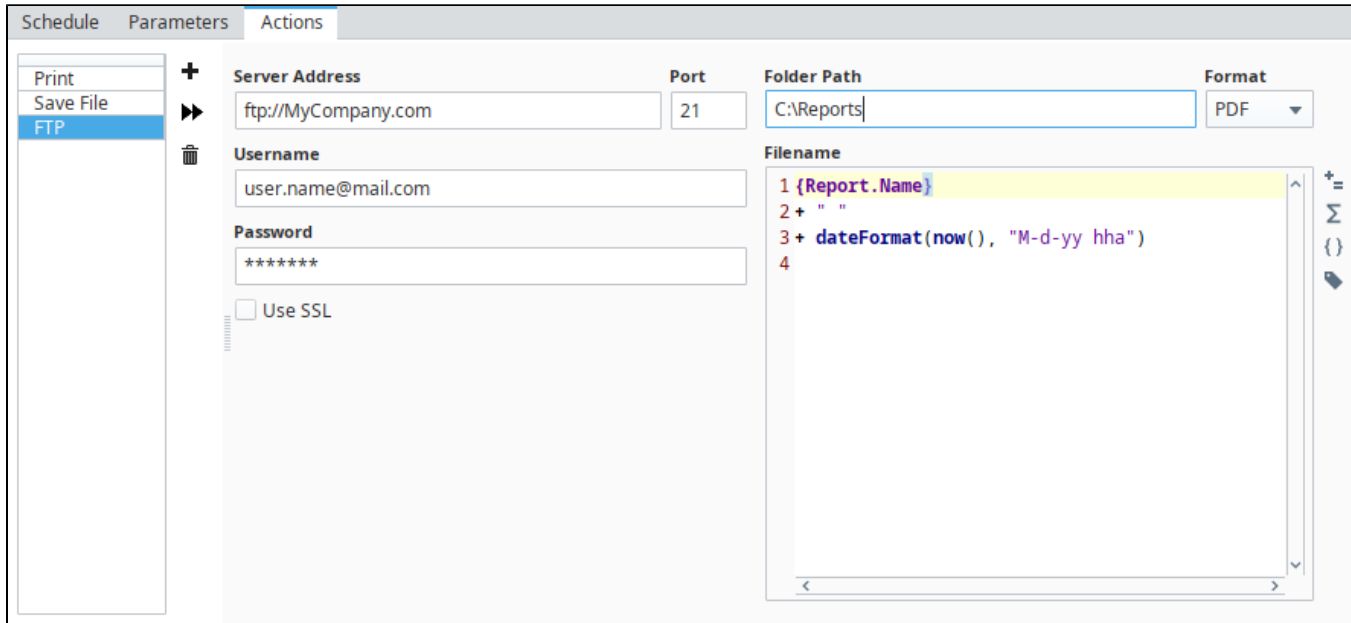
Portrait
▼

FTP Action

The **FTP Action** can be used to automatically upload your reports to a file server for backups or storage. Here are a list of property descriptions for the **FTP Action**.

Property Name	Description
Server Address	The server address where the report file will be transferred.
Port	The port of the file transfer.
Folder Path	The folder path that the report file will be transferred to.
Format	The file format of the report.
Username	The username that will be used to access the FTP server.
Password	The password that will be used to access the FTP server.

SSL	Will use SSL encryption if True.
Filename	The name of the report file. The Filename property is constructed using the expression language.



Save File Action

The **Save File Action** will save a copy of the report to any folder the Ignition server has access to, such as a local folder or network shared drive. Here are a list of property descriptions for the **Save File Action**.

Property Name	Description
Folder Path	The folder path to save the report files to. This folder path is for the Ignition Gateway server.
Format	The file format of the report.
Filename	The name of the report files. The Filename property is constructed using the expression language.



INDUCTIVE
UNIVERSITY

Scheduling Actions
- Save

[Watch the Video](#)



Email Action

The Email Action distributes a report via email when the report is finished executing. There is a Recipients Source property that allows you to send emails using either Email Addresses or User Roles. The 'From Address,' 'Subject,' 'Body,' and 'Attachment Filename' are all configurable. The Subject, Filename, and Body editors can utilize Expressions to dynamically add content or change names.

Email Server settings must first be configured on the Gateway webpage under **Configure > Networking > Email Settings** page, or in **Email Actions** and clicking the **Create new server** link. Once you create



INDUCTIVE
UNIVERSITY

and save an SMTP profile, you can test your email settings for your mail server on the Gateway webpage under **Email Settings**.



Scheduling Actions - Email

[Watch the Video](#)


Creating an email server

Before you set up any reports to be emailed, an email server must be configured. To create an email server if one doesn't exist, use the '**Create new server**' link. This link will take you to **Configure > Email Settings** on the Gateway webpage. There, you will be able to create an SMTP server. For more information, refer to [Gateway Settings](#).

Here are a list of property descriptions for the **Email Action**.

Property Name	Description																
From Address	The Email address from which the report is sent from.																
Mail Server	The mail server to use to email the report. If one doesn't exist, click on the Create new server link. Refer to Email Settings for more information on that page.																
Format	The file format of the report.																
Retries	The number of retry attempts if the email that was sent failed to be delivered the first time.																
Address Source	<p>Will decide how email addresses are collected. Can be either Email Addresses or User Roles.</p> <div style="border: 1px solid #ccc; padding: 5px; margin: 5px 0;"> <p> There is a 'ReplyTo' Email function that allows you to reply to email actions using the Email Address and User Roles. This simply adds those emails to the "ReplyTo" header of the email sent to the recipient list, so that if recipients choose to reply to that email, their reply is sent to those email addresses as well.</p> </div> <p>A table of email addresses with a method that determines how those addresses will be used. Click the plus icon  on the right side of the window to add additional rows.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Property Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Addresses</td> <td>A list of email addresses.</td> </tr> <tr> <td>Method</td> <td>The corresponding method of what to do with the email addresses. Options are To, CC, BCC, and ReplyTo.</td> </tr> </tbody> </table> <p>A list of roles where anyone with the given role in the specified user source with an email address will receive an email.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Property</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Recipient User Source</td> <td>The User Source to pull users from that match the Recipient Roles to get an email.</td> </tr> <tr> <td>Recipient Roles</td> <td>A list of roles to match with users. Any user that has any of the listed roles will get an email.</td> </tr> <tr> <td>ReplyTo User Source</td> <td>The User Source to pull users from that match the ReplyTo Roles that will be listed in the reply to of the email.</td> </tr> <tr> <td>ReplyTo Roles</td> <td>A list of roles to match with users. Any user that has any of the listed roles will have their email listed in the reply to of the email that gets sent out.</td> </tr> </tbody> </table>	Property Name	Description	Addresses	A list of email addresses.	Method	The corresponding method of what to do with the email addresses. Options are To , CC , BCC , and ReplyTo .	Property	Description	Recipient User Source	The User Source to pull users from that match the Recipient Roles to get an email.	Recipient Roles	A list of roles to match with users. Any user that has any of the listed roles will get an email.	ReplyTo User Source	The User Source to pull users from that match the ReplyTo Roles that will be listed in the reply to of the email.	ReplyTo Roles	A list of roles to match with users. Any user that has any of the listed roles will have their email listed in the reply to of the email that gets sent out.
Property Name	Description																
Addresses	A list of email addresses.																
Method	The corresponding method of what to do with the email addresses. Options are To , CC , BCC , and ReplyTo .																
Property	Description																
Recipient User Source	The User Source to pull users from that match the Recipient Roles to get an email.																
Recipient Roles	A list of roles to match with users. Any user that has any of the listed roles will get an email.																
ReplyTo User Source	The User Source to pull users from that match the ReplyTo Roles that will be listed in the reply to of the email.																
ReplyTo Roles	A list of roles to match with users. Any user that has any of the listed roles will have their email listed in the reply to of the email that gets sent out.																
Subject	The subject of the Email. The Subject property is constructed using the expression language.																
Attachment Filename	The name of the attached report. The Attachment Filename property is constructed using the expression language.																
Body	The body of the email. The Body property is constructed using the expression language.																

Configuring an Email Action

1. In the **Schedule** panel, create a **Schedule** to automatically email a report by clicking on the plus  icon, if you don't already have one.
2. Next, click on the **Actions** tab.

3. Click on the plus icon **+**, and select the **New Email Action** from the dropdown list.
4. Enter the sender's email address in the **From Address** field.
5. Select the **Mail Server** from the dropdown list. If one does not exist, click the **'Create new server'** link to create one.
6. Select the **Format** from the dropdown list.
7. Enter the number of **Retry** attempts in the event the email failed to be delivered the first time.
8. You can send emails to users using either Email Addresses or User Roles. Under **Address Source** select either **Email Addresses** or **User Roles**.
 - a. **Email Addresses** - enter individual email addresses under in the **Recipient and ReplyTo Emails** area. To add multiple addresses, click the plus icon **+** on the right side of the window. Next, specify the **Method** of how to send the email: **To**, **CC**, **BCC**, or **ReplyTo**

The screenshot shows the 'Actions' tab of the Ignition configuration interface. The 'From Address' is 'JohnDoe@inductiveautomation.com'. The 'Mail Server' is 'SMTPtest'. The 'Format' is 'PDF' and 'Retries' is '0'. The 'Address Source' dropdown is set to 'Email Addresses'. The 'Recipient and ReplyTo Emails' table has the following data:

Address	Method
Trejo@inductiveautomation.com	To

- b. **User Roles** - select the User Source from the dropdown in the **Recipient User Source** field.
 - i. In the **Recipient Roles** field, begin typing a configured role and Ignition will validate it.
 - ii. In the **Reply to User Source**, select the User Source from the dropdown. (Optional)
 - iii. In the **ReplyTo Roles** field, enter the role(s) you want listed in the 'ReplyTo' header of the email. (Optional)

The screenshot shows the 'Actions' tab of the Ignition configuration interface. The 'From Address' is 'Trejo@inductiveautomation.com'. The 'Mail Server' is 'SMTPtest'. The 'Format' is 'PDF' and 'Retries' is '0'. The 'Address Source' dropdown is set to 'User Roles'. The 'Recipient User Source' is 'default', 'Recipient Roles' is 'Administrator', 'ReplyTo User Source' is 'default', and 'ReplyTo Roles' is 'Administrator, Driver'.

9. Enter in values for the **Subject**, **Attachment Filename**, and **Body** fields, or use the defaults.

Note: Email recipients can choose to reply to the email if they prefer, since the email address is added to the 'Reply To' header of the email.

Run Script Action

This **Run Script Action** allows you to store your report in a database, provide special email code, or anything else you can think of. Run Script exposes the function **handleFinishedReport()** which gives you the report name and path, a mapping of the report parameters and datasets, and the bytes in whatever format you want.

Here are a list of property descriptions for the **Run Script Action**.

Property Name	Property Description
---------------	----------------------



Run Script	An area where a script can be created to do something at the scheduled time.
Format	The file format that the reportBytes parameter should be.

Scheduling - Run Script

[Watch the Video](#)

The screenshot shows the Ignition Gateway interface with the 'Actions' tab selected. The 'Run Script' action is highlighted in the left sidebar. The main editor displays the following Python code:

```

1 def handleFinishedReport(reportName, reportPath, dataMap, reportBytes):
    """
    Provides an opportunity to perform script-based manipulation, review or
    action on a generated Report following its execution. Scheduled Report
    Actions execute on the Ignition Gateway and will have Gateway scope, file
    paths, etc.

    Arguments:
        reportName: The name of the report for which this script should run
        reportPath: The path of the report in your project
        dataMap: A PyDictionary containing the data Objects that were supplied
                  to the report
        reportBytes: The generated Report's byte array in the file format
                     specified in the Format selection box
    """
2

```

The 'Format' dropdown menu is set to 'PDF'.

Arguments

The handleFinishedReport function has the following arguments:

- **String reportName** - The name of the report for which this script should run.
- **String reportPath** - The path to the report in your project.
- **PyDictionary dataMap** - The Python Dictionary containing the Parameters and Data Sources that were supplied to the report. This argument allows you to directly access Parameters and Data Sources in the report. **Note** that once handleFinishedReport() has been called, the report has already been generated, so changing the parameters from this function will not alter the resulting report. Instead, parameters should be altered from the **Parameters** tab.
- **byte[] reportBytes** - The report, presented in a byte array. The format of the report depends on the format specified in the **Format** dropdown list.

The dataMap Argument

There is a special argument in the RunScript Action called **dataMap** that may be used to review the raw data that was used to generate the report. Below is a demonstration of using dataMap.

Using dataMap

```

# The dataMap argument is simply a Python Dictionary with the name of each Parameter and Data Source acting
# as a key.

# Assuming a Report Parameter named 'shift', the value of 'shift' may be accessed with the following
dataMap['shift']

# Similar syntax may be used to extract the value from a Data Source.
data = dataMap['myDataSource']

# Rows objects, while similar in nature to a dictionary, are different objects.

# Individual rows in the Data Source may be accessed by index.
firstRow = data[0]

# getKeys() may be called on a row to list all of the column headers in the row.
firstHeader = firstRow.getKeys()[0]

```

```
# getKeyValue() may be used to access the value of a column in the row.  
firstColumnInRow = firstRow.getKeyValue(firstHeader)
```

Common Reporting Tasks

This section contains examples for items we've identified as common tasks: undertakings that many users are looking to utilize when first starting out with a specific module or feature in Ignition. Additionally, this section aims to demystify some of the more complex or abstract tasks that our users may encounter.

The examples in this section are self-contained explanations that may touch upon many other areas of Ignition. While they are typically focused on a single goal or end result, they can easily be expanded or modified after the fact. In essence, they serve as a great starting point for users new to Ignition, as well as experienced users that need to get acquainted with a new or unfamiliar feature.

On this page ...
<ul style="list-style-type: none">• Report Workflow• Generate Barcodes• Converting Legacy Reports

Report Workflow

Creating a sample report from start to finish. The [Reports Workflow Tutorial](#) will take you through all the different steps on how to bring in data, design a page, and schedule your report. This is a great place to start if you have used the Report module in the past and need a refresher, or if you are new to it and prefer concrete examples over the normal reference style pages of the manual.

Generate Barcodes

Creating sheets of product labels. The report module can quickly and easily generate [Labels with Embedded Barcodes](#). Using one of the special components in the Report Designer components list, you can easily set up standard sizes of paper to print multiple labels from a dataset and even include scan-able barcodes and QR codes. Once printed, these labels can be used for shipping, product identification, or tracking.

Converting Legacy Reports

Convert old reports to the current format. The report module was updated in 7.8 and any reports older than that were designed in a different way. The current report module has a built in converter to [Convert Legacy Reports](#) to the new reporting format so you can start scheduling those reports without recreating them from scratch. This example demonstrates the simple conversion process to get you started in the new format.

[In This Section ...](#)

Tutorial: The Report Workflow

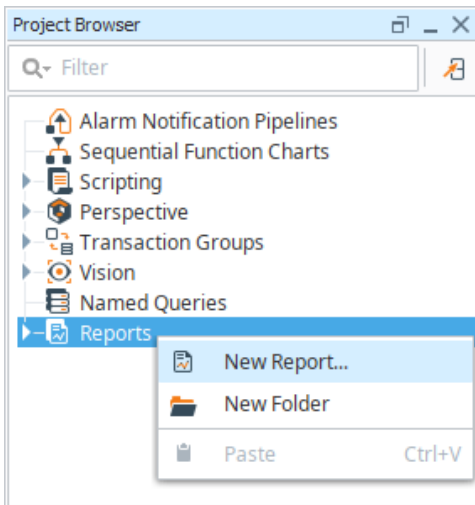
There are three fundamental steps to creating Reports:

1. Gather Data.
2. Design the Report.
3. View or Schedule Delivery.

Conveniently, the Reporting interface makes it simple to complete each step and view your result. This tutorial provides a walkthrough of creating a simple report, and reinforces some of the concepts and terminology involved. If you are already familiar with the basics of how to create a Report, you may be interested in the [Reporting Reference in the Appendix](#).

Creating Your First Report

To create your first report, you will need the [Reporting Module installed](#). Once installed, launch the [Ignition Designer](#), and let's get started! To create a new report, right-click on the Reports node in the **Project Browser**, select **New Report**, and enter a name for your report.



The **Report Workspace** will open, and you'll be in the **Report Overview Panel**. The top of the workspace has a series of stylized panels which guide you through the creation of a new report. It's no coincidence that our Report Workspace is laid out in such a way that lets you intuitively follow the three fundamental steps to creating reports with a little help along the way: gathering data, designing reports, and setting up schedules to run and distribute reports.

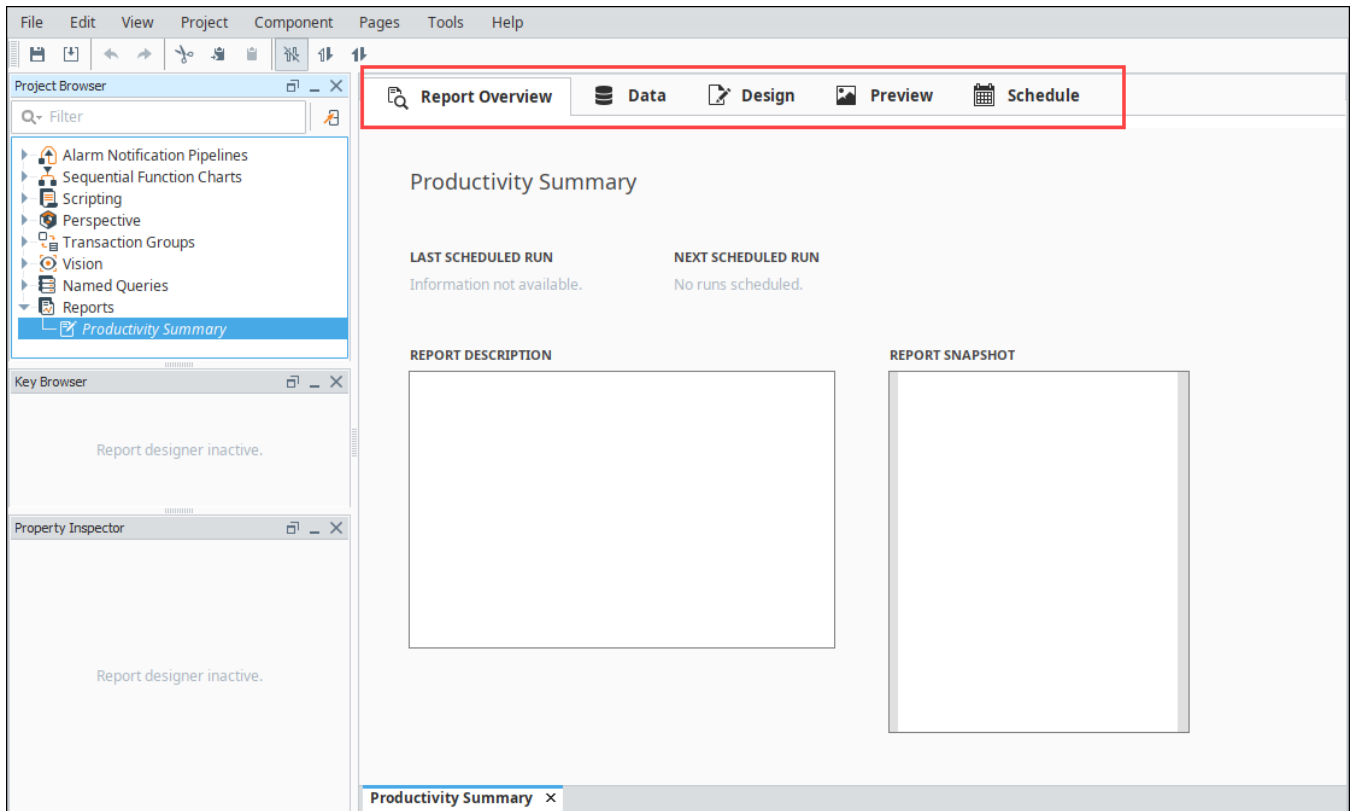
On this page ...

- [There are three fundamental steps to creating Reports:](#)
- [Creating Your First Report](#)
- [Setting Up Data Sources](#)
- [Design the Report](#)
 - [Adding a Table](#)
 - [Configuring Data for your Report](#)
 - [Adding a Header, Date, and Page Numbers to a Report](#)
 - [Adding a Chart to the Report](#)
- [Viewing Reports in Runtime](#)
- [Creating a Schedule](#)
 - [Scheduling a Save Action](#)
 - [Disabling the Schedule](#)



Simple Report

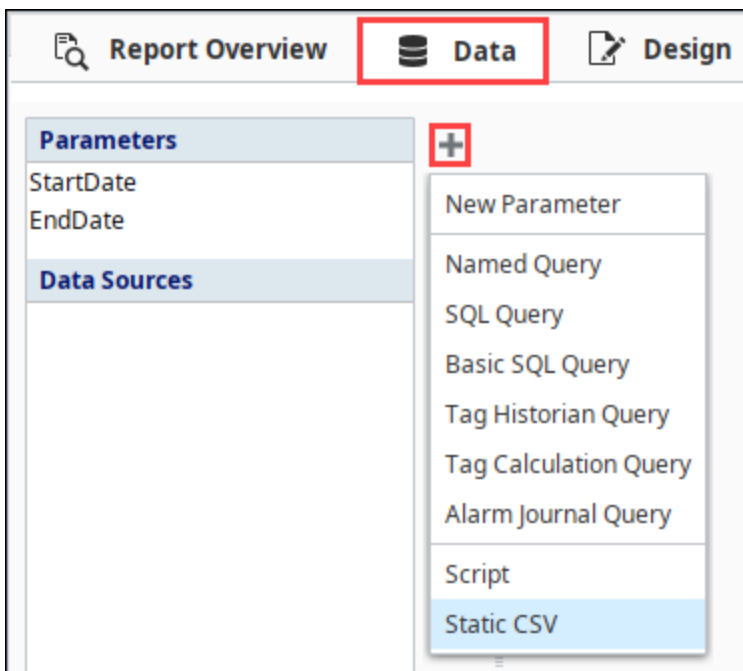
[Watch the Video](#)



Setting Up Data Sources

For any Report, you will first need to specify one or more Data Sources in the **Data** panel.

1. Click the **Add +** icon to add a simple **Static CSV** Data Source, but if you would like to use data from a database connection, feel free to add a **SQL Query data source** instead.



2. With the **Static CSV** Data Source editor open, copy and paste the CSV data below into your Data Source editor, and give your data a meaningful name in the **Data Key** field: we named ours **WidgetProduction Data**. This example models data collected from all the

International Widget Factories, complete with production capacity, number of widgets produced, and the number of minutes it took to produce them.

Static CSV Data Source

```
Facility, Capacity, Produced, Minutes
"California", 2000, 665, 345
"Texas", 3424, 1674, 924
"South Africa", 734, 232, 154
"Brazil", 1131, 882, 325
"China", 5324, 2764, 297
"Norway", 436, 143, 383
"Kenya", 1431, 423, 164
"Italy", 543, 524, 234
"Romania", 154, 78, 45
"Peru", 624, 523, 732
```

The screenshot shows the 'Data' panel of a report design tool. The 'Data Sources' folder is expanded to show 'Static CSV - WidgetProduction Data'. The 'Data Key' dropdown is set to 'WidgetProduction Data'. The 'Data' table is visible, showing the CSV data with the 'Peru' row highlighted.

- Next go to the **Design panel**, and expand the **Data Sources** folder in the **Key Browser**. You'll see that each column of data in the Static CSV data source is represented by its own Data Key. These **Data Keys** are automatically generated based on the table of data fed to the report system. A key may have subkeys, or 'child-keys'. Accessing the data from a child-key is accomplished using the path of a key in relation to its parent, sometimes referred to as a Keychain is simply the path to your data key using 'dot notation'. For instance, the Keychain dragged into a report for our **Facility** key would be `@WidgetProduction Data.Facility@`.

The Key Browser window shows the hierarchy of data keys. The 'Data Sources' folder is expanded to show 'WidgetProduction Data', which contains subkeys for 'Capacity', 'Facility', 'Minutes', and 'Produced'.



The Power of Data Keys

You can have multiple keys in any text field. In addition, you can use multiple keys and common numeric operators to do calculations within the `@` symbols. These are called **Keychain Expressions**.

Now that you have some data, let's create a simple design and see how the Data Keys can be used to create dynamic reports.

Design the Report

The Design Panel is where we start building the report. This section of the report has many components, such as tables and charts, that can display the results of your data sources and parameters. Components in a report are typically assigned one or more keys, which then feeds the results of our data sources and parameters directly to the component. To create comprehensive reports, you will likely find yourself combining different components. Fortunately, this is pretty easy to do using the visual Designer, so let's continue with your first report and see how easy it is.

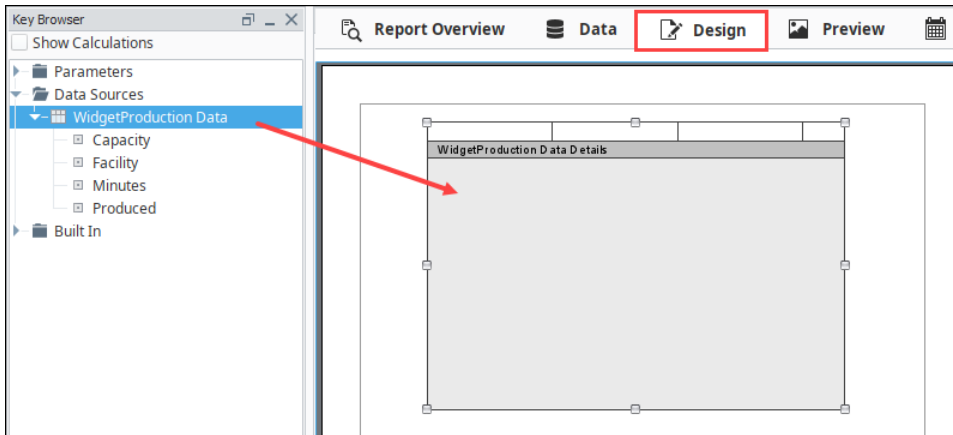
For this report, we will aim for the following requirements:

1. A **Table** that gives a summary of how many widgets each factory produced, and the totals for each data column.
2. Factory efficiency on a units/minute basis which needs to be calculated.
3. A Header / Title for the report.
4. Page numbers in case the report gets to be to long.
5. A **Bar Chart** that visualizes widget production. This will use a separate data source that summarizes the Table's data.


Adding a Table

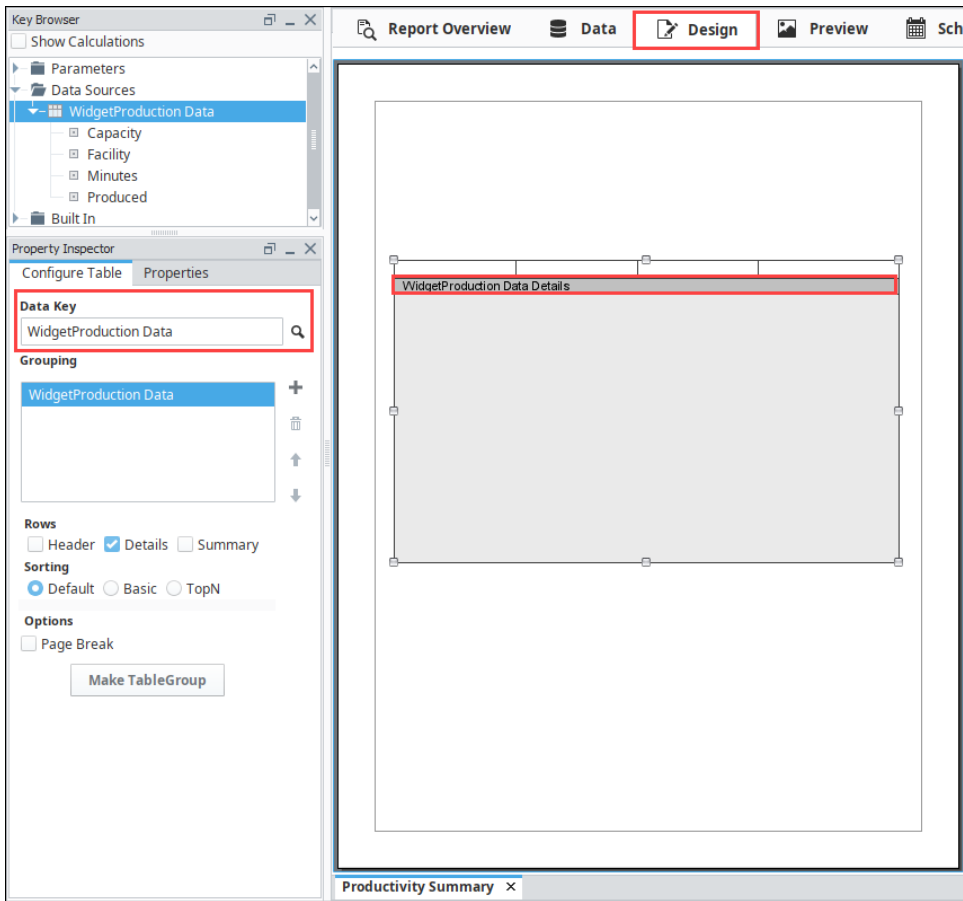
Let's add a table and Datasource.

1. In the **Design panel**, drag and drop the **WidgetProduction Data** key from the Key Browser onto the page. When you let go of the mouse button, this will create a Table component.



2. With the Table selected, in the bottom left corner of the **Property Inspector**, you will see the **Configure Table** tab. Note that the Data Key property on the table was automatically assigned to our **WidgetProduction Data** data source. Additionally, you'll see the Details Row (the dark gray bar on the table component) displays the name of our data source. This means that the table was property assigned a data source.

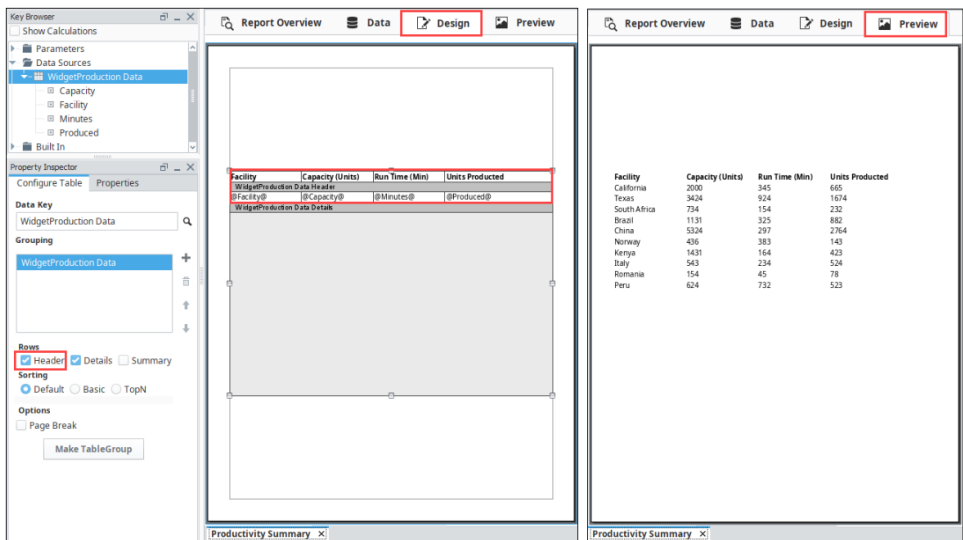
Alternatively, you may drag a Table component from the Component Palette on the right side of the interface. If you take this approach, you must manually assign a data source by interacting with the **Data Key** property on the **Configure Table** tab: click the  icon next to the Data Key property, and select the **WidgetProduction Data** data source. It should look like the image below.



Configuring Data for your Report

Now, let's add some keys to the table.

1. In the **Design panel**, drag and drop data keys from the **Key Browser** (i.e., Facility, Capacity, Minutes, Produced) into the **Details** columns. You'll notice that the keys you dropped are surrounded by '@' symbols. The @ symbols tell the report engine that text inside is a key, and it should try and find the key's value when the report is generated.
2. Next, add a header for each of the columns by enabling the **Header** checkbox under the **Configure Table** tab of the Property Inspector. Next, select each header column, type in your header name, and if you like, you can make it stand out by bolding the text in the **Text Editor** pane.
3. Go to the **Preview panel**, to check the data in your report.

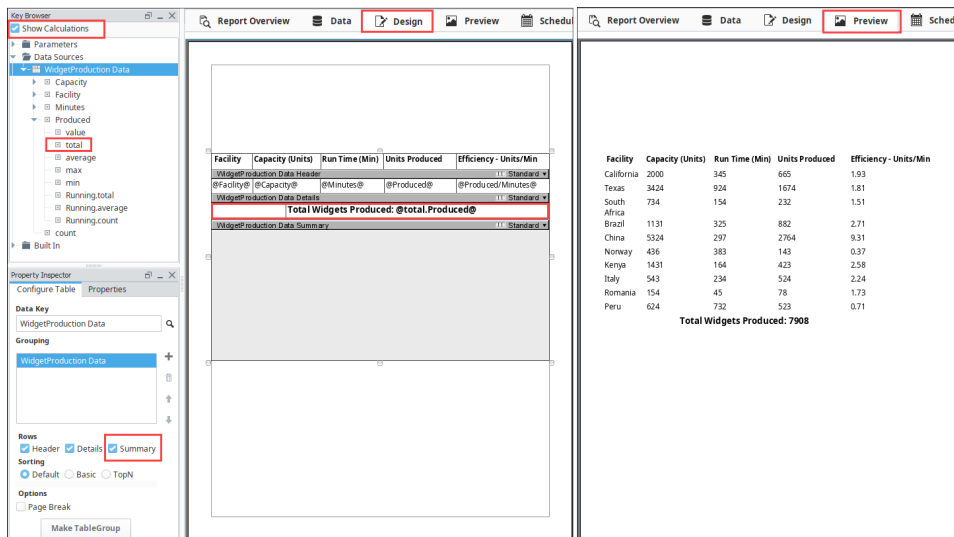


Let's take it a step further. Say you would like to calculate the efficiency of units produced per minute of production runtime. You can use the [keychain expression](#) `@Produced/Minutes@`, which will return the value of the Produced key divided by the value of the Minutes key. Note that this calculation is executed for each row, so it is always taking the current rows values to determine the quotient. To do this, you need to add another column in the Details Row and Header Row, using the following steps:

1. With the Table selected, double click the **Details Row** to select it. This opens the **Property Inspector** containing all the Details properties. Under the **Properties** tab, change the **Column Count** property from '4' to '5'. The new column will be added on the right side of your table. You may have to adjust your column widths to see it.
2. Click on the new cell and enter '`@Produced/Minutes@`'. (Refer to the screenshot in the Design panel below).
3. Double click the **Header Row**, and add a column by changing the **Column Count** property from '4' to '5'.
4. In the **Table**, adjust your column widths, and enter '**Efficiency - Units/Min**' in the new Header column.

Lastly, let's add a total for the number of widgets produced from all facilities to your report using the following steps:


1. With the **Table** selected, mark the **Summary** checkbox in the **Configure Table** tab of the **Property Inspector**. A new Summary row will appear.
2. Double click on the new **Summary Row**, and under the **Properties** tab, change the **Column Count** property from '4' to '2'.
3. In the **Key Browser**, set the **Show Calculations** property to 'true' making the Built-in keys available.
4. Expand the Datasources `WidgetProduction Data` `Produced` object. Drag the **'total'** key from **Produced** to the second column of your new **Data Summary** row.
5. Select the cell you just added to and type the following text in front of your key: '**Total Widgets Produced**'. In the Text Editor you can make it bold so it stands out.
6. Go to the **Preview panel**, to check your report. You should see all your data in the report.



Adding a Header, Date, and Page Numbers to a Report

Header

The [Text Shape](#) is great to use for a report title, although it can be used anywhere in the report to add text to the page.

1. To activate, click on the **Text Shape**  icon on the Component Palette.
2. Once selected, click and drag on the top of the page to create a Text Shape. Give your report a title by typing into the field of the shape.
3. You'll notice the **Edit Text** tab in the lower left corner of the **Property Inspector**, along with some buttons that let you customize the look and layout of the text. This configuration area will change depending on the selected component on the report. In the same area, you'll notice there is a tab titled **Properties**. The Properties tab provides access to a component's various properties. Feel free to experiment with the

settings like font in the editor or property table to customize your title and/or text shapes.



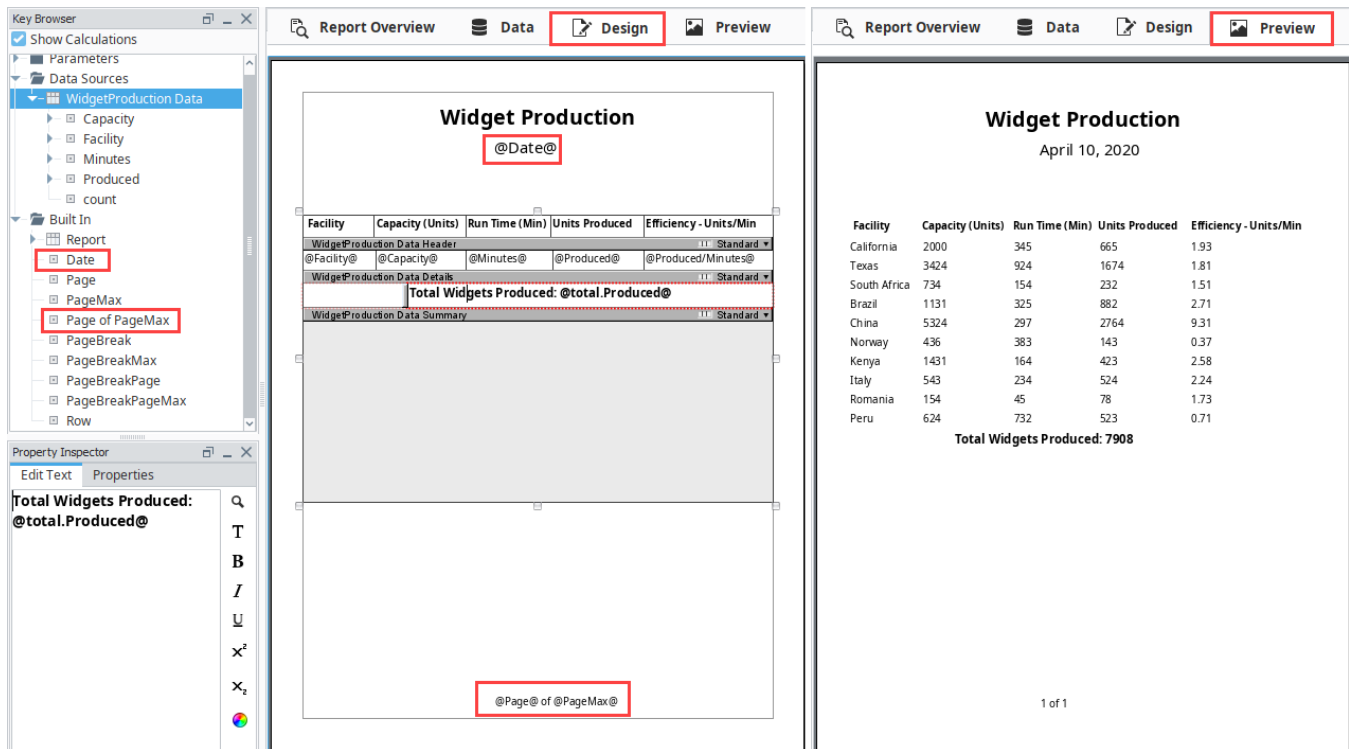
Date and Page Numbers

In addition to a title, let's add some metadata, such as the date the report was generated. However, you don't want to type the date into the text field, because all reports you run (could be today, tomorrow or next year!) will show what date you typed. Instead, you want the date to reflect the day the report was generated.

1. If you look at your **Key Browser**, you'll see a folder called **'Built In'**. Expand this folder and you'll see a number keys that are common in reports.
2. Drag the **'Date'** key and drop it on the report just below your title. This key represents the date and time the report is executed. By default, Text Shapes initially show just the date, but this can be modified by via the **Date Format** property, which is located in under the **Properties** tab when the Text Shape is selected. This example will use the default Date Format, but feel free to modifying this.
3. Next, drag the **'Page of PageMax'** key to the bottom of your report.

Adding text on a page shared with a repeating component (such as a table) will add the text to all pages created by the component. In some cases, like page numbers, date, and title at the top, this is desirable.

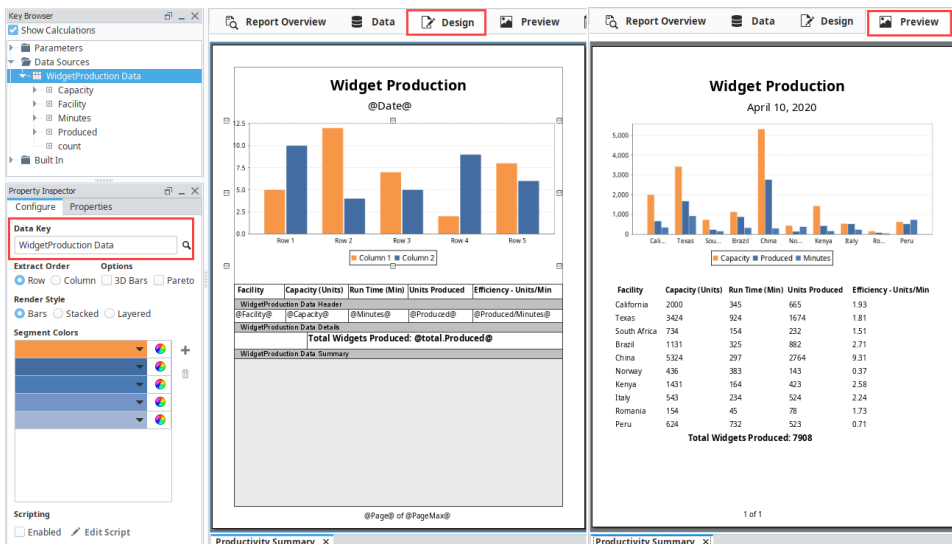
Select the Preview panel to see how the header, date, and page numbers all appear on the report.



Adding a Chart to the Report

To finish the report, we just need to add a chart. You may need to resize the table a little bit and add a chart right under the page header.

1. In the **Design** panel, drag a **Bar Chart** component from Report Palette into the space above the table. You will need to resize the Bar Chart and the Table to your report page.
2. Next, drag the **Data Source** (i.e., WidgetProduction Data) from the **Key Browser** to the **Data Key** on the **Configure** tab.
3. Go to the **Preview** panel to see how your report looks. Instantly, you'll notice the Bar Chart adds a bar for each column in the data, where our goal for this report is to only show the number of widgets produced. If you want to change the colors of the bars, go back to the Design panel and edit the **Segment Colors** under the Configure Tab.



Viewing Reports in Runtime

Once your report is created, you can view the report from the Ignition runtime or Preview Mode of the Designer with the **Report Viewer** component. You can add the Report Viewer component to any Perspective View and Vision window. In Perspective, enter source path for the report by doing a copy path on the report. In Vision, select the name of your report from the **Report Path** parameter dropdown. To learn more, refer to [Perspective Report Viewer](#) and [Vision Report Viewer](#) pages.

Saving a report from the Report Viewer is simply a matter of right clicking on the report in the Ignition runtime or Preview Mode of the Designer, and selecting the format you wish to save it as. Selecting from the menu will open a Save or Print dialog in the Client window as well as in Preview Mode.

Saving a report from the Report Viewer in Vision is simply a matter of right clicking and selecting the format you wish to save it as from the dialog box in the Client and Preview Mode. In Perspective, you have the option to download a [report](#) to your local device or print a [report](#) to your local printer using the built-in controls at the bottom of the report.

Creating a Schedule

Once a report is designed, you can have a list of scheduled times and actions that will execute automatically at specified times using the [Report Schedule](#) functionality.

Use the following steps to create a schedule for your report.

1. In the **Schedule panel**, click on the plus icon **+** on the right side of the panel. A new row is added to the table. You'll notice the user interface is split into two sections. On the top is a table which contains a list of all the schedules for your report as well as the Actions that will occur following report generation.
2. The **Schedule tab** is where you set up the schedule for your report to run. The UNIX Crontab format is used to set up schedules, but there are some common schedules available in the dropdown list to select from. To set a schedule, select a row in the Schedule Table. Once the row is selected, the Schedule section becomes editable.

The screenshot shows the 'Schedule' panel in the Ignition Report Designer. At the top, there are tabs for 'Report Overview', 'Data', 'Design', 'Preview', and 'Schedule'. The 'Schedule' tab is selected and highlighted. Below the tabs, there is a table with two columns: 'Schedule' and 'Actions'. The first row in the table is 'At 12:00 am and 12:00 pm'. To the right of the table, there is a plus icon (+) and a trash icon. Below the table, there are three tabs: 'Schedule', 'Parameters', and 'Actions'. The 'Schedule' tab is selected. Under the 'Schedule' tab, there is a 'Common Settings' dropdown menu set to 'Twice Per Day (0 0,12 * * *)'. Below this, there are several input fields for 'Minutes', 'Hours', 'Days', 'Months', and 'Weekdays'. Each field has a dropdown menu and a checkmark icon. The 'Minutes' field is '0', 'Hours' is '0,12', 'Days' is '*', 'Months' is '*', and 'Weekdays' is '*'. Below these fields, there is an 'Options' section with a checkbox labeled 'Enabled' which is checked.

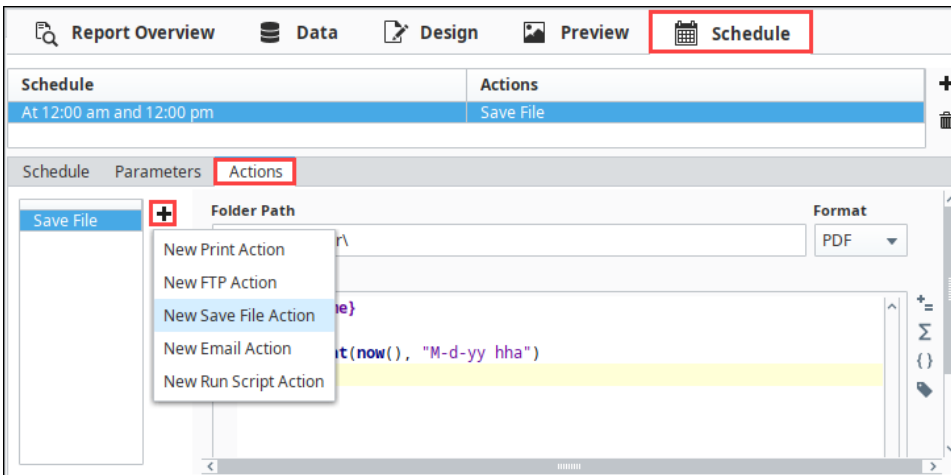
3. To create a new schedule, select a schedule from the Common Settings dropdown list, or choose from one of the Common Settings and customize it using any of the selection boxes. In this example, the **Hours** dropdown was set to **Every 12 hours (*12)**, which will run the report daily every time the current hour changes to 12 (i.e. midnight and noon). Now that a schedule has been created, we need to specify an Action to occur at the scheduled times.

Scheduling a Save Action

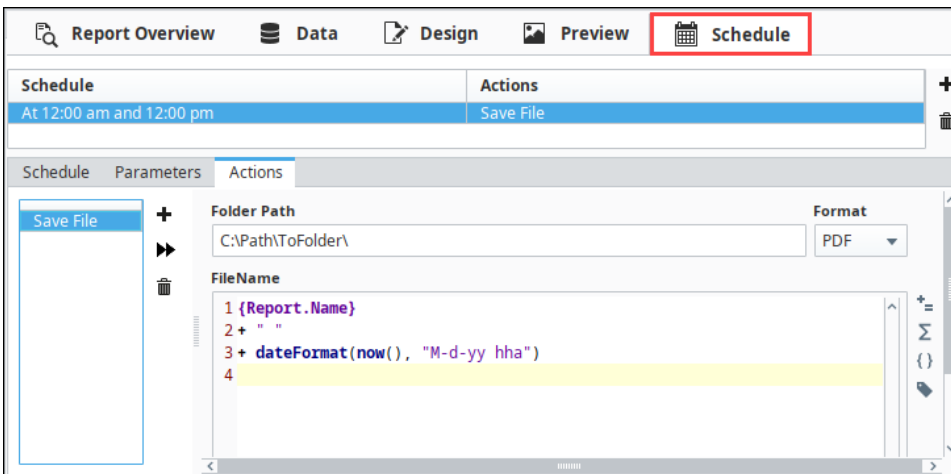
The [Save File Action](#) saves a copy of the report to any folder Ignition has access to whether it's on your local computer or a shared drive.

Here are a few simple steps to saving a report:

1. In the **Schedule** panel, create a **Schedule** to automatically save a report by clicking on the plus icon **+**, if you don't already have one.
2. Next, click on the **Actions** tab.
3. Click on the **Add** **+** icon, and select **New Save File Action** from the dropdown list.



4. Enter the **Folder Path** where you want to save your report to. You can save them to your local computer or a shared drive. Here are a couple of examples of a folder path for a local computer and a shared drive: "**C:\Reports**" and "**Share_Drive\Folder**". This path is always relative to the Gateway, so you should always take the Gateway server's operating system into consideration when specifying the path.
5. Select the file **Format** from the dropdown list that you want to save the report as. PDF is a very common format.
6. By default, your file will be saved with your report name followed by month, day, year, and hour. If you prefer not to use the default filename, you can change it.



7. Save your project. Your schedule is now configured and you can test out the action by clicking on the **Run Immediately** icon.

Disabling the Schedule

Now that your schedule is complete, you may want to disable the schedule so that a new copy of this example report is not created every 12 hours. The **Schedule** tab has an **Enabled** checkbox that can be disabled to prevent the schedule from occurring.

Labels with Embedded Barcodes

Using labels with embedded barcodes is very common practice. Barcodes are typically used to print out mailing and shipping labels, and can be used to identify and track almost anything. When creating any type of label, you can specify your own dimensions or use standard Avery label sheets. The size of the labels component is based on the values in the Label Attribute section of the Configure Labels tab. When you update any of the Label Attributes (i.e., Rows, Columns, Width, Height, and Spacing), the labels automatically resize based on the values that were specified.


The best way to explain how to set up labels with embedded barcodes is to use an example. Let's create a set of custom labels for the University Research Lab to better track the DNA Primers used in experiments. We want the labels to contain information about each Primer and where it belongs. In addition, we want the barcode to contain the Primer name, number of the lab, and the location within the lab that the samples belong to. We also want the labels to include a sequence number to identify the process order for each step in the experiment. A barcode will be embedded into the template label to create QR Codes, and a QR Code Scanner will be used to track the Primers.

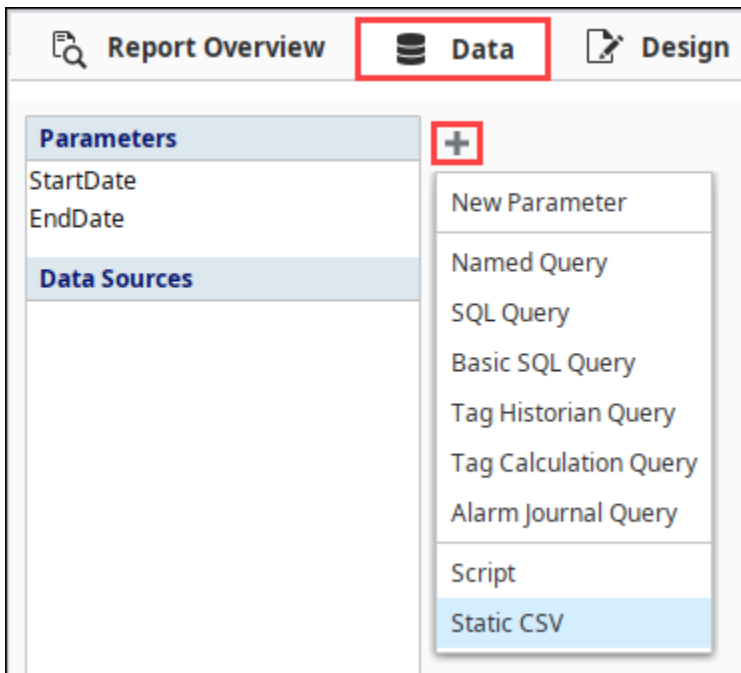
On this page ...

- [Setting Up Data Sources](#)
- [Configuring the Label and Embedded Barcode](#)
- [Printing Labels](#)

Setting Up Data Sources

Before setting up any labels, you first need to specify a **Data Source** in the **Data panel**.

1. Click the plus icon  to add a **Static CSV** Data Source. This example will use the static data listed below, but could easily substitute real data from a database by using a [SQL Query Data Source](#) data source instead.

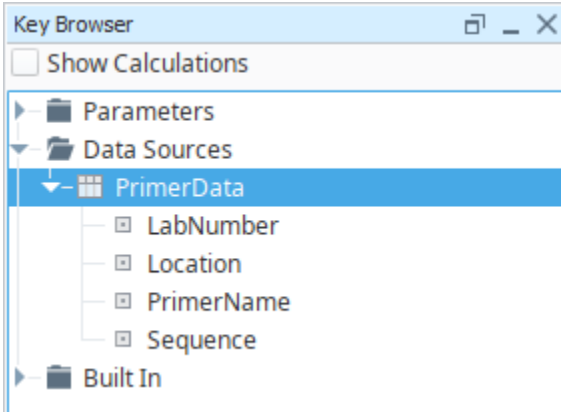


2. With the **Static CSV** Data Source editor open, copy and paste the CSV data below into your Data Source editor, and give your data a meaningful name (i.e., PrimerData) in the **Data Key** field.

PrimerData Datasource

```
PrimerName, Sequence, LabNumber, Location
16F, CGG TTA CCT TGT TAC GACT T, 2, Cooler 2
3A0X, GCA AAT GGC ATT CTG ACA TCC, 3, Freezer
EGFPC1R, CAT TTT ATG TTT CAG GTT CAG GG, 2, Cooler 2
pGLrev, CTT TAT GTT TTT GGC GTC TTCC, 1, Cooler 1
M13R, CAG GAA ACA GCT ATG ACC, 3, Freezer
SV40-promoter, TAT TTA TGC AGA GGC GAGG, 1, Cooler 1
pBabe5, CTT TAT CCA GCC CTC AC, 2, Cooler 2
EGFPC1R, CAT TTT ATG TTT CAG GTT CAG GG, 2, Cooler 2
pGLfor, GTA TCT TAT GGT ACT GTA ACT G, 3, Walk-in Shelf A
EF-1xForward, TCA AGC CTC AGA CAG TGG TTC, 1, Cooler 1
```

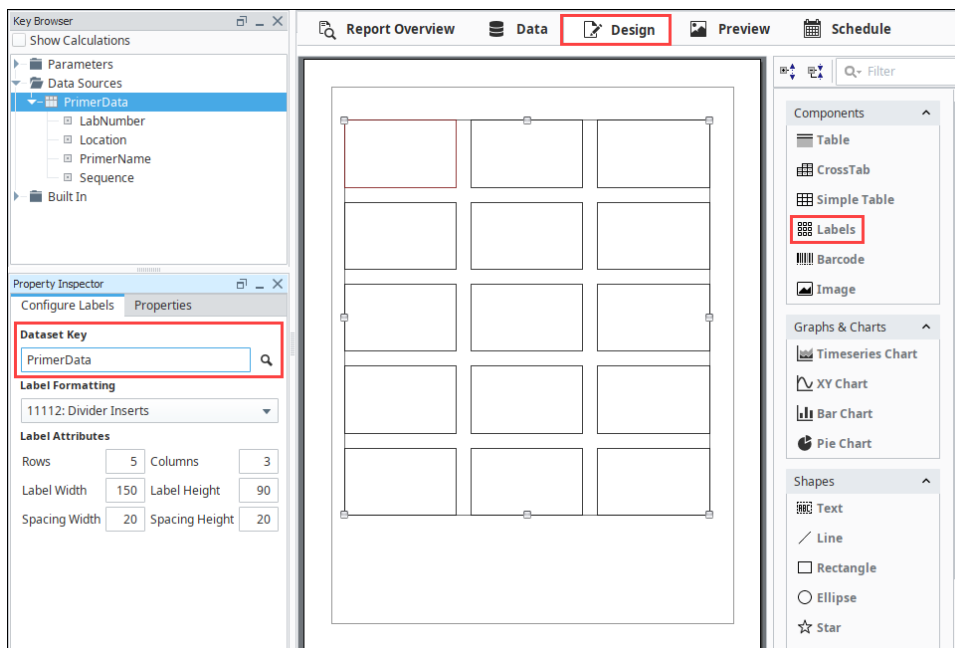
- Next, go to the **Design panel**, and expand the **Datasources** folder in the **Key Browser**. You'll see that each column of data in the Static CSV datasource is represented by its own Data Key.



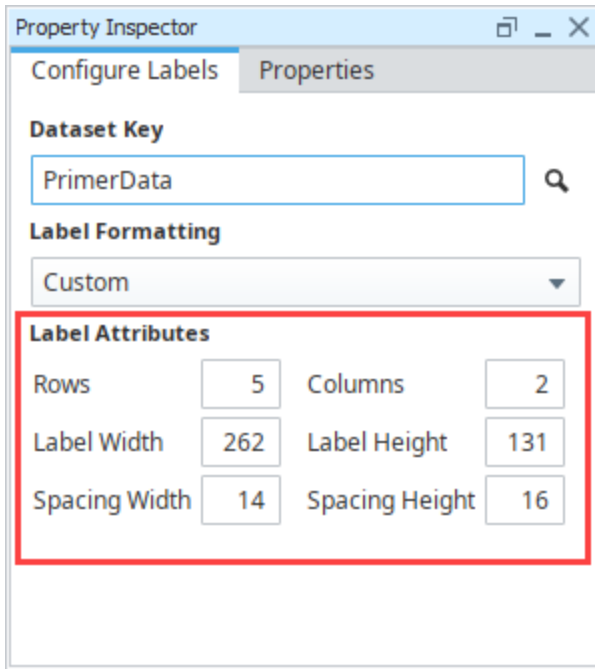
Configuring the Label and Embedded Barcode

Now that you know what information you want on your labels, you have to decide what you want your labels to look like.

- In the **Design panel**, drag a **Labels** component from the Report Component Palette to your workspace. By default, 15 labels will display in your workspace.
- Specify the Data Key that maps to the Data Source you want to drive the label.
- Drag your **Datasource** (i.e., PrimerData) to the **Dataset Key** in the Configure Labels tab of the Property Inspector.
- Select the **Labels** component.



- Edit the number and size of the labels in the **Configure Labels** tab. The size of the label is based on the values for the Label Attributes: Rows, Columns, Label Width, Label Height, Spacing Width, and Spacing Height. When any of the Label Attributes are changed, the labels on the page are automatically resized based on these values.
- Enter any value changes to the Label Attributes and hit **Enter** to commit.



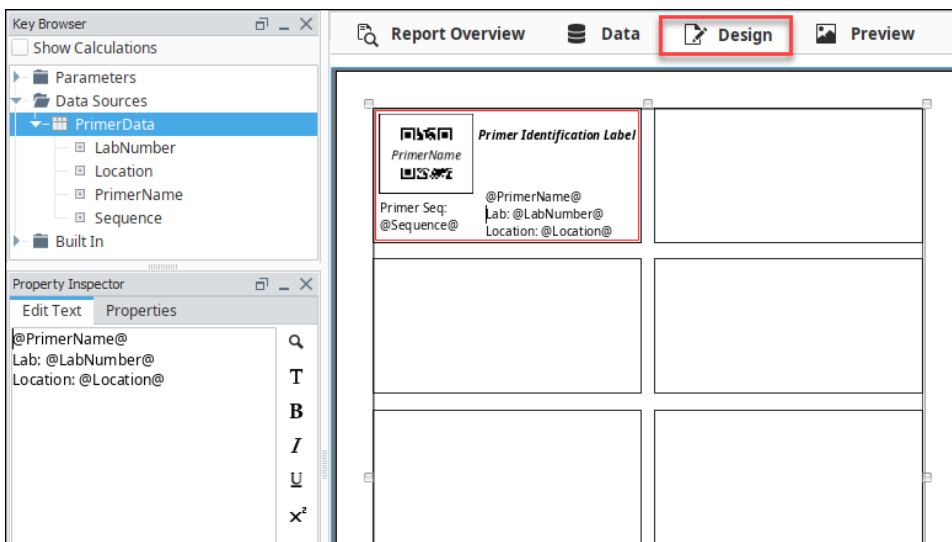
i **Template Label**

The top-left label in the Labels component is the template label. To edit a component or shape within a template label, super select it by double-clicking on it. When you **Preview** and **Print** your labels, the template label pulls all the data from the dataset and populates all your labels.

- Next, drag a **Barcode** component and place it in the upper left corner of the template label.
- Now you're ready to drag your data keys (i.e., PrimerName, LabNumber, and Location) from the Key Browser to the label template.

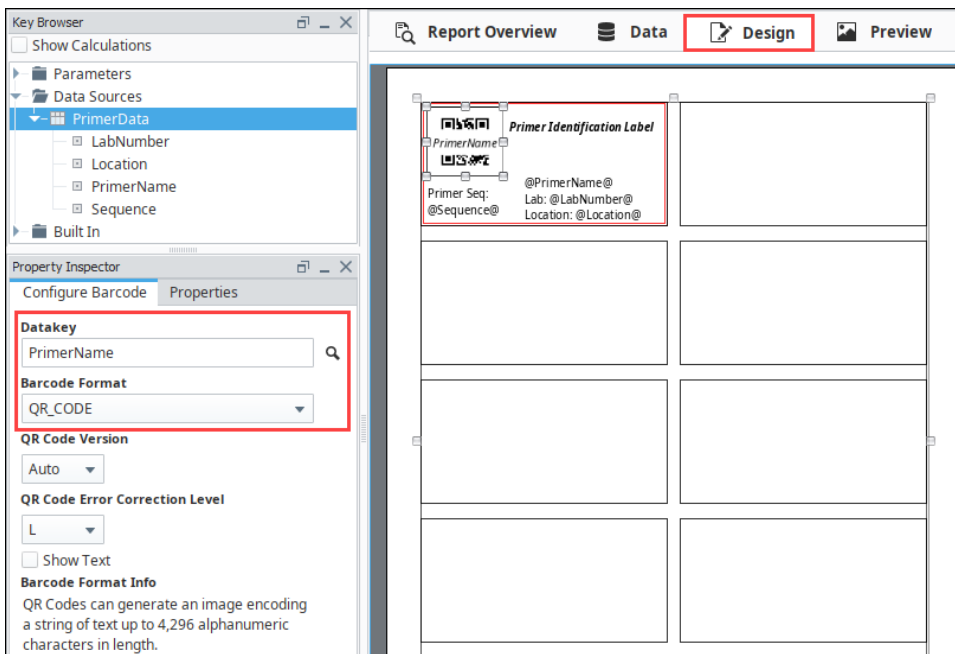
Note: Once you drag the first data key to your label template, the **Edit Text** tab will open. You can drag the other data keys directly into the Edit Text tab so they make one shape. This makes editing a little easier.

- Next, drag the **Sequence** data key into the lower left corner of the label.

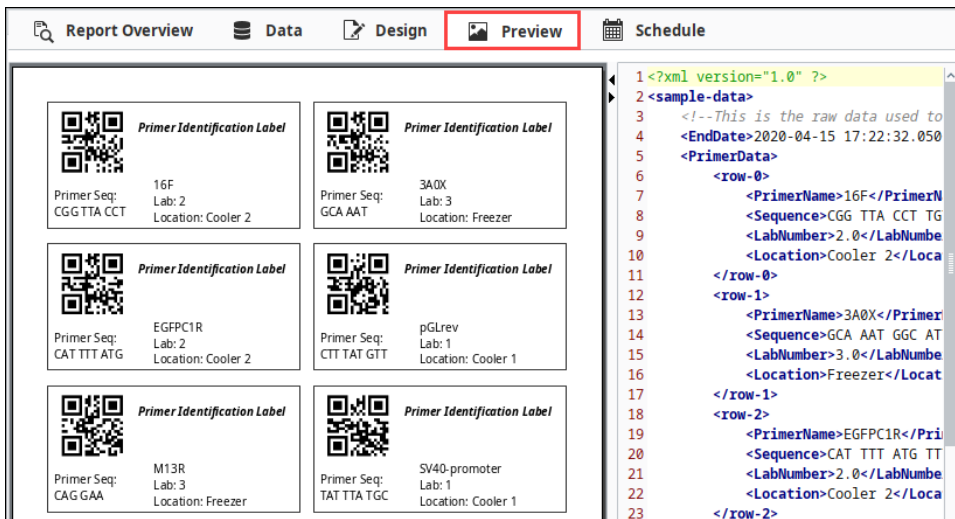


- Go to the **Preview** panel to verify if all your data populated all the labels, and check how all the data fits on the label because you may need to resize the barcode or text components.
- Go back to the **Design** panel, click on the **barcode** component, and specify the data key you want to map to the Data Source to drive the barcode. In this example, we want the barcode to encode the PrimerName to the Data Key field of the barcode.
- Next, choose a **Barcode Format** from the dropdown list based on your label requirements (i.e., product type, number of characters required, label spacing, etc.). For this example, select the **QR_CODE** barcode format. Note: If you want to push multiple fields into the barcode text, you can concatenate them together in the Data section of the report.


13. Add a Title for your label by dragging a **Text Shape** component to the top of the label. Enter a title inside the text box (i.e., Primer Identification Label). In addition, in the Edit Text tab, you can mix plain text and data keys, change the font, size, and style, and even bold text that you want to stand out.



14. Go to the **Preview Panel**, to view the finished design layout and the HTML code. It's not uncommon to go back to the Design panel to resize components and shapes on the label several times so they fit correctly and the data is readable. This may take several iterations.



Printing Labels

You can print your labels by creating a [Print Action](#) on the Schedule tab, and run it by clicking the **Run Immediately**  icon. If you receive new shipments of Primers on a set schedule, you can set up the labels to print automatically. For this example, we want to simply print our labels on

demand.

The screenshot displays a software interface with a top navigation bar containing 'Report Overview', 'Data', 'Design', 'Preview', and 'Schedule'. The 'Schedule' tab is active and highlighted with a red box. Below the navigation bar, a table shows a schedule entry: 'At 12:00 am' with an associated 'Print' action. The 'Print' action is selected in a list on the left, and its configuration is shown in the 'Actions' panel on the right. The 'Actions' panel includes settings for 'Primary Printer' (Default Printer), 'Backup Printer' (None), 'Print Mode' (Vector selected, Raster at 300 dpi), 'Copies' (1), 'Options' (Print on both sides, Collate, and Use AutoLandscape Mode), and 'Page Orientation' (Portrait).

Schedule	Actions
At 12:00 am	Print

Actions

Print +

▶▶

Primary Printer
Default Printer

Backup Printer
None

Print Mode
 Vector
 Raster 300 dpi

Copies
1

Options
 Print on both sides
 Collate
 Use AutoLandscape Mode

Page Orientation
Portrait

Converting Legacy Reports

Overview of Converting Legacy Reports

The complete rebuild of the Reporting Module in Ignition 7.8 brought many improvements that would have been impossible to add to the legacy module. To preserve report functionality and prevent problems with backward compatibility, any existing reports will function as they always have. To get the most out of your reports and enable new functionality in such as Scheduled Reports, you'll have to convert your old Vision Report Panel components to new Report Resources. In an effort to minimize barriers, we created a Report Conversion process that will attempt **to convert Ignition 7.7- reports into 7.8+ reports.**

We encourage users to convert their reports to the new format if they feel they would benefit from the added functionality, but in doing so, it's important to keep some things in mind:

Some Components have Imperfect Conversion

There are a number of components that have been upgraded or completely rebuilt. Due to the changes, some components and/or configurations may not convert perfectly to the newer module. Specifically, the upgrade to the Barcode component and addition of 2D barcodes utilizes a new encoding system that does not have perfect parity with the legacy encoder. The new component does not explicitly support encoders for some of the Narrow and Extended codes, as well as MSI. Reports which require the Narrow or Extended Code 39/Codabar or MSI barcodes will not convert perfectly. Lastly, the changes to the Charts brought many improvements, but will look a little different and may require some configuration.

Data Sources will need Configuration

Data Sources are a huge improvement in the new module. It's now far easier to collect and use data from nearly anywhere in Ignition. Unfortunately, the custom properties in the Legacy Report module do not directly map over to the new Data Sources. When a report is converted, its custom properties will be converted to [Parameters](#). Parameters are great in that they allow for a quick conversion and enable things like [Scheduled Reports](#), but Data Sources will need to be manually configured if desired.

These are the two major caveats to be aware of when converting. The conversion tool has been tested with a variety of legacy reports, but there may be additional factors preventing conversion. We encourage users to either visit the [Inductive Automation forum](#) or contact our [support department](#), and let us know – improvements to the conversion tool can only occur if we are made aware of errors!

Conversions are Non-Destructive

With the potential issues covered, it's important to note that report conversions **do not** destroy or alter the original report. If the conversion isn't successful for some reason, the new report can simply be deleted and the old one exists as it always did. If the conversion is successful, the old report can be saved to an unused window or exported as a backup before being deleted. Conversions may not always be perfect, but there is no risk in trying.

How to Convert

Converting a legacy report is quite simple. In the Ignition Designer, open the window where the Report Viewer component exists. If you right click on the component, the new "**Convert...**" option is at the top of the popup menu. Selecting it will start the conversion process which first prompts for the "OK" to proceed, then asks what the new Report Resource should be called.

On this page ...

- [Overview of Converting Legacy Reports](#)
- [Some Components have Imperfect Conversion](#)
- [Data Sources will need Configuration](#)
- [Conversions are Non-Destructive](#)
- [How to Convert](#)



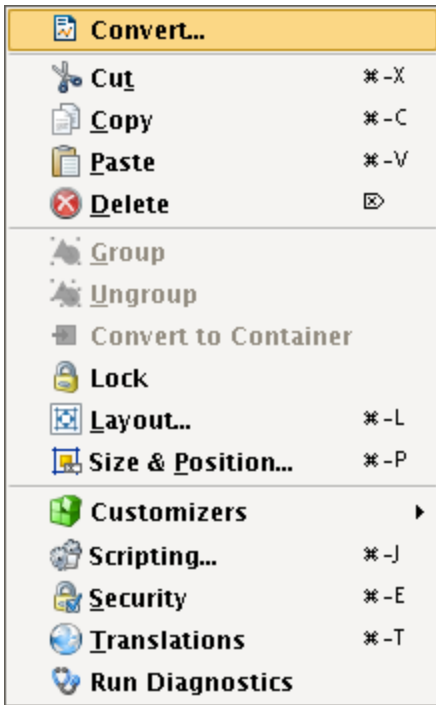
Converting Legacy Reports

[Watch the Video](#)



Sample Legacy Report Conversion

[Watch the Video](#)



Upon conversion, a new Report Viewer will be added to the existing window in addition to the old report. If you check the Project Browser, you'll see the new report under the Reports tree node. Open the newly created Report Resource to make any edits, to add data sources, or add Scheduled Actions!

Related Topics ...

- [Report Data](#)