EspressReport® Version 7.1

EspressReport 7.1 User's Guide

Quadbase

Quadbase Systems Inc.

EspressReport® 7.1

EspressReport User's guide

Welcome to the EspressReport Online User's Guide.

Here you will find information about all the functions and features in EspressReport. For additional API resources, please refer to the API JavaDoc.

EspressReport Users guide consists of several elements:

QuickStart Guide This is a good starting point for EspressReport first time users. This section explains many of the most commonly used features, and provides Report Designer and API exercises to illustrate them.

Designer Guide This explains all functionalities of the Report Designer, how to develop and import data sources, and how to format and manipulate data to create polished reports.

Programming Guide This covers the Report API and teaches you how to create reports programmatically, as well as incorporate them into servlets, JSPs, and applications.

Charting Guide This covers all charting features and teaches you how to format and manipulate charts using the Chart Designer, as well as with the Report API.

Tip

If you're reading the PDF version, enable *Bookmarks* toolbar in your PDF viewer.

Please select an option from the menu on the left to begin.

Quick Start Guide

Q.1. Overview

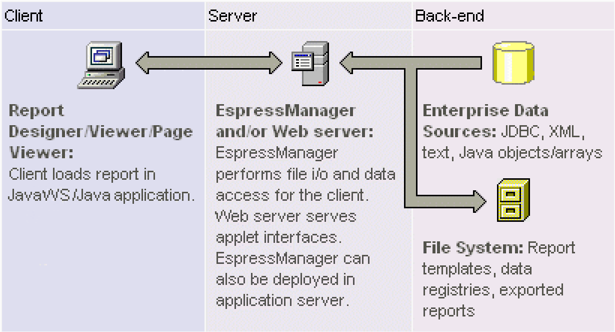
Welcome to EspressReport. EspressReport is a powerful Java reporting tool that makes it easy to design and deploy information-rich reports with your internet or intranet applications. EspressReport can connect to virtually any data source and easily format and render the information in DHTML style sheets, print-quality PDF, Microsoft Excel, or Rich text. It can also extract data to text, CSV, and XML formats. The powerful Java API makes it easy to include reporting functionality in applications, servlets, JSPs, and JavaWS.

This is the EspressReport Quick Start Guide. This is the recommended starting point if you are new to EspressReport and would like to get up to speed fairly quickly, or evaluate the capabilities of the product. This portion of the documentation provides examples for designing and running reports, as well as sample reports and code that illustrate some of the most commonly used features of EspressReport.

Q.1.1. EspressReport Architecture

EspressReport has a number of different configurations in which it can run both at design-time and run-time. At design-time, the Report Designer GUI interface, which includes data access tools and charting tools, can be loaded as an application on a client machine or as an applet in a client server architecture.

When Report Designer is running, the EspressManager component runs on the server-side. The EspressManager performs the data access and the file I/O which is prevented by the client applet due to security restrictions. The EspressManager also provides connection and data buffering for database connections, as well as concurrency control for a multi-user development environment. The EspressManager must be run in conjunction with the Report Designer.


Client
Server
Back-end
Report Designer/Viewer/Page Viewer: Client loads report in JavaWS/Java application.
EspressManager and/or Web server: EspressManager performs file i/o and data access for the client. Web server serves applet interfaces. EspressManager can also be deployed in application server.
Enterprise Data Sources: JDBC, XML, text, Java objects/arrays
File System: Report templates, data registries, exported reports

EspressReport Architecture at Design-Time

At run-time, the EspressManager does not need to be running. EspressReport is designed to run embedded within other application environments and it can have a minimal deployment of the API classes and report template files. When running in an application server, the Report API can be used to generate reports within servlets and JSPs and stream the generated report to the client browser. Clients can also view reports by loading the Report Viewer applet.

Report generation can be handled on demand, triggered by application processes, or scheduled. A scheduler interface is also provided with EspressReport. In order to run the scheduler, the EspressManager must also be running.

For more information about EspressReport architecture and deployment options/configurations, please see Section 1.2 - Architecture & Installation, and Section 2.5 - Servlets and Java Server Pages of the Programming Guide.

Q.2. Key Features

This section explains some of the key features and functionality that can be found in the EspressReport package.

Pure Java Architecture

Written completely in Java with no native calls, EspressReport deploys easily on most platforms including Windows, Solaris, Linux, Mac OS X, HP UX, and other Unix platforms.

Full-Featured Report Designer The Report Designer interface provides an easy-to-use GUI where users can design and layout their reports visually. All report formatting commands are available within the Designer which allows complete reports to be pre-designed and limits the amount of deployment code necessary to run reports. The Report Designer can run as an application, it can be loaded remotely through a Web browser with zero client install, and it can even be integrated within other applications and launched via the API in a number of different configurations.

Draw Data from any Source EspressReport can directly retrieve data from relational databases (JDBC), XML files, text files, and EJBs. Data from Java objects/arrays can be retrieved through a class file or passed into the report at run-time via the API.

Powerful Query Tools EspressReport provides several different query tools for reports running against relational databases. For advanced users, EspressReport offers the capability to write SQL statements with full control over the query. For intermediate users, a graphical query builder is provided which allows users to construct queries in a point and click QBE interface. For users without knowledge of the underlying database structure, EspressReport provides Data Views interface in which they can easily define queries by pre-arranged fields and functions. Data Views allows administrators to translate complicated database names/structures into a folder/field hierarchy organized according to the user preferences. These local "catalogs" allow users to easily select fields and write simple filtering conditions.

Extensive Data Visualization EspressReport provides an extensive built-in charting library. Users can easily plot report data in one of over 30 different 2D and 3D chart types. Three-dimensional charts are rendered in true 3D, allowing data series to be plotted on the Z axis, as well as pan/zoom, rotation, and light source modifications. The imbedded Chart Designer allows users to edit chart properties visually before inserting them into a report.

Flexible Deployment Model With a pure API-based deployment configuration, EspressReport easily integrates into virtually any application environment. With no proprietary servers required to deliver reports, the reporting engine can be embedded directly in an application server. This allows users to leverage the scalability of their existing platforms. With a deployment package as simple as a couple of API classes and report templates, EspressReport can be completely integrated within third party applications.

Q.3. Designer Quick Start

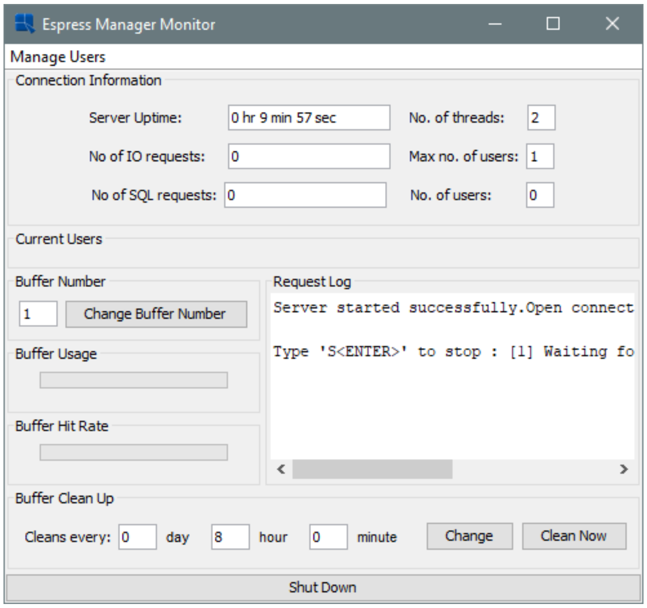
This section contains a series of short tutorials designed to illustrate some of the basic features of the Report Designer. For more details about any of the features described in this section, please see the Designer Guide portion of the documentation.

Q.3.1. Start Report Designer

This example assumes that you will be running Report Designer locally on the machine on which it is installed. For details about running it remotely, please see Section 1.2.5 - Starting Report Designer.

Before Report Designer can start, EspressManager must be running. EspressManager is the back-end component that manages data access and file I/O for Report Designer (this allows it to run both locally and remotely). To start EspressManager, execute the `EspressManager.bat` file in the root directory of your installation (`EspressManager.sh` for Unix installations). By default, the monitor will open in a new window when EspressManager starts.

"Manage Users" button can be used to add and remove username and encrypted password in the config.txt file.



Server Uptime: 0 hr 9 min 57 sec
No. of threads: 2
No of IO requests: 0
Max no. of users: 1
No of SQL requests: 0
No. of users: 0

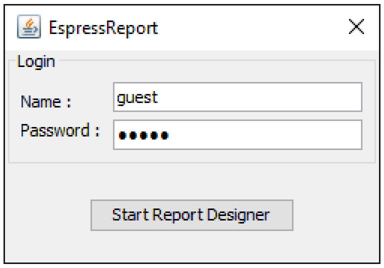
Buffer Number: 1

```
Server started successfully.Open connect

Type 'S<ENTER>' to stop : [1] Waiting fo
```

Cleans every: 0 day 8 hour 0 minute

EspressManager Monitor

With EspressManager running, you can then start Report Designer by executing the `ReportDesigner.bat` file in the root directory of your installation (`ReportDesigner.sh` for Unix installations). A dialog box will then appear prompting you to log in. To log in as the default user, use the user name "guest" with no password (see Section 1.2.3 - Configuration of the Designer Guide for more about configuring users).



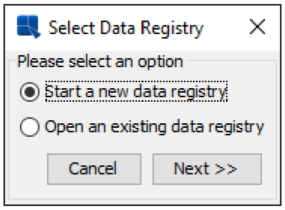
Name : guest
Password : •••••

Designer Login Window

After you enter the user name and password, click the *Start Report Designer* button and the Report Designer will open in a new window.

Q.3.2. Set Up Data Sources

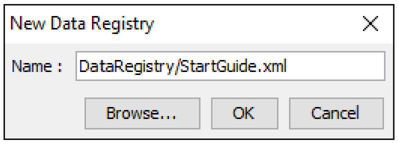
Data sources are maintained within a data registry. To create a new data registry, first select *New* from the *File* menu in the navigation bar. This will bring up a dialog prompting you to either use an existing registry or to create a new one.



Please select an option
● Start a new data registry
○ Open an existing data registry

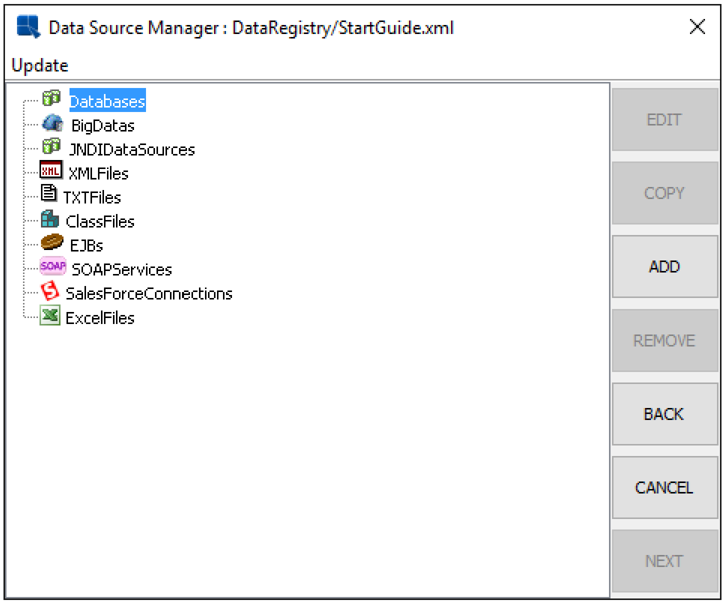
Registry Selection Dialog

Select to start a new data registry and click the *Next* button. This will open a dialog prompting you to specify a name for the registry. Enter any name you would like for the data registry and click the *Ok* button. The data registry will open in a new window.



Name : DataRegistry/StartGuide.xml

New Data Registry dialog



Data Registry Window

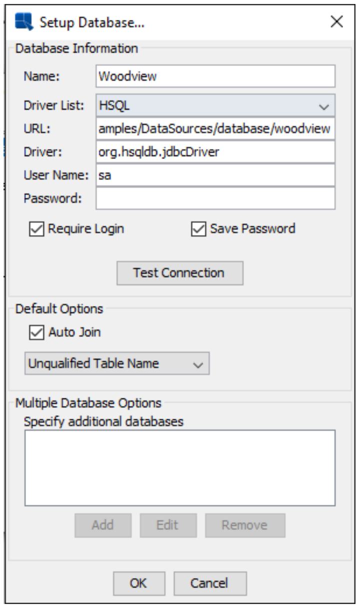
Q.3.2.1. Setup A Database Connection

EspressReport allows you to connect to JDBC compliant data sources. Examples are included within your installation guide.

Q.3.2.1.1. Setup a JDBC Connection

In this tutorial, we will set up a JDBC connection to the Woodview HSQL database that comes with the EspressReport Installation. (HSQL is an open source Java application database).

From the Data Source Manager, click on the *Databases* node in the left frame and then click the *Add* button. A dialog will appear prompting you to enter connection information for the new database. Enter **`Woodview`** as the name of the Database, select HSQL from the Driver List, enter **`jdbc:hsqldb:help/examples/DataSources/database/woodview`** for the URL and enter **`org.hsqldb.jdbcDriver`** as the driver. Click on both the *Require Login* and *Save Password* boxes. Then enter **`sa`** for the username and leave the password blank.



Setup Database...

Database Information

Name: Woodview

Driver List: HSQL

URL: amples/DataSources/database/woodview

Driver: org.hsqldb.jdbcDriver

User Name: sa

Password:

☑ Require Login ☑ Save Password

Default Options

☑ Auto Join

Unqualified Table Name

Multiple Database Options

Specify additional databases

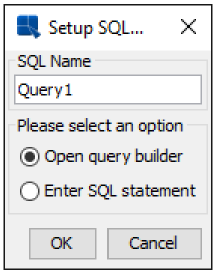
Add Database Dialog

Leave the *Auto Join* and table name properties alone and click the *Test Connection* button to make sure you have entered the information correctly. Then click the *OK* button to add this database connection to the Data Source Manager window, where there will be a new node under "Databases" for Woodview.

Q.3.2.2. Create a Query

EspressReport provides a number of different interfaces to query a database to retrieve the report data. You can execute an SQL statement, use the Query Builder, or use data views to create a query interface that insulates the end user from the database structure. In this example, we will use the Query Builder to create a query.

To create a new query, click on the *Woodview* node in the left frame of the Data Source Manager to expand it. Two sub-nodes will appear, one called *Queries* and one called *Data Views*. Select the *Queries* node and click the *Add* button. A dialog will appear prompting you to specify a name for the query and to select whether to launch the Query Builder or to enter an SQL statement.



Setup SQL...

SQL Name

Query1

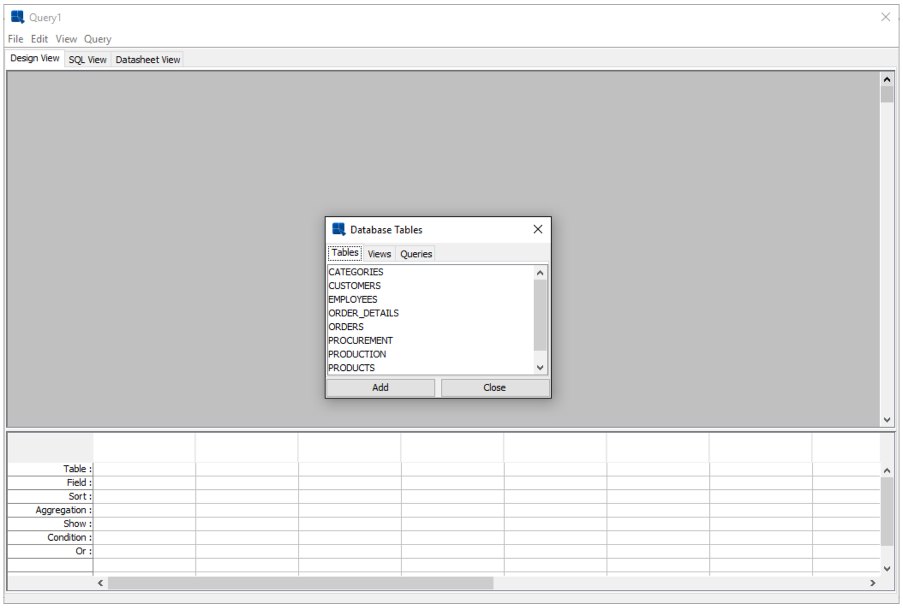
Please select an option

● Open query builder

○ Enter SQL statement

Query Name Dialog

Enter any name you would like, select *Open Query Builder* and click on the *Ok* button. The Query Builder will launch. You will see a separate window containing all of the tables for Woodview sitting over the top of the main Query Builder window.

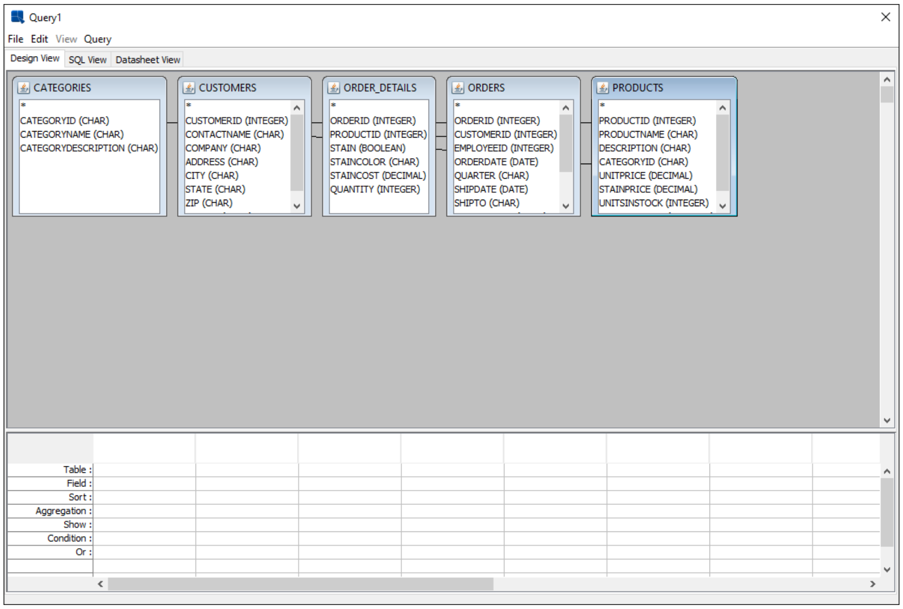


Query Builder Dialog

To add a table to the query, select a table in the Tables window and click the *Add* button. You can also double click on the table name. Using one of the two methods, add the following tables to the query:

```
CATEGORIES
CUSTOMERS
ORDER_DETAILS
ORDERS
PRODUCTS
```

When you are done, click the *Close* button to close the *Database Tables* window. The tables will appear in the top half of the Query Builder window. You will see join lines connecting various fields in the tables.



Query Builder with Tables

To add a field to the query, you can double click on the field in the table window. You can also double click on the *Table* and *Field* fields in the lower (QBE) portion of the Query Builder window and then select the table and field from the drop-down menus. Using either method, add the following fields to the query:

ORDERID (INTEGER) from *ORDERS*

COMPANY (CHAR) from *CUSTOMERS*

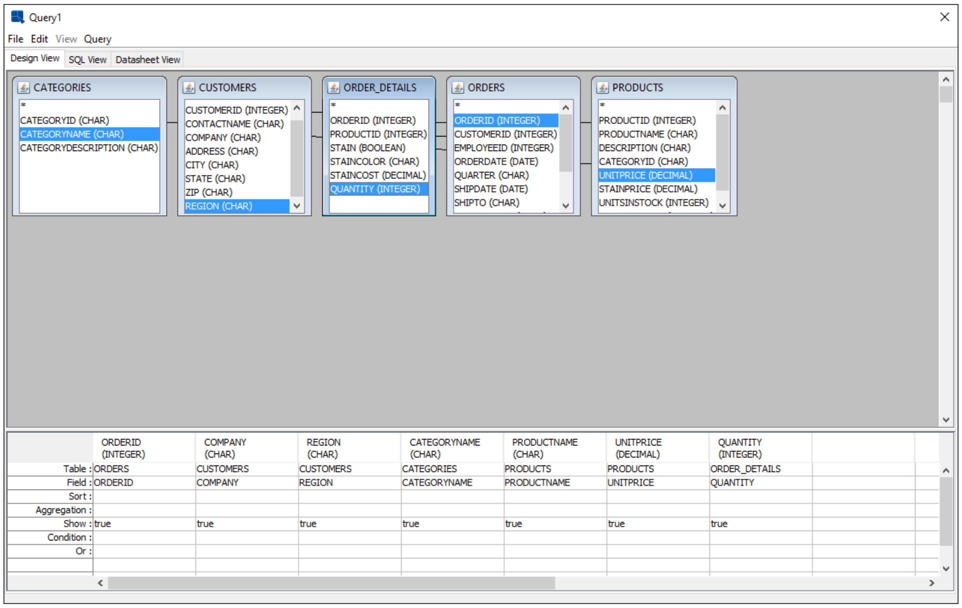
REGION (CHAR) from *CUSTOMERS*

CATEGORYNAME (CHAR) from *CATEGORIES*

PRODUCTNAME (CHAR) from *PRODUCTS*

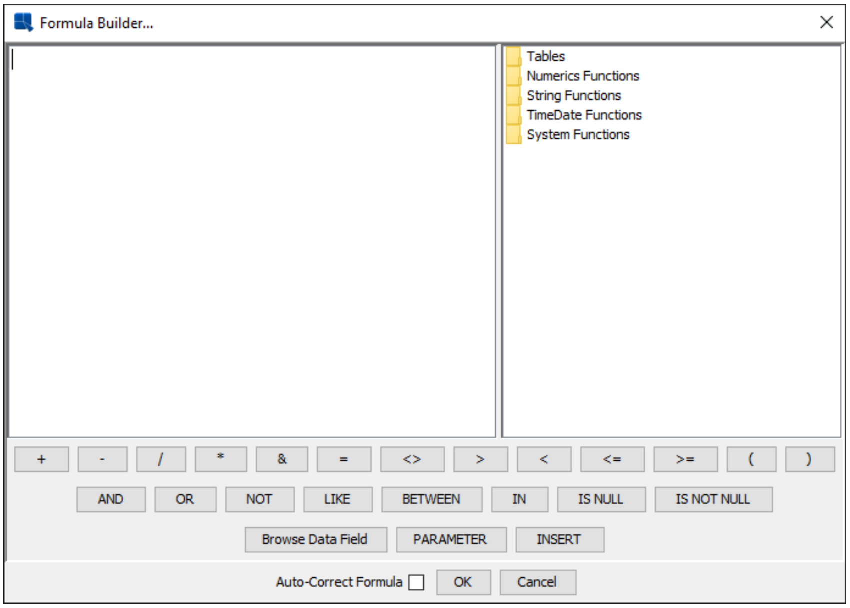
UNITPRICE (DECIMAL) from *PRODUCTS*

QUANTITY (INTEGER) from *ORDER_DETAILS*



	ORDERID (INTEGER)	COMPANY (CHAR)	REGION (CHAR)	CATEGORYNAME (CHAR)	PRODUCTNAME (CHAR)	UNITPRICE (DECIMAL)	QUANTITY (INTEGER)	
Table :	ORDERS	CUSTOMERS	CUSTOMERS	CATEGORIES	PRODUCTS	PRODUCTS	ORDER_DETAILS	
Field :	ORDERID	COMPANY	REGION	CATEGORYNAME	PRODUCTNAME	UNITPRICE	QUANTITY	
Sort :								
Aggregation :								
Show :	true	true	true	true	true	true	true	
Condition :								
Or :								

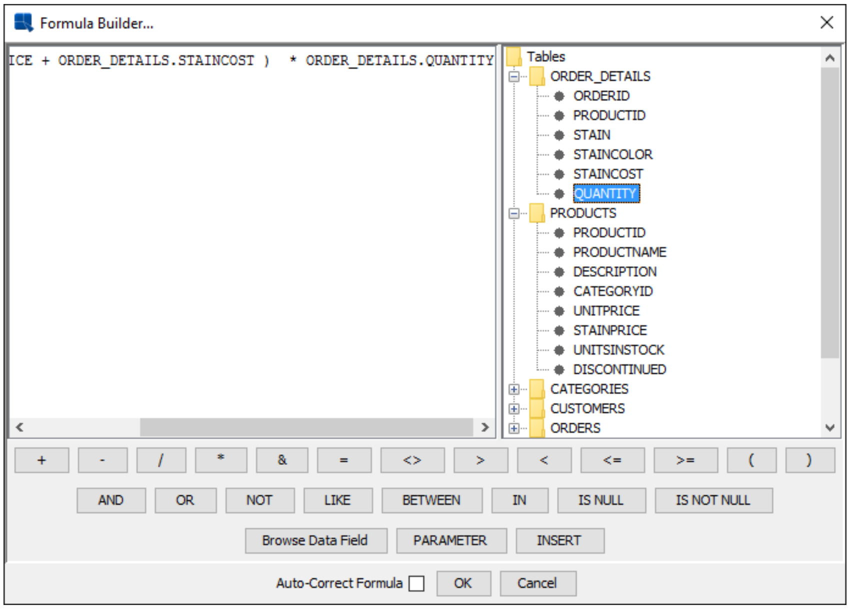
In the eighth column, which should be blank, right click in the *Field* field and select *Build* from the pop-up menu. This will open the Formula Builder interface allowing you to create a computed column.



Formula Builder Window

First, click the *left parenthesis* button to build a column. Then double click on the *Tables* folder to expand it into five nodes, one for each of the tables that you selected for the query. Opening a table folder will list all of the column fields for that table. Open the *PRODUCTS* folder, select *UNITPRICE* and click the *INSERT* button. Then click the add (+) button. Next, insert *STAINCOST* from the *ORDER_DETAILS* table. Then click the *right parenthesis* button followed by the multiply (*) button. Finally, insert *QUANTITY* from the *ORDER_DETAILS* table. The finished formula should look like this:

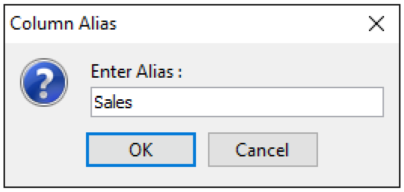
```
( PRODUCTS.UNITPRICE + ORDER_DETAILS.STAINCOST ) * ORDER_DETAILS.QUANTITY
```



```
ICE + ORDER_DETAILS.STAINCOST )  * ORDER_DETAILS.QUANTITY
```

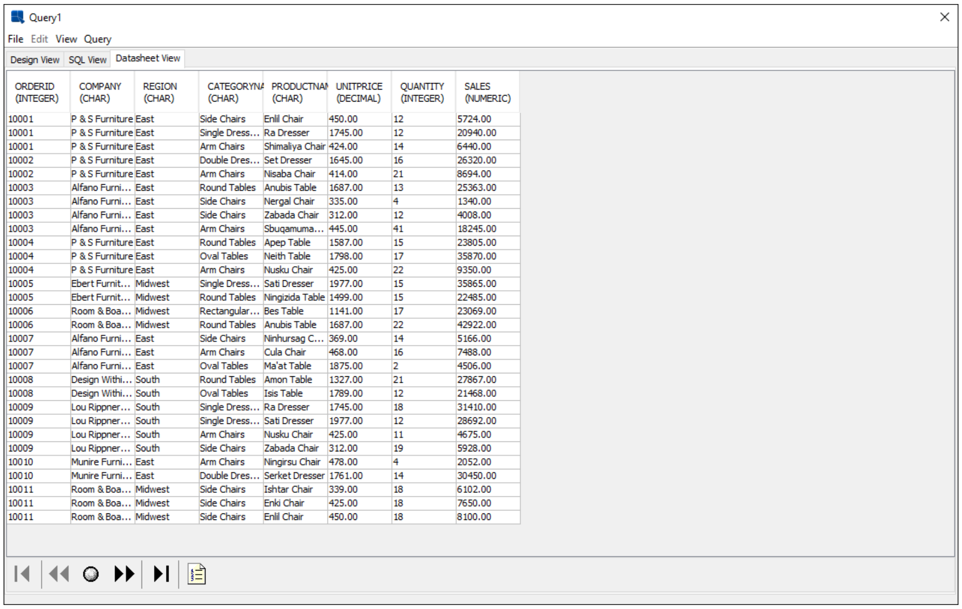
Formula Builder Window with Formula

Click the *Ok* button and the built column will be added to the query. Next, we will give the column an alias. Right click on the column and select *Alias* from the pop-up menu. A window will appear asking you to enter a column alias. Enter "SALES" (without quotation marks) and click the *OK* button.



Column Alias Dialog

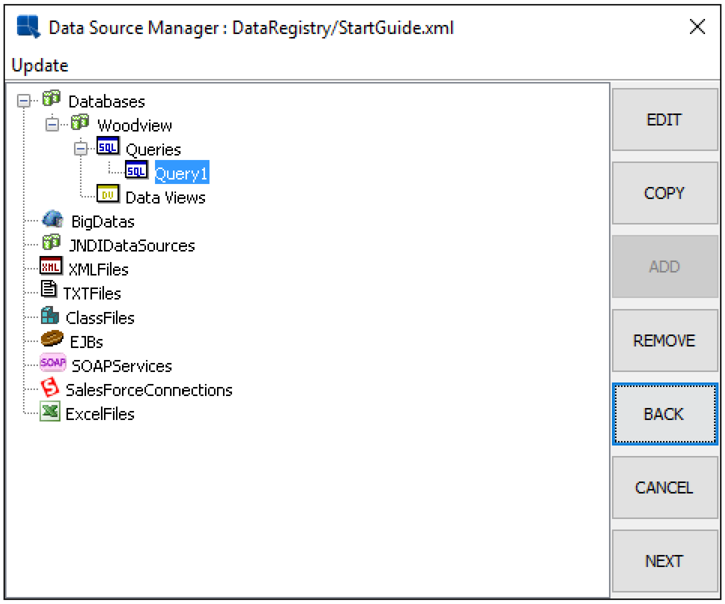
Click the *Ok* button and you will see the column name change in the Query Builder. Now click on the *Datasheet View* tab in the Query Builder. Your query will run and you should see the first thirty records of the query results.



ORDERID (INTEGER)	COMPANY (CHAR)	REGION (CHAR)	CATEGORYNA (CHAR)	PRODUCTNAI (CHAR)	UNITPRICE (DECIMAL)	QUANTITY (INTEGER)	SALES (NUMERIC)
10001	P & S Furniture	East	Side Chairs	Enlil Chair	450.00	12	5724.00
10001	P & S Furniture	East	Single Dress...	Ra Dresser	1745.00	12	20940.00
10001	P & S Furniture	East	Arm Chairs	Shimaliya Chair	424.00	14	6440.00
10002	P & S Furniture	East	Double Dres...	Set Dresser	1645.00	16	26320.00
10002	P & S Furniture	East	Arm Chairs	Nisaba Chair	414.00	21	8694.00
10003	Alfano Furni...	East	Round Tables	Anubis Table	1687.00	13	25363.00
10003	Alfano Furni...	East	Side Chairs	Nergal Chair	335.00	4	1340.00
10003	Alfano Furni...	East	Side Chairs	Zabada Chair	312.00	12	4008.00
10003	Alfano Furni...	East	Arm Chairs	Sbuqamuma...	445.00	41	18245.00
10004	P & S Furniture	East	Round Tables	Apep Table	1587.00	15	23805.00
10004	P & S Furniture	East	Oval Tables	Neith Table	1798.00	17	35870.00
10004	P & S Furniture	East	Arm Chairs	Nusku Chair	425.00	22	9350.00
10005	Ebert Furnit...	Midwest	Single Dress...	Sati Dresser	1977.00	15	35865.00
10005	Ebert Furnit...	Midwest	Round Tables	Ningizida Table	1499.00	15	22485.00
10006	Room & Boa...	Midwest	Rectangular...	Bes Table	1141.00	17	23069.00
10006	Room & Boa...	Midwest	Round Tables	Anubis Table	1687.00	22	42922.00
10007	Alfano Furni...	East	Side Chairs	Ninhursag C...	369.00	14	5166.00
10007	Alfano Furni...	East	Arm Chairs	Cula Chair	468.00	16	7488.00
10007	Alfano Furni...	East	Oval Tables	Ma'at Table	1875.00	2	4506.00
10008	Design Withi...	South	Round Tables	Amon Table	1327.00	21	27867.00
10008	Design Withi...	South	Oval Tables	Isis Table	1789.00	12	21468.00
10009	Lou Rippner...	South	Single Dress...	Ra Dresser	1745.00	18	31410.00
10009	Lou Rippner...	South	Single Dress...	Sati Dresser	1977.00	12	28692.00
10009	Lou Rippner...	South	Arm Chairs	Nusku Chair	425.00	11	4675.00
10009	Lou Rippner...	South	Side Chairs	Zabada Chair	312.00	19	5928.00
10010	Munire Furni...	East	Arm Chairs	Ningirsu Chair	478.00	4	2052.00
10010	Munire Furni...	East	Double Dres...	Serket Dresser	1761.00	14	30450.00
10011	Room & Boa...	Midwest	Side Chairs	Ishtar Chair	339.00	18	6102.00
10011	Room & Boa...	Midwest	Side Chairs	Enki Chair	425.00	18	7650.00
10011	Room & Boa...	Midwest	Side Chairs	Enlil Chair	450.00	18	8100.00

Query Builder Datasheet View

Now that you have finished designing the query, select *Done* from the File menu to save the changes. This will close the Query Builder window and return you to the Data Registry Manager window. There will now be a node under "Queries" for the query you have just designed.



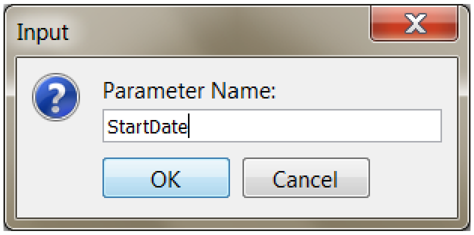
Data Source Manager With Query

Q.3.2.2.1. Add Query Parameters

EspressReport allows you to easily parameterize report queries, allowing report data to be dynamically filtered at run-time. In this tutorial we will add parameters to the query created in Section Q.3.2.2 - Create a Query

To open the query you created, select it and click the *Edit* button in the Data Source Manager. Then click the *OK* button to confirm the name. Your query will re-open in the query builder. The Tables window will open on top of the Query Builder. Click the *Close* button to close the Database Tables window and right click in the *Condition* field under the ORDERID column in the lower part of the Query Builder and select *Build* from the pop-up menu. This will bring up the Formula Builder which will allow you to construct a condition for the query.

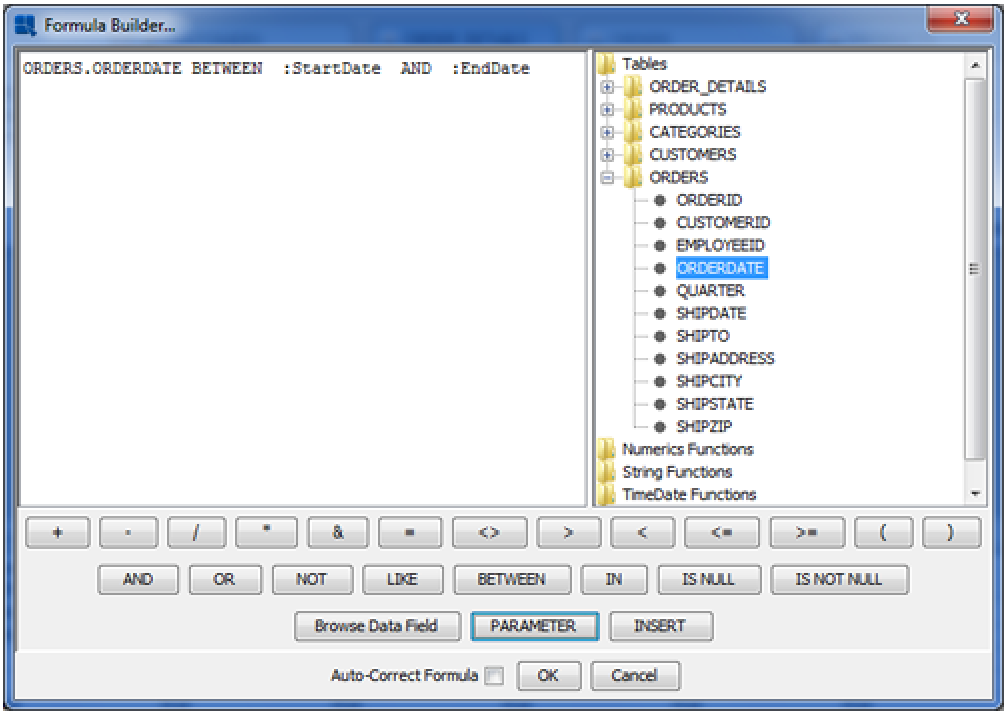
Within the Formula Builder, double click on the *Tables* folder to expand it. Then expand the *ORDERS* node and double click on the *ORDERDATE* field. Next, click the *BETWEEN* button and then click the *PARAMETER* button. This will bring up a dialog prompting you specify a name for the query parameter.



Parameter Name Dialog

Enter **`StartDate`** as the parameter name and click the *OK* button. The parameter will be added to the query. Then click the *AND* button followed by the *PARAMETER* button again. Enter **`EndDate`** as the second parameter name. The finished condition should look like this

```
ORDERS.ORDERDATE BETWEEN :StartDate AND :EndDate
```

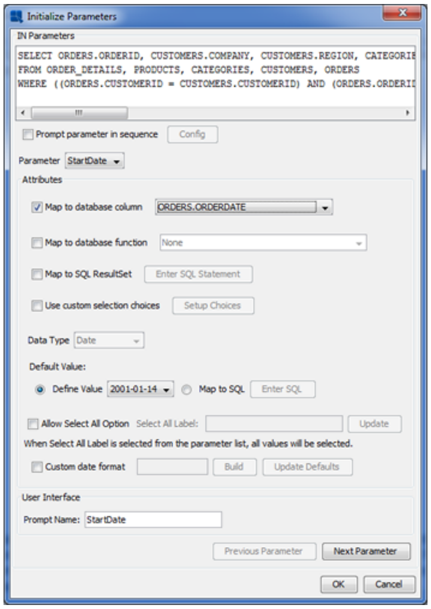


```
ORDERS.ORDERDATE BETWEEN  :StartDate  AND  :EndDate
```

Auto-Correct Formula □

Formula Builder with Conditions

Click the *Ok* button to close the Formula Builder and to return to the Query Builder window. Now click on the *Datasheet View* tab. Because you have just added two parameters to the query, an initialization dialog will appear, asking you to specify a few properties for the query parameters.



```
SELECT ORDERS.ORDERID, CUSTOMERS.COMPANY, CUSTOMERS.REGION, CATEGORIE
FROM ORDER_DETAILS, PRODUCTS, CATEGORIES, CUSTOMERS, ORDERS
WHERE ((ORDERS.CUSTOMERID = CUSTOMERS.CUSTOMERID) AND (ORDERS.ORDERII
```

□ Prompt parameter in sequence

Parameter: StartDate

☑ Map to database column: ORDERS.ORDERDATE

□ Map to database function: None

□ Map to SQL ResultSet

□ Use custom selection choices

Data Type: Date

Default Value:

● Define Value: 2001-01-14 ○ Map to SQL

□ Allow Select All Option Select All Label:

When Select All Label is selected from the parameter list, all values will be selected.

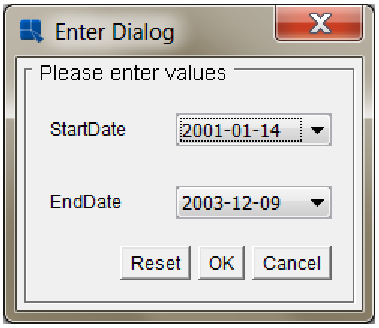
□ Custom date format

Prompt Name: StartDate

Parameter Initialization Dialog

From this window, click on *Map to a database column* and select *ORDERS.ORDERDATE* from the drop-down menu. This will automatically fill the *Default Value* and *Data Type* options. Next, enter **`Start Date`** into the *Prompt Name*, then click the *Next Parameter* button and map the *EndDate* parameter to the same column. Click on the *Define Value* drop-down menu to select an end date. Select a date far enough from the start date that by default you will have more than a couple records to work with (this makes report design easier). Change its *Prompt Name* to **`End Date`**.

Click the *Ok* button to close the initialization window once you have specified all the options. A new dialog will appear prompting you to select a date range by which to filter the result set.



StartDate 2001-01-14

EndDate 2003-12-09

Parameter Selection Dialog

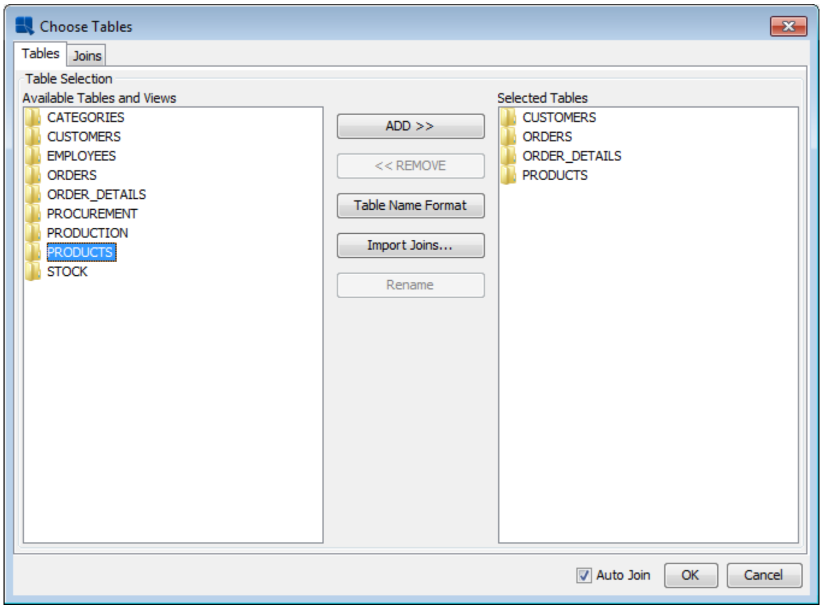
Select the Start and End date that you would like and click the *OK* button. You will now see the filtered result in the datasheet window. Now, click *Done* from the File menu to save the changes you have made to the query.

Q.3.2.3. Create a Data View

A unique feature in EspressReport is an ability to create data views. Data views are local schemas/views that allow an administrator to pre-configure a group of tables and fields. This allows end users to only select fields and define simple conditions to create a query. To create a data view, select the *Data Views* node under *Woodview* and click *Add*.

This will open a new dialog asking you to select database tables you want to use. Select the following tables and click the *ADD>>* button to add them to the *Selected Tables* window:

```
CUSTOMERS
ORDERS
ORDER_DETAILS
PRODUCTS
```



Available Tables and Views
- CATEGORIES
- CUSTOMERS
- EMPLOYEES
- ORDERS
- ORDER_DETAILS
- PROCUREMENT
- PRODUCTION
- PRODUCTS
- STOCK

Selected Tables
- CUSTOMERS
- ORDERS
- ORDER_DETAILS
- PRODUCTS

☑ Auto Join

Data View Tables Dialog

Next, click on the *Joins* tab. You will see a representation of the tables like in the Query Builder. You can see the auto-join lines between the tables. This window can be used to join the tables or to modify the auto-joins, if necessary. Click the *Ok* button to finalize the table selection. The next window allows you to select and group fields for the view. At the top of the window you can specify a name for the view. Name it **`Invoicing`**.

Next, double click on the *CUSTOMERS* folder to reveal the fields for that table. Add the following fields by selecting them and clicking the *ADD>>* button:

```
CONTACTNAME
COMPANY
ADDRESS
CITY
STATE
ZIP
```

Now add fields from other tables as follows:

ORDERS:

```
ORDERDATE
SHIPDATE
SHIPTO
SHIPADDRESS
SHIPCITY
SHIPSTATE
SHIPZIP
```

ORDER DETAILS:

```
ORDERID
STAIN
STAINCOLOR
QUANTITY
```

PRODUCTS:

```
PRODUCTNAME
UNITPRICE
STAINPRICE
```

Now click the *Add Heading* button. At the prompt, specify the name **`Customer Info`**. Add two more headings in the same way, one called **`Shipping Info`** and one called **`Order Info`**. Once they are created, select the following fields (Using **CTRL**+**Click** or **SHIFT**+**Click** for multiple selection):

```
CONTACTNAME
COMPANY
ADDRESS
CITY
STATE
ZIP
```

Once the fields are selected, click the *Group Fields* button and select *Customer Info* from the drop-down list. The fields will be moved under that heading. Next, select the following fields in the same way:

```
SHIPDATE
SHIPTO
SHIPADDRESS
SHIPCITY
SHIPSTATE
```

```
SHIPZIP
```

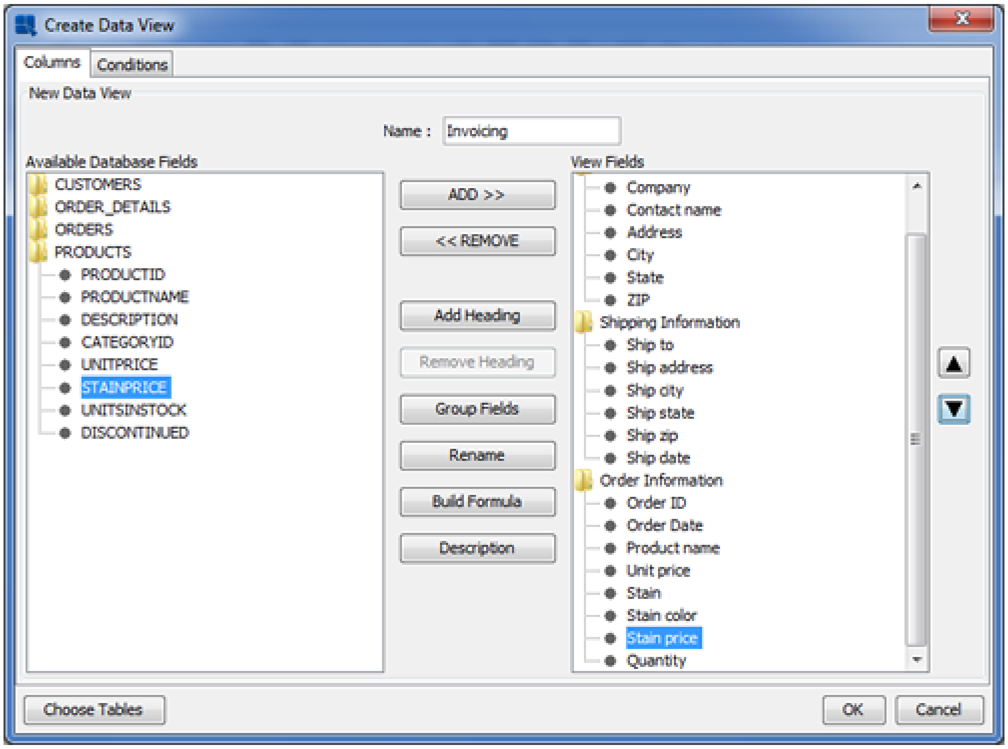
Add these fields to the *Shipping Info* group the same way you did before. Next, select the following fields:

```
ORDERDATE
ORDERID
STAIN
STAINCOLOR
QUANTITY
PRODUCTNAME
UNITPRICE
STAINPRICE
```

Add these fields to the *Order Info Info* group. Next, select the *CONTACTNAME* field on the right side and click the *Rename* button. In the dialog, specify the name **`Contact name`**. Repeat this for every field in order to give it proper names.

Next, select the *Order ID* field on the right side and click the *up arrow* button to move the field to the top of the "Order Info" heading. Use the arrows to arrange the items in the "Order Info" heading in the following order:

```
Order ID
Order Date
Product name
Unit Price
Stain
Stain Color
Stain Price
Quantity
```



Name : Invoicing

Data View Fields Window

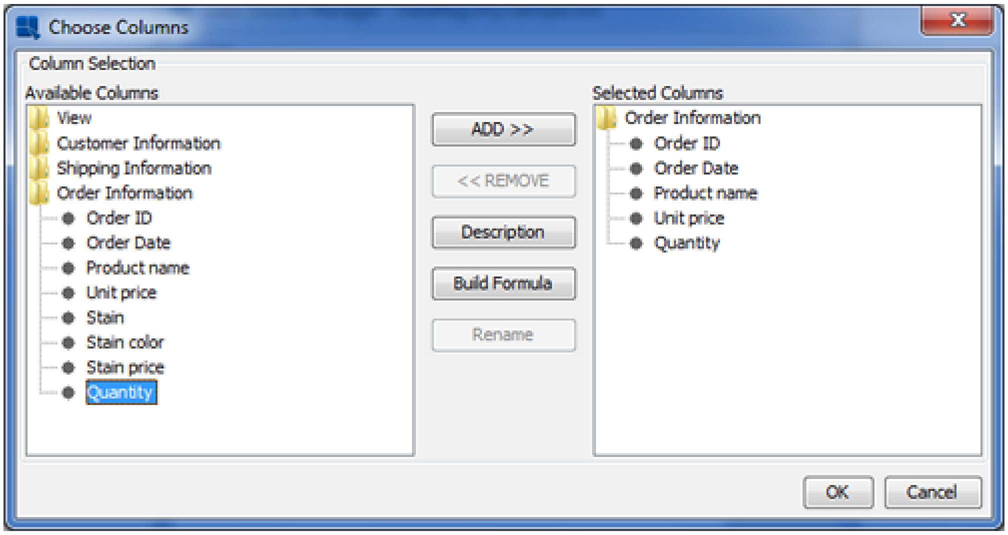
Now click the *OK* button in the fields window to save the view. It will be saved as a new node under *Data Views* in the Data Source Manager.

Q.3.2.3.1. Query a Data View

Now that the data view has been created, you can write queries against the view. This allows users to develop queries without knowing the underlying structure of the database. It also allows administrators to limit which database elements users has access to. In this tutorial we will create a query for the data view you created in Section Q.3.2.3 - Create a Data View

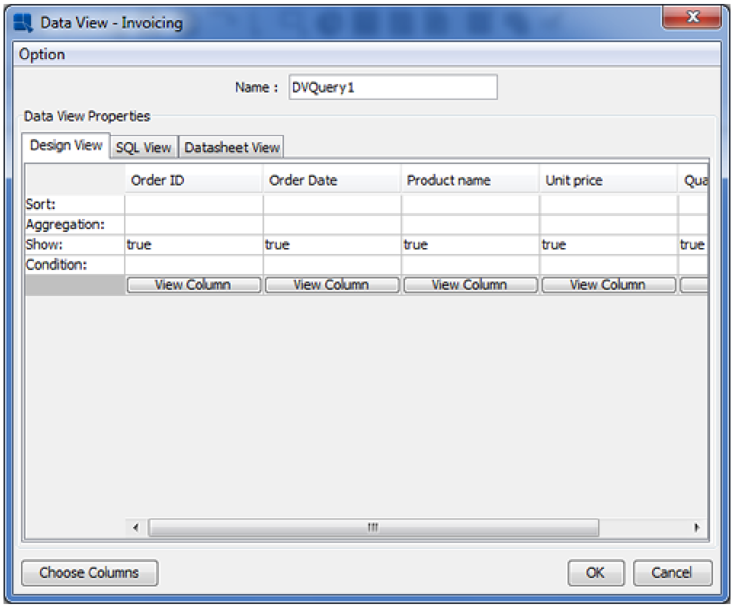
In the Data Source Manager, select the *Invoicing* data view. Then click the *Next* button. This will open a dialog prompting you to select fields from the view. To select fields, first double click on a heading to expand it. Add the following fields to the query by selecting them and clicking the *Add* button:

```
Order ID
Order Date
Product name
Unit Price
Quantity
```



Data View Query Field Selection Dialog

Once you finish adding the fields, click the *OK* button. This will bring up a new window which enables you to set conditions, grouping, and ordering for the query. Like the Query Builder, this window also allows you to preview the query result with the *Datasheet View* tab.

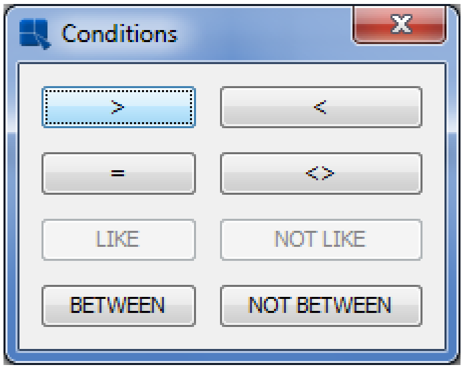


Name : DVQuery1

	Order ID	Order Date	Product name	Unit price	Qua
Sort:					
Aggregation:					
Show:	true	true	true	true	true
Condition:					
	View Column	View Column	View Column	View Column	

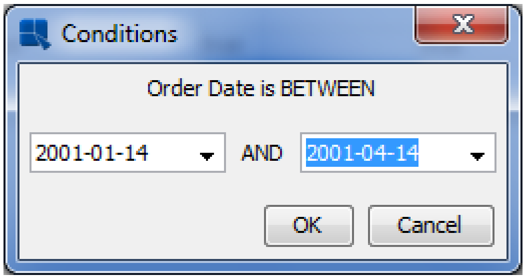
Data View Conditions Dialog

First, specify a name for the query in the space provided at the top. Then double click on the *Condition* field for the Order Date column. This will bring up a dialog allowing you to specify a condition for the field.



Specify Condition Dialog

Click the *Between* button. A new dialog will appear prompting you to specify a start and end date with which to filter the results. Select **`2001-01-14`** as the first date, and **`2003-12-09`** as the second.



Order Date is BETWEEN

2001-01-14 AND 2001-04-14

Specify Condition Fields Dialog

Click *OK* to close the dialog and add the condition. You will be taken back to the conditions window. Now you can click on the *Datasheet View* tab to preview the query.

Click *OK* in the main window and the query will be saved using the name you provided. You will then be taken to the first step in the Report Wizard. Click *Cancel* in this dialog to close the interface and then return to the Data

Source Manager by clicking the *New File* button. There will now be a new node for your query under the *Invoicing* data view.

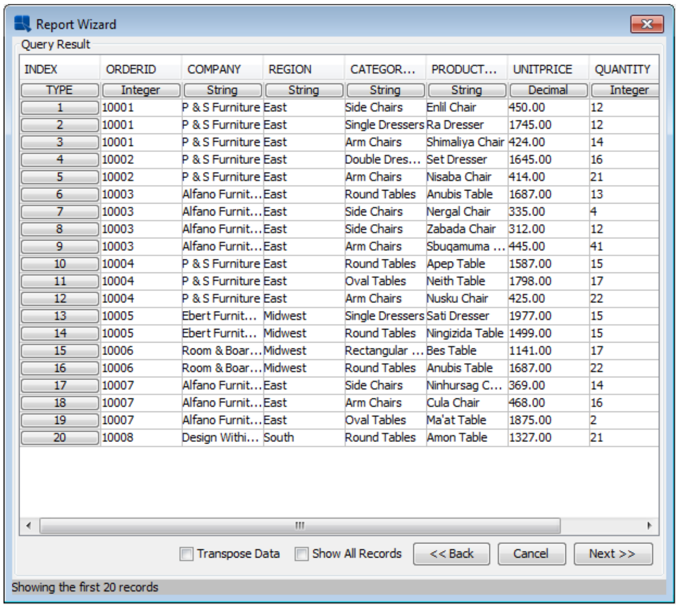
Q.3.3. Report Mapping

This section looks at the different ways data from the data source can be mapped into a report structure. In this section, we will take the result set from the query created in Section Q.3.2.2 - Create a Query and use it to generate reports in different layouts supported by EspressReport. For detailed information about all the different layout and mapping options, see Section 1.4 - Report Types and Data Mapping of the Designer Guide.

Q.3.3.1. Simple Columnar Layout

The simple columnar layout is the most straight-forward type of mapping. Columns from the data source are drawn as a straight table in the report, without any grouping or breaks.

To begin mapping a report, first select the query you created before in *Queries* node in the Data Source Manager and click the *Next* button. A new window will open with a table containing the results of the query (the first 20 records only). Notice that because your query contains a parameter, it initially runs with the default values that were specified when the parameters were initialized.

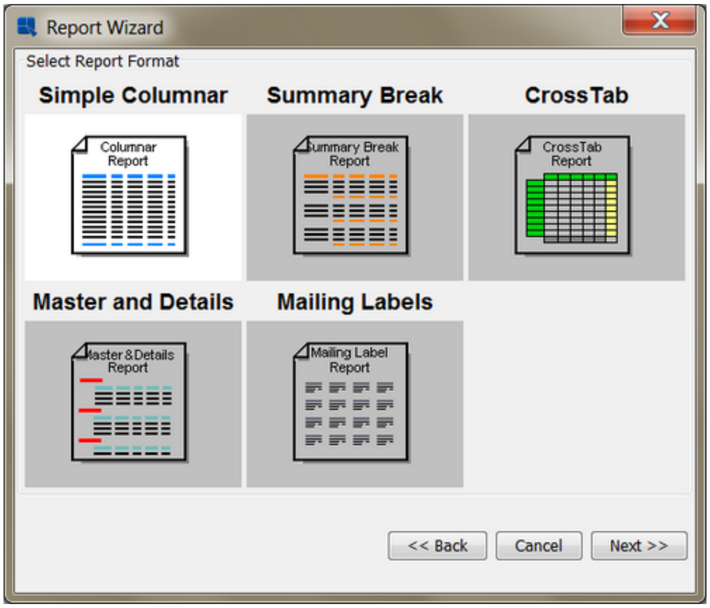


INDEX	ORDERID	COMPANY	REGION	CATEGOR...	PRODUCT...	UNITPRICE	QUANTITY
TYPE	Integer	String	String	String	String	Decimal	Integer
1	10001	P & S Furniture	East	Side Chairs	Enlil Chair	450.00	12
2	10001	P & S Furniture	East	Single Dressers	Ra Dresser	1745.00	12
3	10001	P & S Furniture	East	Arm Chairs	Shimaliya Chair	424.00	14
4	10002	P & S Furniture	East	Double Dres...	Set Dresser	1645.00	16
5	10002	P & S Furniture	East	Arm Chairs	Nisaba Chair	414.00	21
6	10003	Alfano Furnit...	East	Round Tables	Anubis Table	1687.00	13
7	10003	Alfano Furnit...	East	Side Chairs	Nergal Chair	335.00	4
8	10003	Alfano Furnit...	East	Side Chairs	Zabada Chair	312.00	12
9	10003	Alfano Furnit...	East	Arm Chairs	Sbuqamuma ...	445.00	41
10	10004	P & S Furniture	East	Round Tables	Apep Table	1587.00	15
11	10004	P & S Furniture	East	Oval Tables	Neith Table	1798.00	17
12	10004	P & S Furniture	East	Arm Chairs	Nusku Chair	425.00	22
13	10005	Ebert Furnit...	Midwest	Single Dressers	Sati Dresser	1977.00	15
14	10005	Ebert Furnit...	Midwest	Round Tables	Ningizida Table	1499.00	15
15	10006	Room & Boar...	Midwest	Rectangular ...	Bes Table	1141.00	17
16	10006	Room & Boar...	Midwest	Round Tables	Anubis Table	1687.00	22
17	10007	Alfano Furnit...	East	Side Chairs	Ninhursag C...	369.00	14
18	10007	Alfano Furnit...	East	Arm Chairs	Cula Chair	468.00	16
19	10007	Alfano Furnit...	East	Oval Tables	Ma'at Table	1875.00	2
20	10008	Design Withi...	South	Round Tables	Amon Table	1327.00	21

Showing the first 20 records

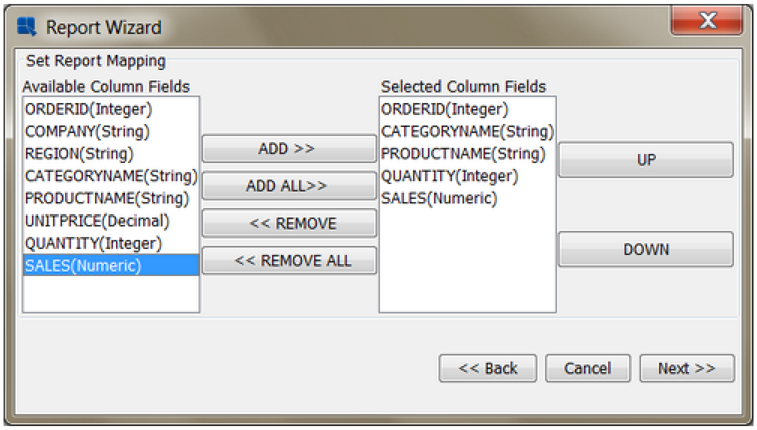
Query Result Screen

To continue with the Report Wizard, click the *Next* button. This will bring up a dialog asking you which report layout option you want to use.



Select Report Layout Dialog

From this dialog, select *Simple Columnar* as the layout and click the *Next* button. The next step in the Report Wizard allows you to select which columns from the query you want to use in the report, as well as re-arrange the column order. (Although the column order is not particularly important for the simple columnar layout, it can have a significant impact on other layouts depending on the mapping options selected).



Available Column Fields
ORDERID(Integer)
COMPANY(String)
REGION(String)
CATEGORYNAME(String)
PRODUCTNAME(String)
UNITPRICE(Decimal)
QUANTITY(Integer)
SALES(Numeric)

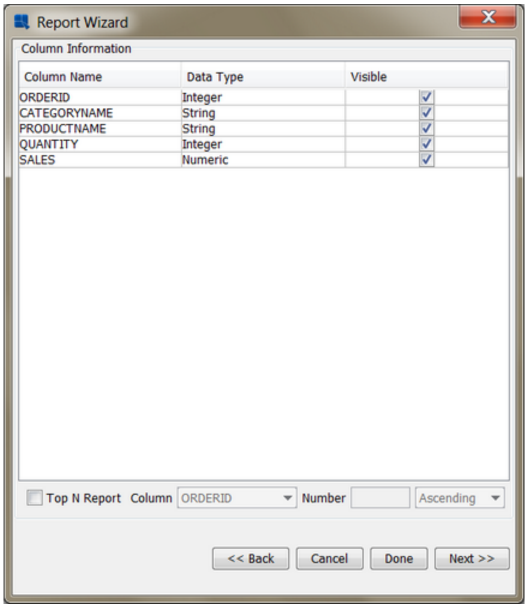
Selected Column Fields
ORDERID(Integer)
CATEGORYNAME(String)
PRODUCTNAME(String)
QUANTITY(Integer)
SALES(Numeric)

Column Selection/Ordering Dialog

In this dialog, select the following fields for the report:

```
ORDERID
CATEGORYNAME
PRODUCTNAME
QUANTITY
SALES
```

Once you select the fields, click the *Next* button to continue on with the Wizard. The next dialog is the data mapping dialog. This is where you can select how to map fields from the data source into the selected report layout. Because this is a simple columnar layout, the only options are whether to set columns visible or not, or to generate a top N presentation.

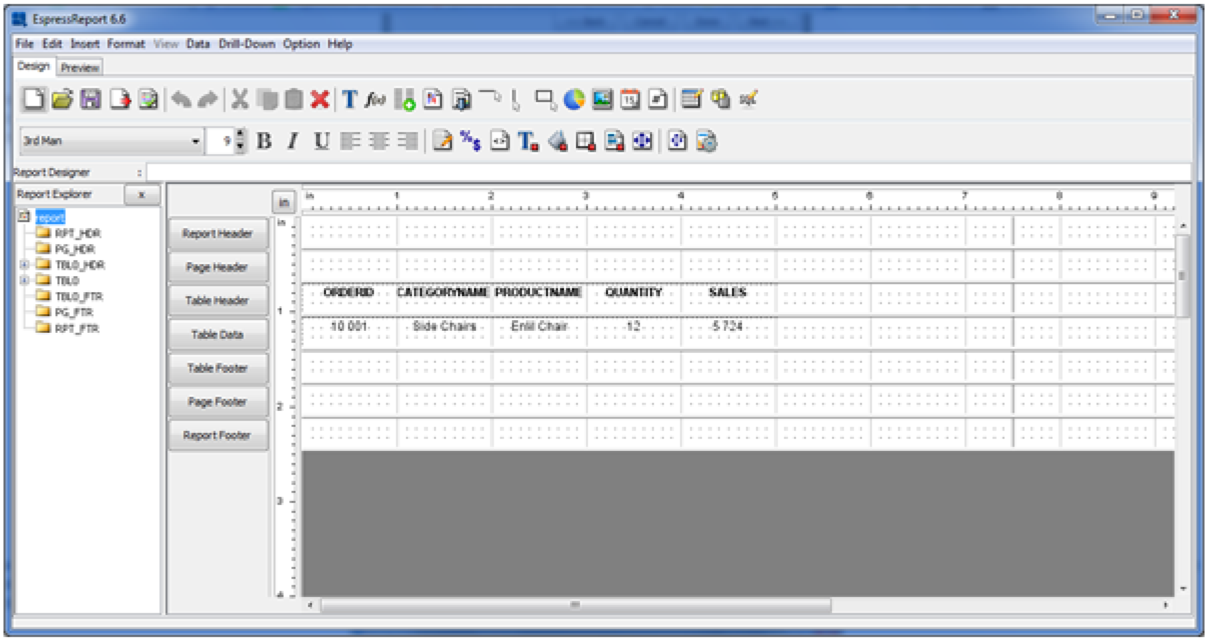


Column Name	Data Type	Visible
ORDERID	Integer	☑
CATEGORYNAME	String	☑
PRODUCTNAME	String	☑
QUANTITY	Integer	☑
SALES	Numeric	☑

□ Top N Report Column ORDERID Number Ascending

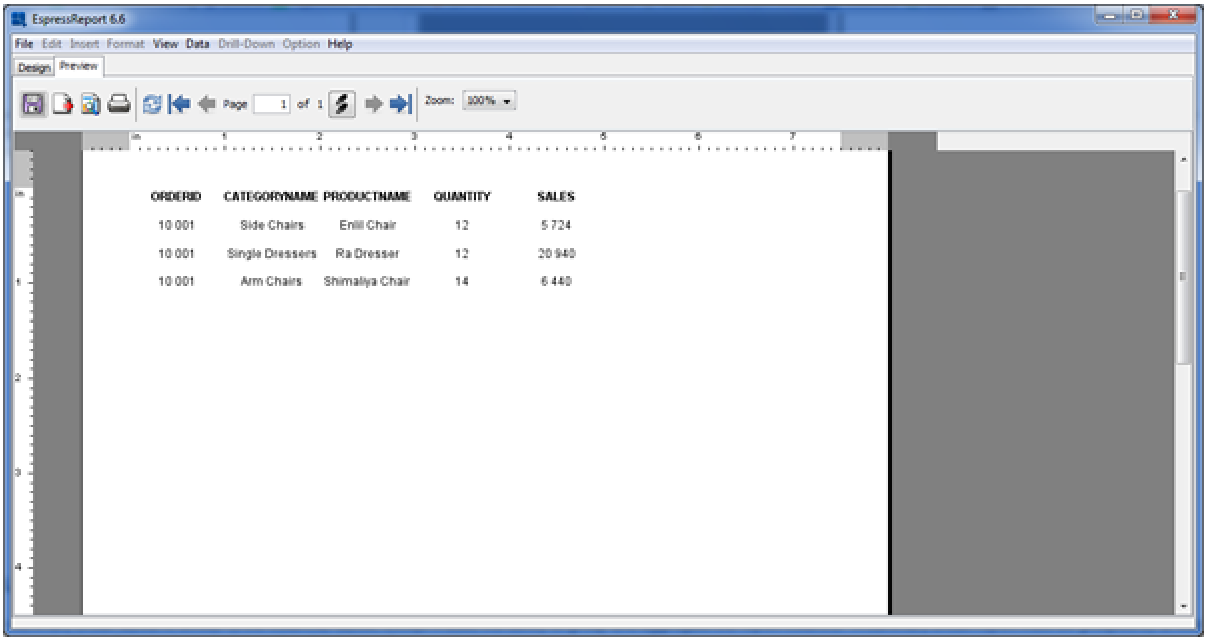
Data Mapping Dialog

In this dialog, select to leave all the columns visible and do not select the top N option. (For more about top N presentations, please see Section 1.4.1.1.1 - Top N Report of the Designer Guide). Next, click the *Done* button. (There are some optional additional steps in the Wizard which will be explained later in this section). This will bring up the Report Designer interface with an unformatted version of your report.



Simple Columnar Report in Design Window

Now click on the *Preview* Tab in the upper-left corner of the window and select *Use Live Data*. A parameter selection dialog will now appear prompting you to select a start and end date by which to filter the report. Specify a large enough range that you will get more than a couple of records and click the *OK* button. You will then see the report output. Notice how the simple columnar layout places the columns directly into the report, without any grouping, sorting, or summaries.



ORDERID	CATEGORYNAME	PRODUCTNAME	QUANTITY	SALES
10 001	Side Chairs	Enlil Chair	12	5 724
10 001	Single Dressers	Ra Dresser	12	20 940
10 001	Arm Chairs	Shimaliya Chair	14	6 440

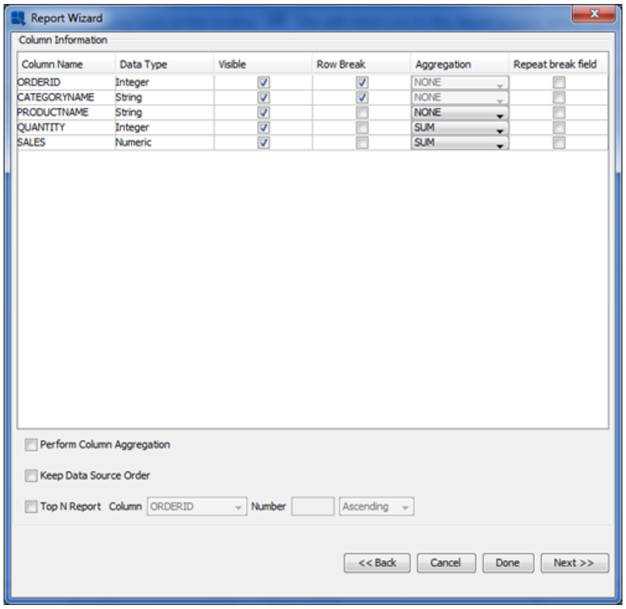
Simple Columnar Report Preview

Q.3.3.2. Summary Break Layout

The summary break layout is similar to the columnar layout, except it adds the ability to group and aggregate the report columns. A summary break report must be grouped by at least one column. Using the report created in Section Q.3.3.1 - Simple Columnar Layout, we will convert it into a summary break layout.

Go to the design tab and select the *Change Data Mapping* icon on the toolbar: . This will return you to the Report Wizard, where we will select a different report layout. From this dialog, hit the *Back* button twice until you get back to the *Select Report Format* window.

Change the report layout type to *Summary Break* and click the *Next* button. On the next screen, keep the same column selection and click the *Next* button again to go to the data mapping window. You can notice there are more options in this window than for the columnar layout.



Column Name	Data Type	Visible	Row Break	Aggregation	Repeat break field
ORDERID	Integer	☑	☑	NONE	□
CATEGORYNAME	String	☑	☑	NONE	□
PRODUCTNAME	String	☑	□	NONE	□
QUANTITY	Integer	☑	□	SUM	□
SALES	Numeric	☑	□	SUM	□

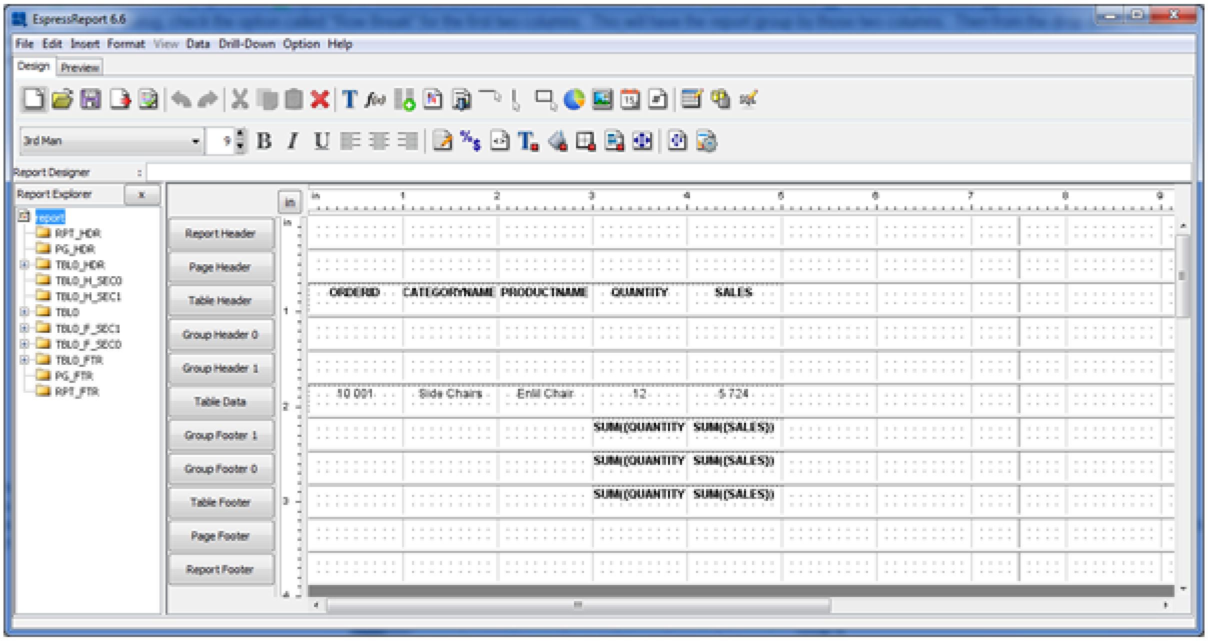
□ Perform Column Aggregation

□ Keep Data Source Order

□ Top N Report Column ORDERID Number Ascending

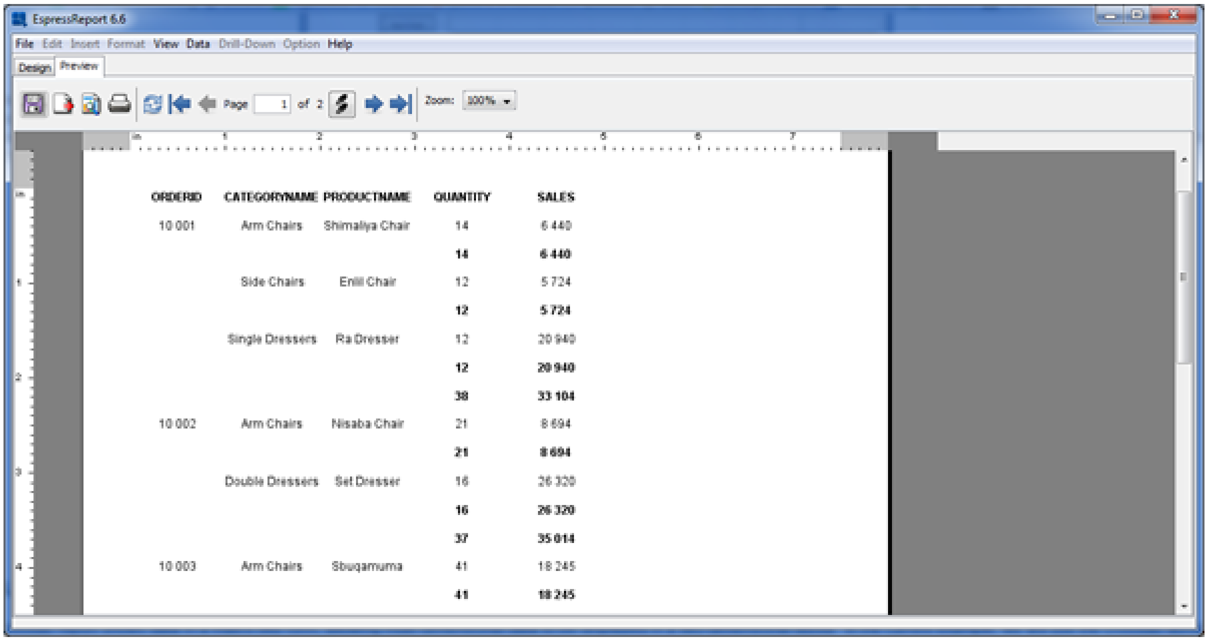
Data Mapping Screen for Summary Break Layout

In the data mapping dialog, check the option called *Row Break* for the first two columns. This will group the report by those two columns. From the drop down menus under *Aggregation*, select *SUM* for QUANTITY and SALES columns. Also, uncheck the *Apply Template* option, as you do not need to carry over the formatting from the simple columnar layout. Once you finish specifying the options, click the *Done* button and then *Yes* on the following warning message. You will be taken back to the Report Designer where the new mapping has taken effect.



Summary Break Report in Design Window

In the Designer, you will notice that there are two levels of nested grouping in the report; therefore, there are now corresponding Group Header and Footer sections for each. For more about report sections and their behavior, please see Section 1.5.1 - Report Sections of the Designer Guide. Now click on the *Preview* tab to preview the report. Again, you will be prompted to specify parameter values. Once you see the report in the Preview window, notice how the data is grouped by Category Name and Order ID, and that intermediate summaries are calculated for each group.



ORDERID	CATEGORYNAME	PRODUCTNAME	QUANTITY	SALES
10 001	Arm Chairs	Shimaliya Chair	14	6 440
			14	**6 440**
	Side Chairs	Enlil Chair	12	5 724
			12	**5 724**
	Single Dressers	Ra Dresser	12	20 940
			12	**20 940**
			38	**33 104**
10 002	Arm Chairs	Nisaba Chair	21	8 694
			21	**8 694**
	Double Dressers	Set Dresser	16	26 320
			16	**26 320**
			37	**35 014**
10 003	Arm Chairs	Sbuqamuma	41	18 245
			41	**18 245**

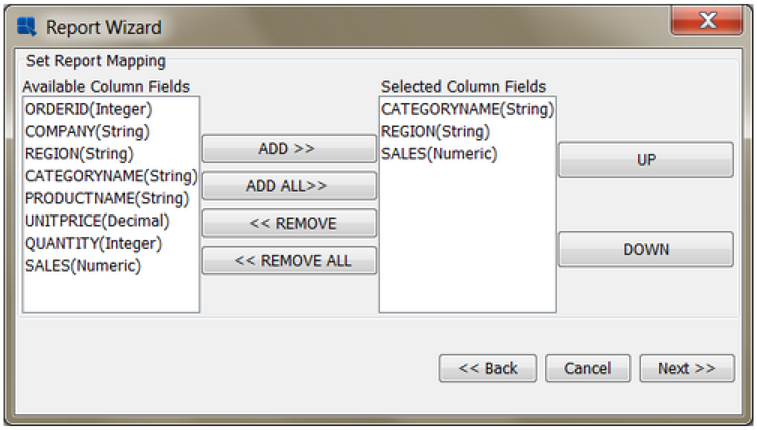
Summary Break Report Preview

Q.3.3.3. Crosstab Layout

A crosstab report shows data in a matrix-like form, allowing multi-dimensional data to be displayed in a two-dimensional layout. In the current example, we will use the crosstab layout to break-down the sales column by product category and by region.

Go to the Design tab and click on the *Change Data Mapping* icon. When the Report Wizard re-opens, click the *Back* button until you get back to the layout selection screen. From this screen, select *CrossTab* layout and click the *Next* button. At the column selection/ordering screen (next in the Wizard), change the selection to the following:

```
CATEGORYNAME
REGION
SALES
```


Report Wizard

Set Report Mapping

Available Column Fields

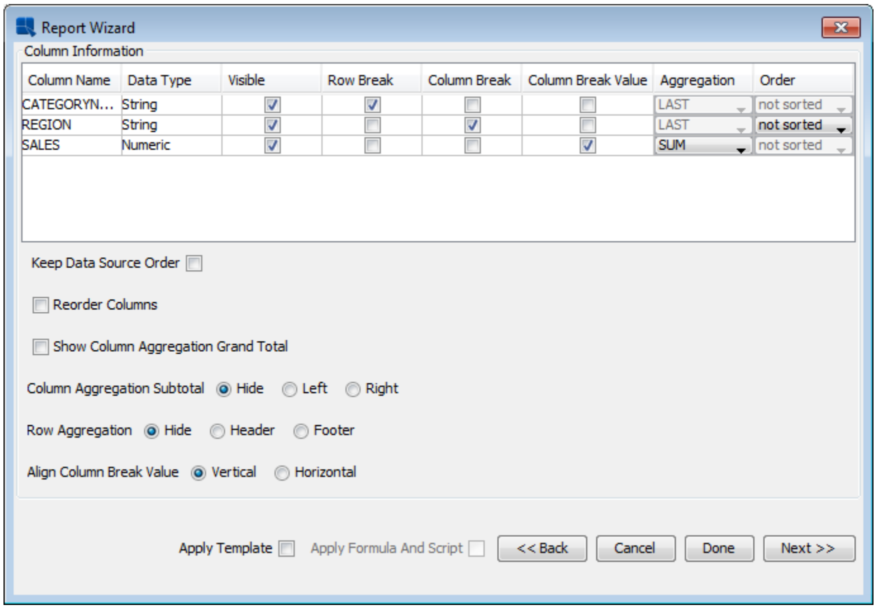
- ORDERID(Integer)
- COMPANY(String)
- REGION(String)
- CATEGORYNAME(String)
- PRODUCTNAME(String)
- UNITPRICE(Decimal)
- QUANTITY(Integer)
- SALES(Numeric)

Selected Column Fields

- CATEGORYNAME(String)
- REGION(String)
- SALES(Numeric)

Column Selection for Crosstab Report

Once you specify the columns in correct order, click the *Next* button to bring up the data mapping option for the crosstab layout.


Report Wizard

Column Information

Column Name	Data Type	Visible	Row Break	Column Break	Column Break Value	Aggregation	Order
CATEGORYN...	String	☑	☑	□	□	LAST	not sorted
REGION	String	☑	□	☑	□	LAST	not sorted
SALES	Numeric	☑	□	□	☑	SUM	not sorted

Keep Data Source Order □

□ Reorder Columns

□ Show Column Aggregation Grand Total

Column Aggregation Subtotal ● Hide ○ Left ○ Right

Row Aggregation ● Hide ○ Header ○ Footer

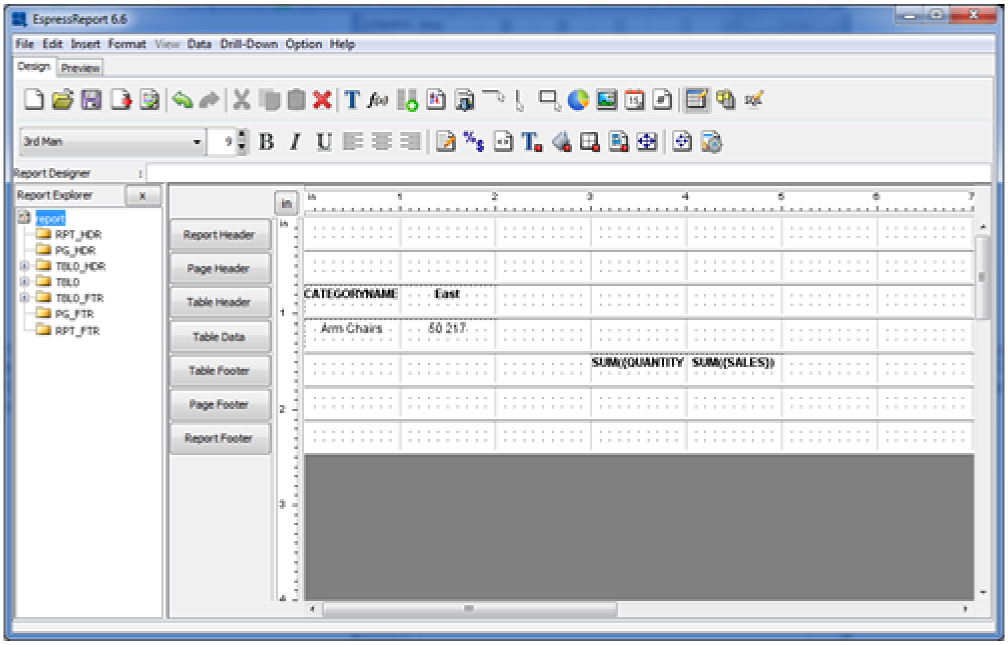
Align Column Break Value ● Vertical ○ Horizontal

Apply Template □ Apply Formula And Script □

Data Mapping Screen for Crosstab Layout

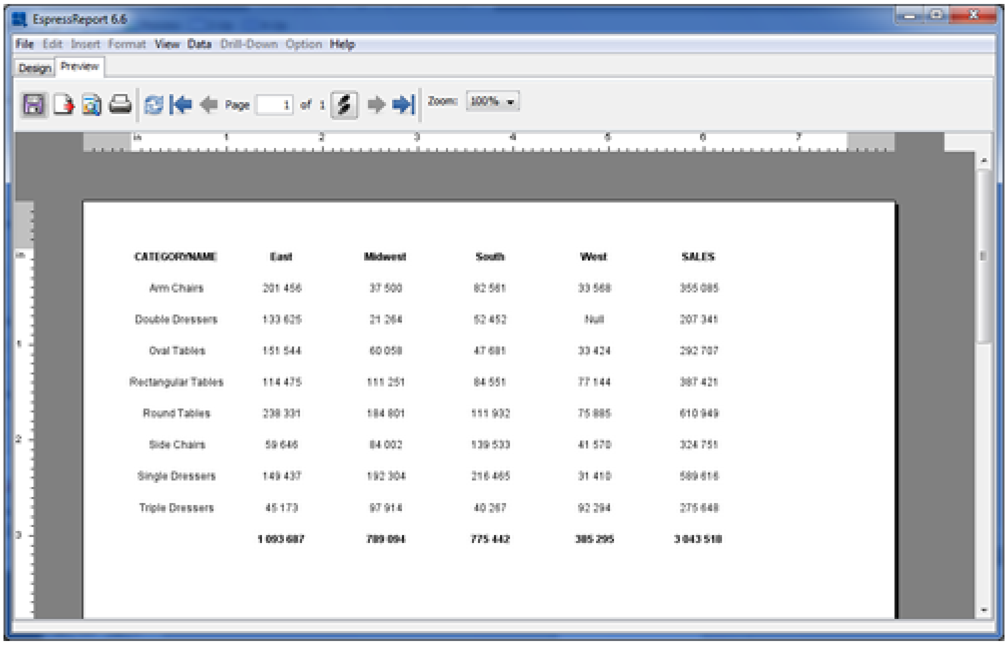
For the CATEGORYNAME column, check the option marked *Row Break*. This will create a row in the report data for each distinct category name. Next, select the *Column Break* option for the Region column. This will create a column in the report data for each distinct region. Leave the *Order* option as **`not sorted`**. Finally, select the

Column Break Value option for the Sales column, then click on the *Aggregation* menu and select **`SUM`**. This will give you total sales for each category and region in the report. Once you finish specifying the options, click the *Done* button, then *Yes* on the warning message to go back to the Report Designer.



Crosstab Report in Design Window

As you can see, a report column has been generated for each region and it has been automatically totaled both vertically (columns) and horizontally (rows). Now, click on the *Preview* tab to preview the crosstab layout.



CATEGORYNAME	East	Midwest	South	West	SALES
Arm Chairs	201 456	37 500	82 581	33 568	355 085
Double Dressers	133 625	21 264	52 452	Null	207 341
Oval Tables	151 544	60 058	47 681	33 424	292 707
Rectangular Tables	114 475	111 251	84 551	77 144	387 421
Round Tables	238 331	184 801	111 932	75 885	610 949
Side Chairs	59 646	84 002	139 533	41 570	324 751
Single Dressers	149 437	192 304	216 465	31 410	589 616
Triple Dressers	45 173	97 914	40 267	92 294	275 648
	1 093 687	**789 094**	**775 442**	**385 295**	**3 043 518**

Crosstab Report Preview

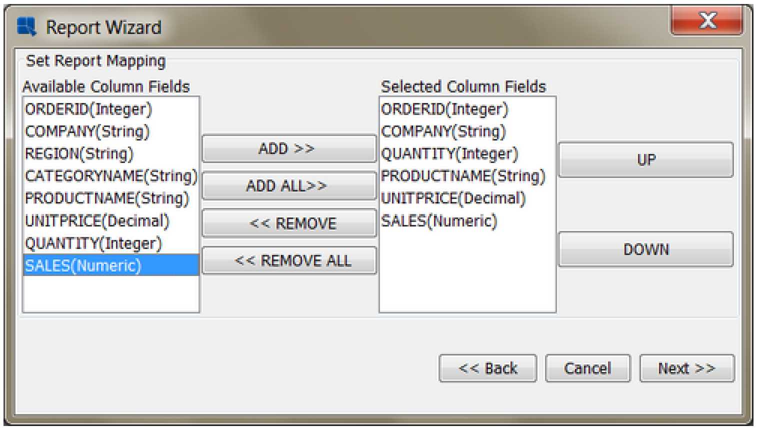
Try previewing again (by going to the design window and then clicking *Preview* again), this time specifying a much smaller time range. Notice how the number of columns decreases because the values no longer exist in the data. When you expand the date range again the columns will re-appear.

Q.3.3.4. Master & Details Layout

Like the summary break layout, the Master & Details layout also allows you to group the data. It also allows you to automatically add column fields to the group header section, creating many unique layouts that can be configured in a side-by-side layout.

Go back to the Design tab and select *Change Data Mapping* from the toolbar. Again, navigate back to the layout selection screen. This time, select the Master & Details layout, then click the *Next* button to get to the column selection screen. In the column selection screen, select the following columns:

```
ORDERID
COMPANY
QUANTITY
PRODUCTNAME
UNITPRICE
SALES
```



Report Wizard

Set Report Mapping

Available Column Fields

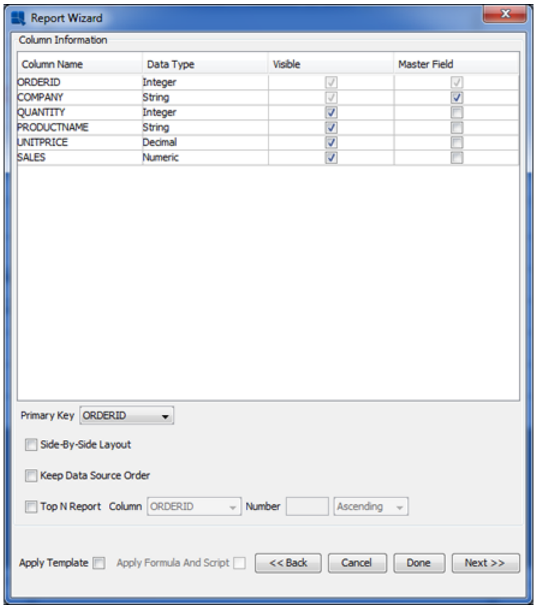
ORDERID(Integer)
COMPANY(String)
REGION(String)
CATEGORYNAME(String)
PRODUCTNAME(String)
UNITPRICE(Decimal)
QUANTITY(Integer)
SALES(Numeric)

Selected Column Fields

ORDERID(Integer)
COMPANY(String)
QUANTITY(Integer)
PRODUCTNAME(String)
UNITPRICE(Decimal)
SALES(Numeric)

Column Selection for Master & Details Report

Once you specify the columns in the correct order, click the *Next* button to bring up the data mapping dialog for the master & details layout.



Report Wizard

Column Information

Column Name	Data Type	Visible	Master Field
ORDERID	Integer	☑	☑
COMPANY	String	☑	☑
QUANTITY	Integer	☑	☐
PRODUCTNAME	String	☑	☐
UNITPRICE	Decimal	☑	☐
SALES	Numeric	☑	☐

Primary Key: ORDERID

☐ Side-By-Side Layout

☐ Keep Data Source Order

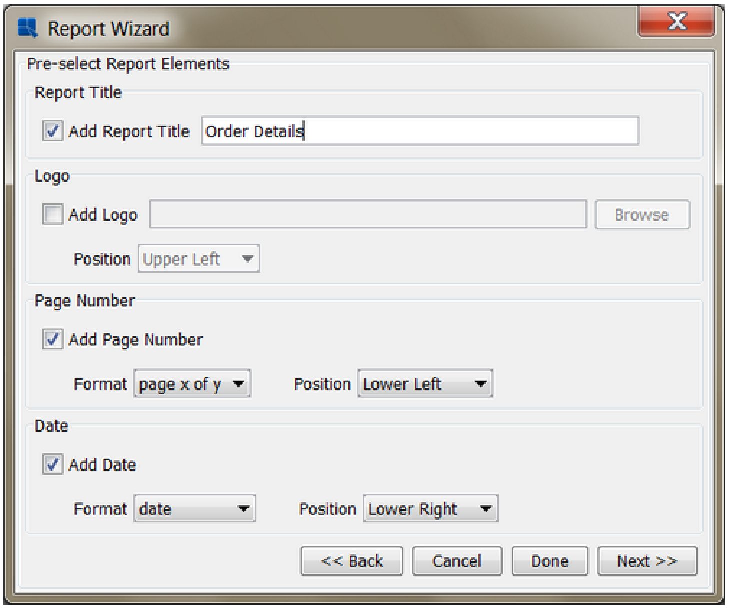
☐ Top N Report Column ORDERID Number Ascending

Apply Template ☐ Apply Formula And Script ☐

Data Mapping Screen for Master & Details Layouts

Select the *ORDERID* field as the *Primary Key* from the drop-down menu in the lower-left portion of the screen. This will group the data by the OrderID field. Next, check the *Master Field* option for the Company column. This will place the Company field in the Group Header section of the report.

Instead of clicking *Done* to get to the Report Designer from this screen, click the *Next* button to invoke additional pre-formatting options. Once again, click the *Yes* button on the warning message. The first dialog allows you to add several elements to the report.



Pre-select Report Elements

Report Title

☑ Add Report Title: Order Details

Logo

□ Add Logo

Position: Upper Left

Page Number

☑ Add Page Number

Format: page x of y Position: Lower Left

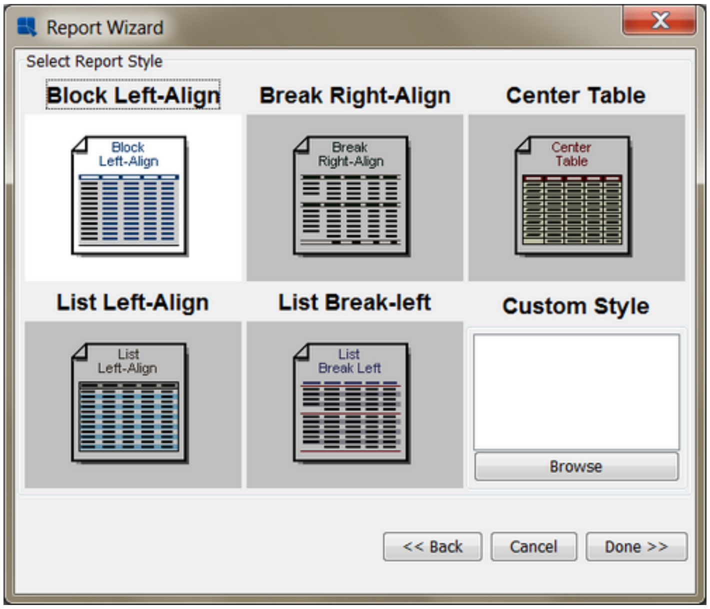
Date

☑ Add Date

Format: date Position: Lower Right

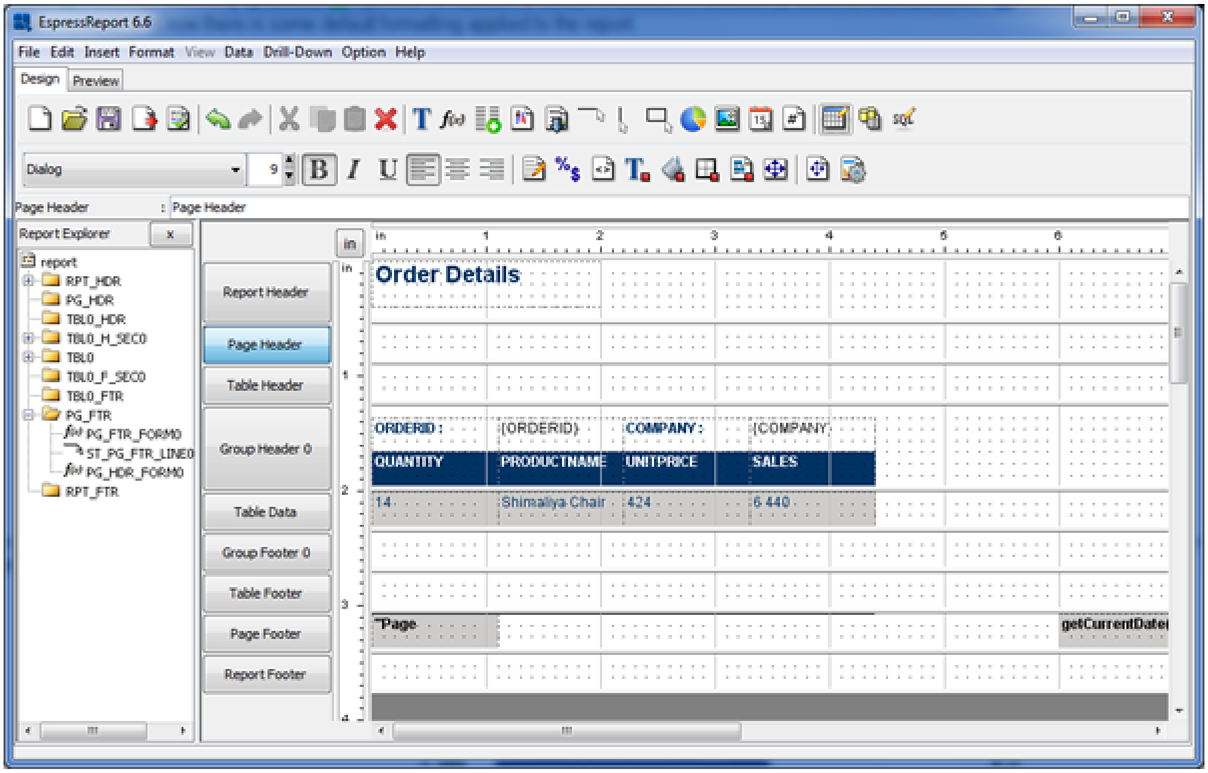
Add Report Elements Dialog

Check the boxes to add a report title, page number, and date. Enter a text you want to use as the report title and specify the format and location of the date and time using the drop-down boxes for each option. Once you finish setting the options, click the *Next* button. A new dialog will open which will enable you to specify a style for the new report.



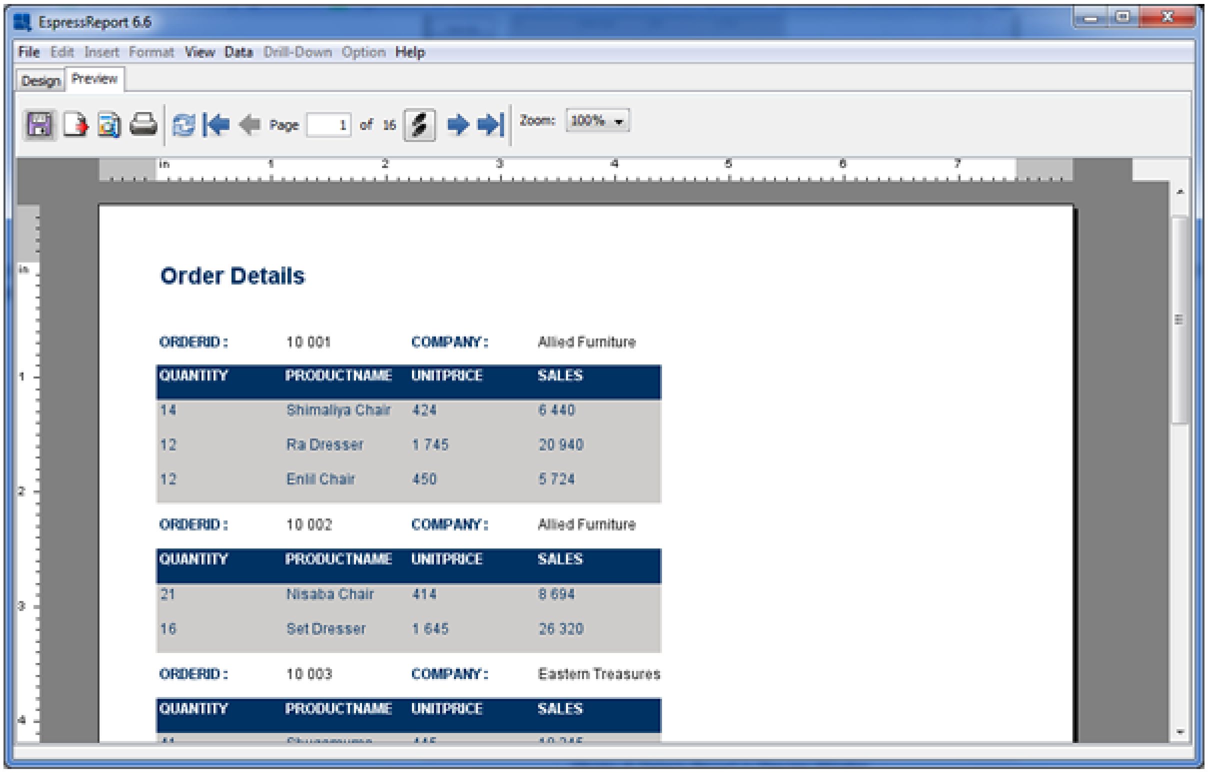
Report Style Selection

Select the *Block Left-Align* style and then click the *Done* button. You will be taken back to the Report Designer window, where new elements have been added and the report will also have default formatting applied.



Master & Details Report in Design Window (with pre-formatting)

As you can see the OrderID and Company fields have been generated in the Group Header section. Now preview the report and you will see a group with each order, as well as the page number and date in the section (header or footer) in which they were placed.



Order Details

ORDERID : 10 001 COMPANY : Allied Furniture

QUANTITY	PRODUCTNAME	UNITPRICE	SALES
14	Shimaliya Chair	424	6 440
12	Ra Dresser	1 745	20 940
12	Enlil Chair	450	5 724

ORDERID : 10 002 COMPANY : Allied Furniture

QUANTITY	PRODUCTNAME	UNITPRICE	SALES
21	Nisaba Chair	414	8 694
16	Set Dresser	1 645	26 320

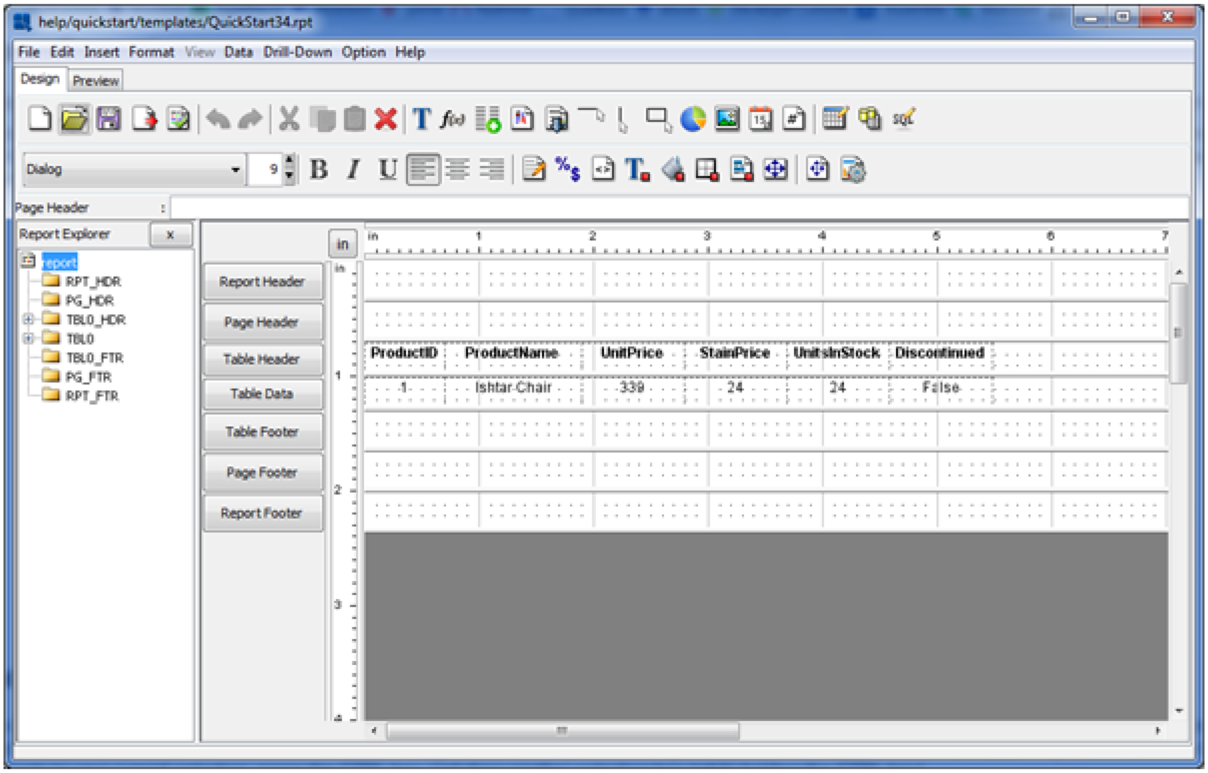
ORDERID : 10 003 COMPANY : Eastern Treasures

QUANTITY	PRODUCTNAME	UNITPRICE	SALES

Master & Details Report in Preview Window

Q.3.4. Basic Report Formatting

In this section, we will open an unformatted template and use some of the basic formatting features in EspressReport to create a polished presentation. From the Design window in Report Designer, click the *Open* button: . This will bring up a dialog prompting you to specify a filename for the template you want to open. Click the *Browse* button and navigate to `<InstallDir>/help/quickstart/templates` directory. Once you're there, select `QuickStart34.rpt` file and open it. The report will open in the Design window. As you can see, the report elements have almost no formatting.

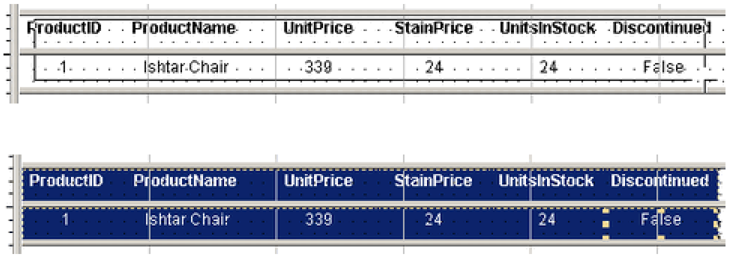


ProductID	ProductName	UnitPrice	StainPrice	UnitsInStock	Discontinued
1	Ishtar Chair	339	24	24	False

`QuickStart34.rpt` *in Design Window*

Q.3.4.1. Move and Align Report Elements

Report elements can be moved one by one by clicking and dragging the cell. Report elements can also be moved as groups. To move a group of elements, you must first select the group using the selection box. To activate the selection box, click and drag to draw a box around the report columns. When you release your mouse, they will become highlighted. To add more elements to your current selection, press **CTRL** key and draw another selection box (while still holding the **CTRL** key).



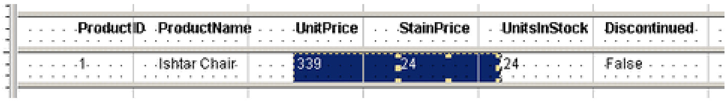
ProductID	ProductName	UnitPrice	StainPrice	UnitsInStock	Discontinued
1	Ishtar Chair	339	24	24	False

Drawing a Selection Box in Report Designer

Once the fields in the report are selected, click and drag to indent them about half an inch. Next, click the left *alignment* button on the toolbar: *selection box* button on the toolbar: . This will align all of the cell text to the left edge of a cell.

Q.3.4.2. Data Formatting

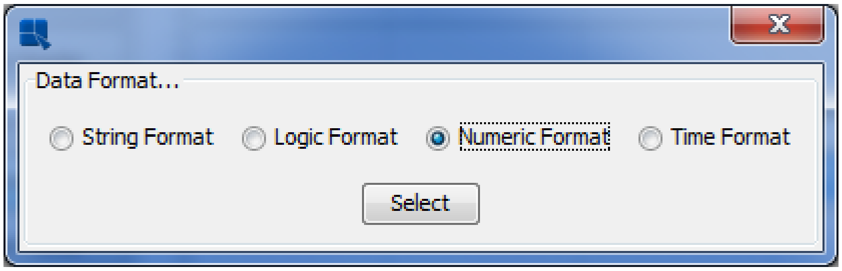
There are number of options available that allows you to control how the data are displayed (date format, decimal places, rounding, etc). Using the selection box again, select the UnitPrice and StainPrice columns (just the column fields not the headers).



ProductID	ProductName	UnitPrice	StainPrice	UnitsInStock	Discontinued
1	Ishtar Chair	339	24	24	False

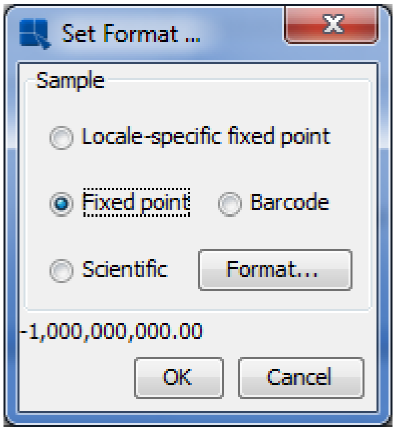
Two Column Selection

Once the columns are selected, select the *Data Format* option from the *Format* menu. This will open a dialog prompting you to select which data type you would like to set the format for. Select *Numeric Format* and click *Select*.

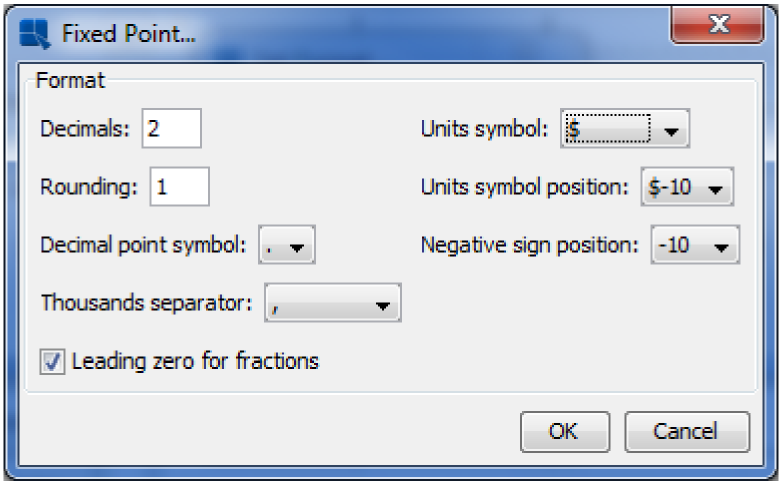


Data Type for Formatting Dialog

This will open a new dialog. From this dialog, select **`Fixed Point`** and click the *Format* button. At the next dialog, specify 2 decimal points and select the dollar sign as the Units symbol.



Set Format Dialog



Format

Decimals: 2

Rounding: 1

Decimal point symbol: .

Thousands separator: ,

☑ Leading zero for fractions

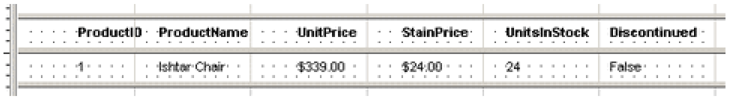
Units symbol: $

Units symbol position: $-10

Negative sign position: -10

Numeric Format Dialogs

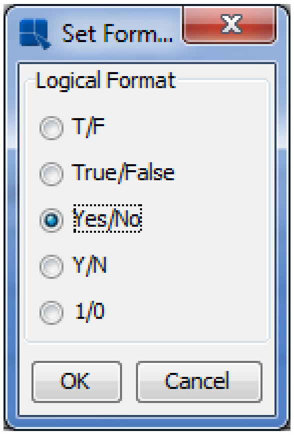
Click *Ok* and *Ok* again at the previous dialog and the selected fields will be converted to currency format.



ProductID	ProductName	UnitPrice	StainPrice	UnitsInStock	Discontinued
1	Ishtar Chair	$339.00	$24.00	24	False

Numeric Data in Currency Format

Next, select the Discontinued column by clicking on it (the border outline will appear). Again, click the *Data Format* button. This time a dialog will appear allowing you to select a format for the Boolean column. Select *Yes/No* and click *Ok*. The data in the column will now change to *No*.



Logical Format

○ T/F

○ True/False

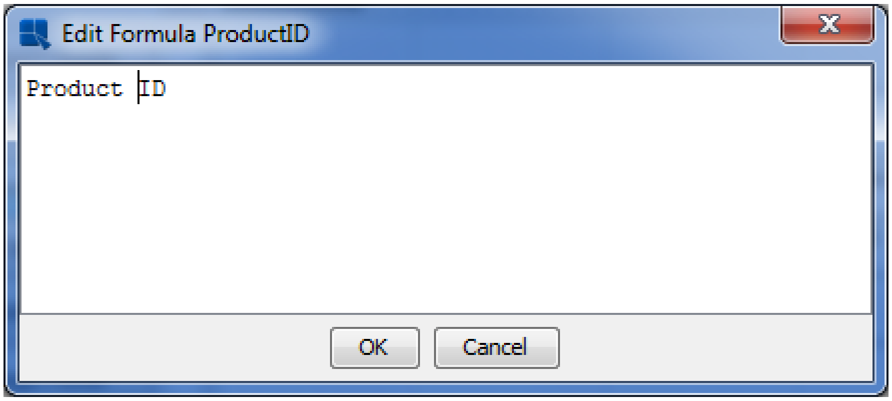
● Yes/No

○ Y/N

○ 1/0

Data Format Dialog for Boolean Data

Now we will edit the label text. By default, column headers will display column names from the database. However, you can override these headers with a custom one. To do so, double click on the *ProductID* cell and a dialog will appear allowing you to modify the column header. Insert a space between "Product" and "ID" and click *OK*. The change will appear in the Design window.



Edit Formula ProductID

```
Product ID
```

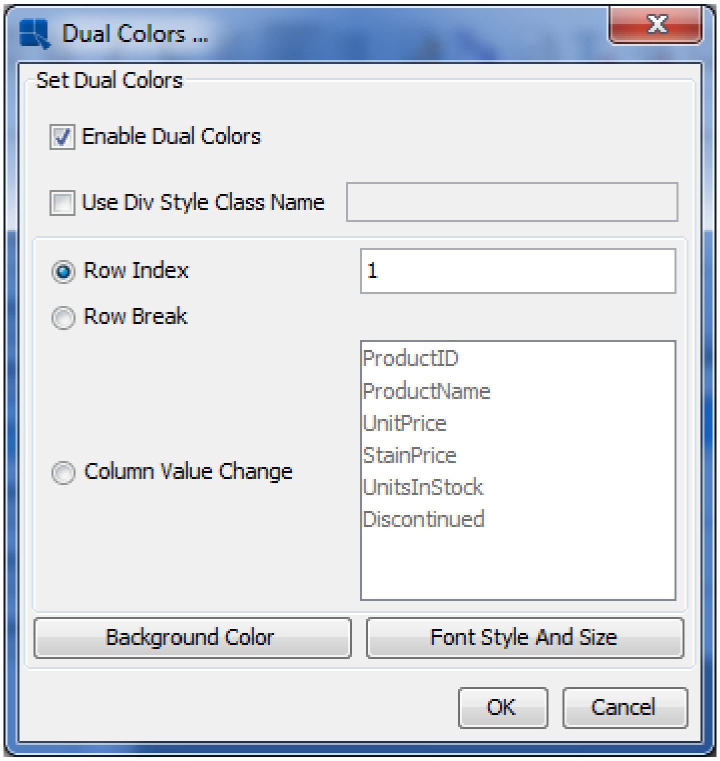
Edit Column Header Dialog

Repeat this for each column label (except "Discontinued") in order to have proper names for all of them.

Q.3.4.3. Set Dual Colors

The dual colors feature in EspressReport allows users to differentiate different rows or groups of data by changing the background color and/or font. To turn on dual colors, use the group selection tool to select all columns in the report (not headers).

Next, select the *Dual Colors* button on the toolbar: . A dialog will appear prompting you to set dual colors for the columns. Click the check box labeled *Enable Dual Colors*. Then select the *Row Index* radio button to indicate that you want to change color based on a row value. Enter **1** for the row index value.



Dual Colors ...

Set Dual Colors

☑ Enable Dual Colors

□ Use Div Style Class Name

● Row Index: 1

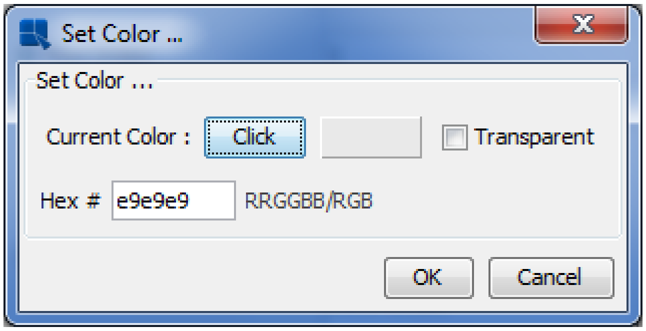
○ Row Break

○ Column Value Change

ProductID
ProductName
UnitPrice
StainPrice
UnitsInStock
Discontinued

Dual Colors Dialog

Next, click the *Background Color* button. A dialog will appear, giving you the option to set the background transparent and showing the current background color.

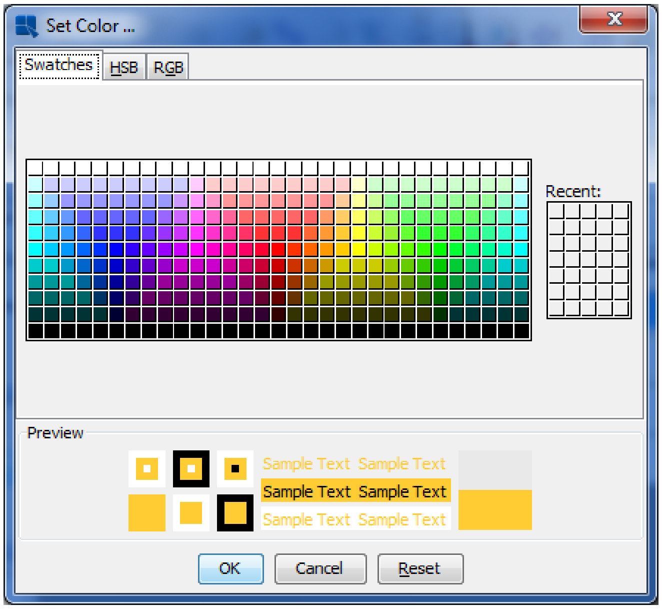


□ Transparent

Hex # e9e9e9

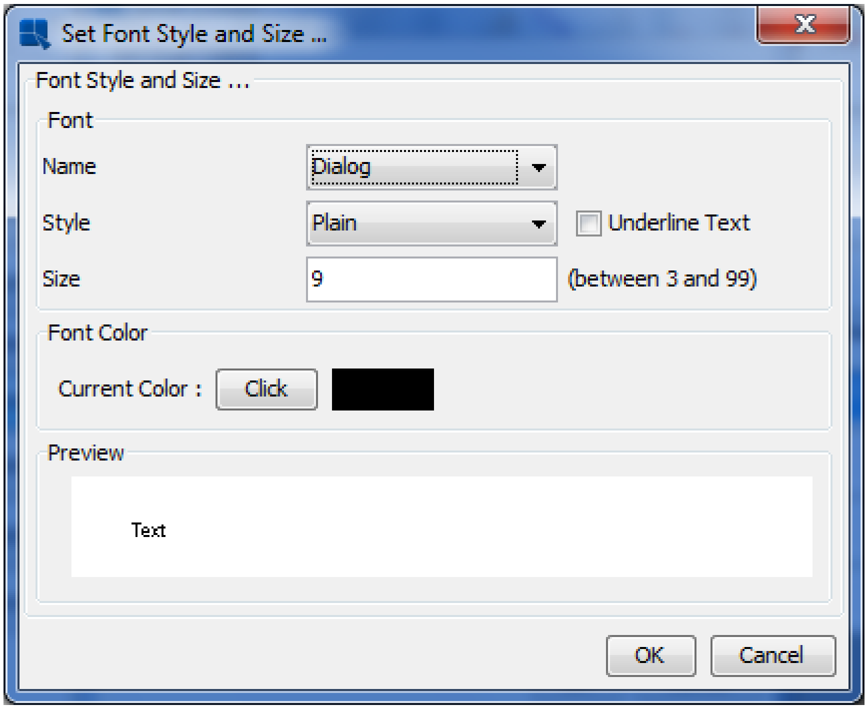
Background Transparency Dialog

Click the button labeled *Click* and a new dialog with color swatches will appear, allowing you to change background color.



Choose Color Dialog

Select a new background color you want to use and click the *OK* button. You will be returned to the first dialog, where the color selection will be reflected. Click the *OK* button again and you will be returned to the Dual Colors dialog. Once you are back at the Dual colors dialog, click the *Font Style and Size* button. This will bring up a dialog allowing you to specify the font, font size, and font style for the alternating rows.



Name: Dialog

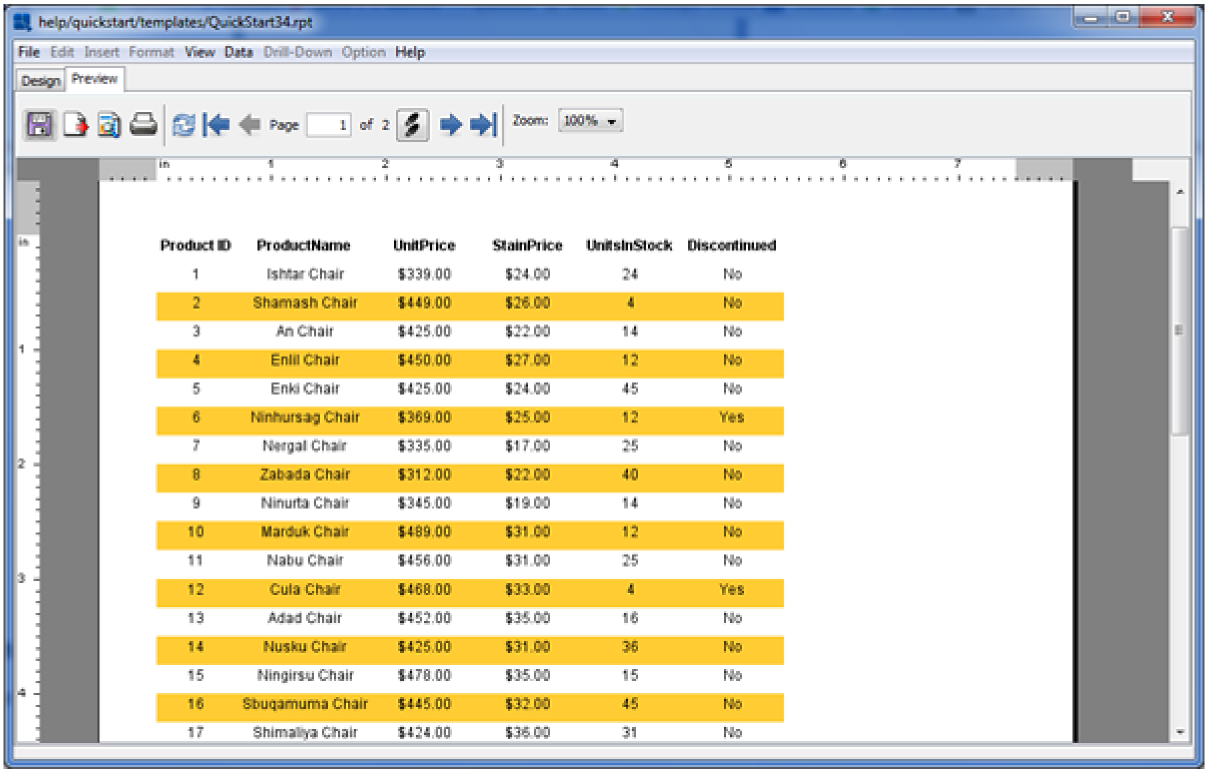
Style: Plain

☐ Underline Text

Size: 9

Font Style and Size Dialog

From this dialog, set *Name* to **`Dialog`**, *Style* to **`Plain`** and *Size* to **`9`**. This will match the fonts for the alternating rows. Click the *Ok* button to go back to the dual colors dialog and once more to return to the Report Designer. Now when you preview the report, you will see alternating bands of color for each row.


help/quickstart/templates/QuickStart34.rpt
File Edit Insert Format View Data Drill-Down Option Help
Design Preview
Page 1 of 2
Zoom: 100%

Product ID	ProductName	UnitPrice	StainPrice	UnitsInStock	Discontinued
1	Ishtar Chair	$339.00	$24.00	24	No
2	Shamash Chair	$449.00	$26.00	4	No
3	An Chair	$425.00	$22.00	14	No
4	Enlil Chair	$450.00	$27.00	12	No
5	Enki Chair	$425.00	$24.00	45	No
6	Ninhursag Chair	$369.00	$25.00	12	Yes
7	Nergal Chair	$335.00	$17.00	25	No
8	Zabada Chair	$312.00	$22.00	40	No
9	Ninurta Chair	$345.00	$19.00	14	No
10	Marduk Chair	$489.00	$31.00	12	No
11	Nabu Chair	$456.00	$31.00	25	No
12	Cula Chair	$468.00	$33.00	4	Yes
13	Adad Chair	$452.00	$35.00	16	No
14	Nusku Chair	$425.00	$31.00	36	No
15	Ningirsu Chair	$478.00	$35.00	15	No
16	Sbuqamuma Chair	$445.00	$32.00	45	No
17	Shimaliya Chair	$424.00	$36.00	31	No

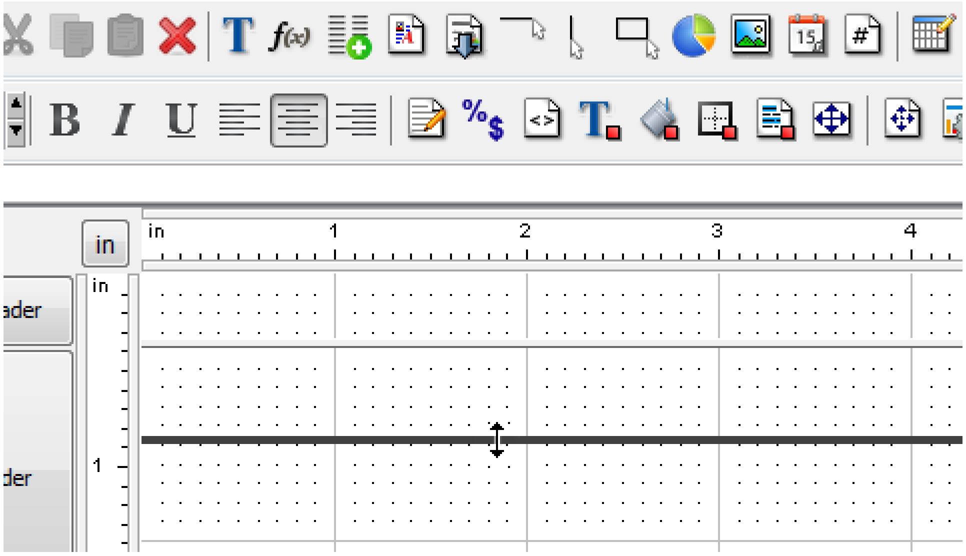
Dual Colors in Preview

Q.3.4.4. Inserting Elements

To further customize reports, many different types of elements can be added to a report template.

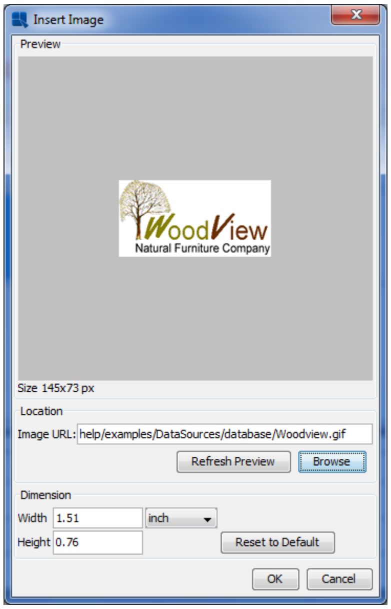
Q.3.4.4.1. Insert an Image

To insert an image into the report header, you first need to resize the report section to fit the image. In the Design window, place the mouse over the lower section divider of the Report Header section, then click and drag down making the section about an inch taller.



Resizing a Report Section

Once the section is resized, click the *Image* button on the toolbar: . A small rectangle will follow your mouse pointer around the Design window. Position the rectangle in the upper left corner of the Report Header section and click. A dialog will appear prompting you to select an image to insert. Click on *Browse* and navigate to `help\examples\DataSources\database\Woodview.gif`. You should now see an image in the preview panel.



Size 145x73 px

Image URL: help/examples/DataSources/database/Woodview.gif

Width 1.51 inch

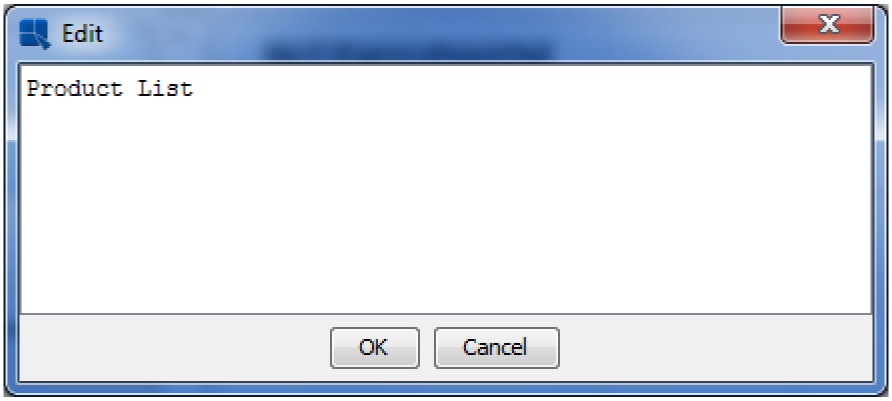
Height 0.76

Insert Image Dialog

Click *OK* and the image will be inserted into the report. The image will be represented by a gray rectangle in the Design view. You can see the image when you preview the report.

Q.3.4.4.2. Insert a Title

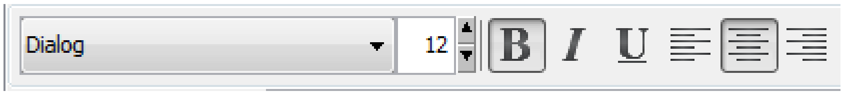
To insert a title, click the *insert label* button on the toolbar: . A small rectangle will now follow your mouse pointer around the Design window. Position the rectangle next to the image you inserted and click. A dialog will appear prompting you to enter the label text. Type in the text of your desired title.


Edit

Product List

Insert Label Dialog

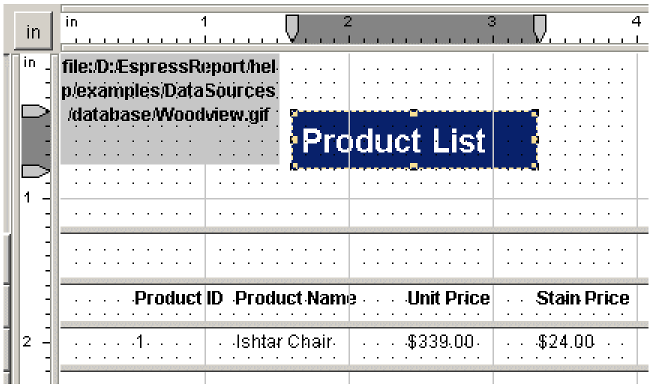
Click *OK* and the title will be added to the report. As you can see, by default the text is fairly small. To change this, change the font size dialog on the toolbar to be 18pt font.


Dialog 12

Toolbar Font Options

When you do this, you will notice that some of the text in the title cell has now disappeared. This is because the text is now larger than the defined space. To resize the cell, click and drag on the resizing handles until you can see the report title again.

Then click the *Left Alignment* button on the toolbar to set the text alignment to the left as you did for the other report elements. Move and position the title cell so it is next to the inserted image.


file:D:/EspressReport/help/examples/DataSources/database/Woodview.gif

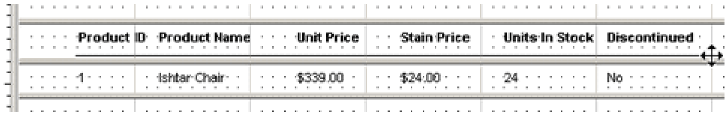
Product List

Product ID	Product Name	Unit Price	Stain Price
1	Ishtar Chair	$339.00	$24.00

Report Header with Image and Title

Q.3.4.4.3. Insert a Line

Next, we will add a horizontal line below the column headers. To do this, first resize the Table Header section slightly to provide some more space below the column headers. Next, select the *Insert Horizontal Line* button on the toolbar: . Your cursor will then change into a cross. Click below the ProductID header and drag across to the last column to draw a line.

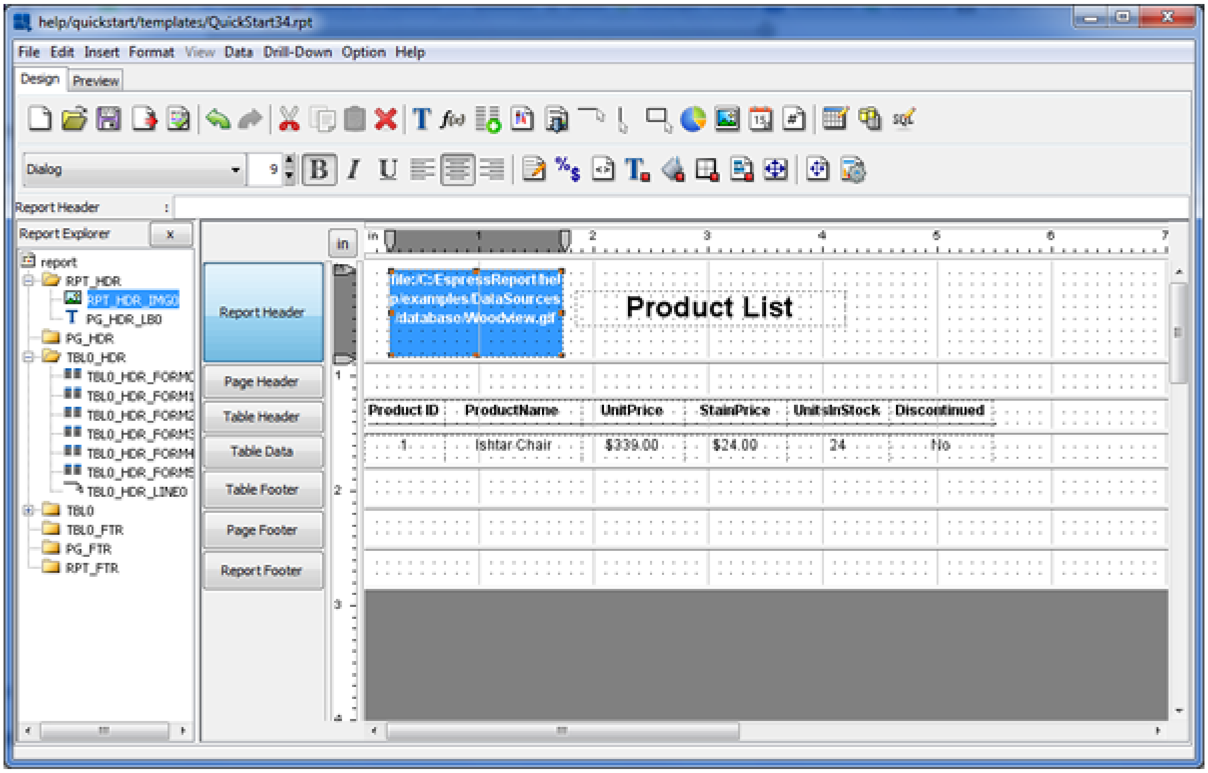


Product ID	Product Name	Unit Price	Stain Price	Units In Stock	Discontinued
1	Ishtar Chair	$339.00	$24.00	24	No

Drawing a Line in Report Designer

Q.3.4.5. View Report Elements in Report Explorer

While selecting report elements, you may have noticed the *Report Explorer* panel on the left side of the Report Designer, showing the report elements in a tree format. Click to expand some of the sections and you will see all the elements in the report represented in the tree. If you select one of the elements in the tree, the corresponding element in the report will be selected (and vice-versa).



Product ID	ProductName	UnitPrice	StainPrice	UnitsInStock	Discontinued
1	Ishtar Chair	$339.00	$24.00	24	No

Designer with Explorer Open

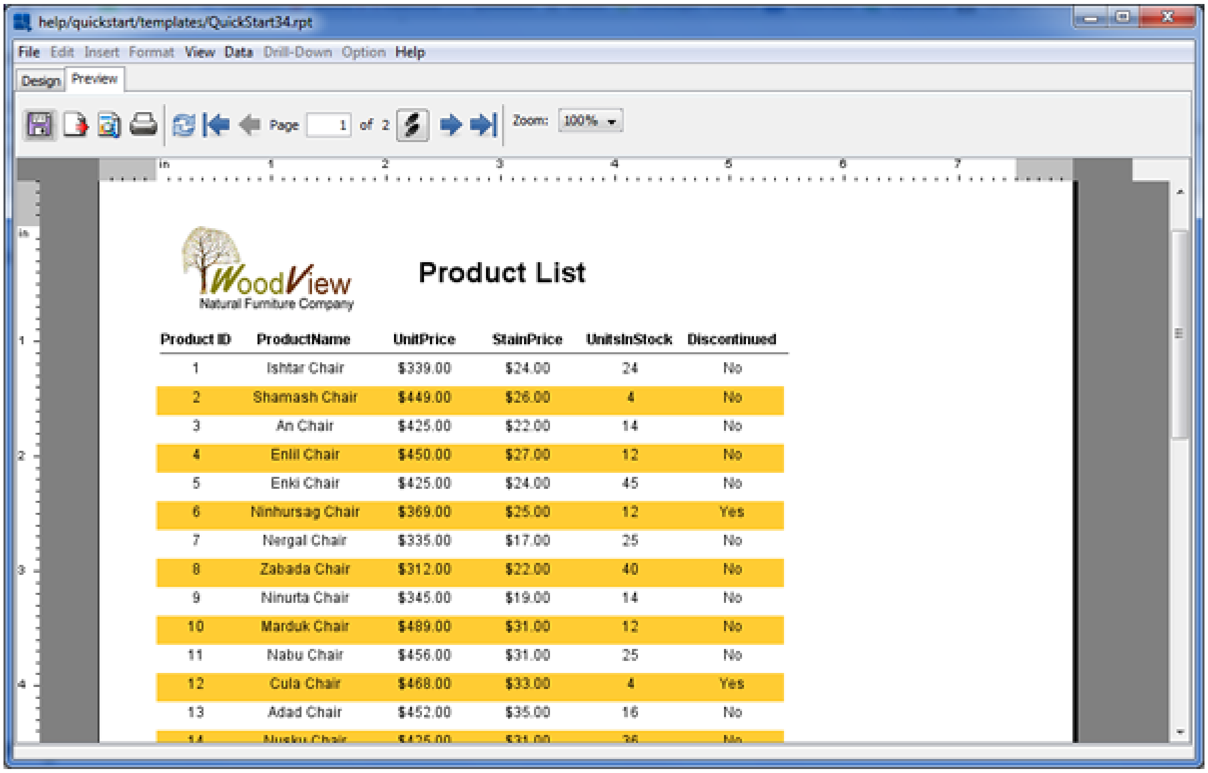
You can close the report explorer by selecting Option → Report Explorer, or by clicking on the *X* button in the top-right corner of the *Report Explorer* panel.

Q.3.4.6. Set Section Options

In EspressReport, each of the report sections has a number of configurable options allowing you to display data in sections in a number of different ways. To invoke the options menu for a section, click the button for that section on the left side of the design window. In this example, bring up the options menu for the Table Header section by clicking on the corresponding button.

From the pop-up menu, select *Repeat On Every Page*. This will cause the Table Header to be drawn on each report page instead of only once (which is the default). For more about report section options, see Section 1.5.3 - Section Options of the Designer Guide.

Now that you have finished formatting the report, preview it to see the results.


help/quickstart/templates/QuickStart34.rpt
File Edit Insert Format View Data Drill-Down Option Help
Design Preview
Page 1 of 2
Zoom: 100%
WoodView
Natural Furniture Company

Product List

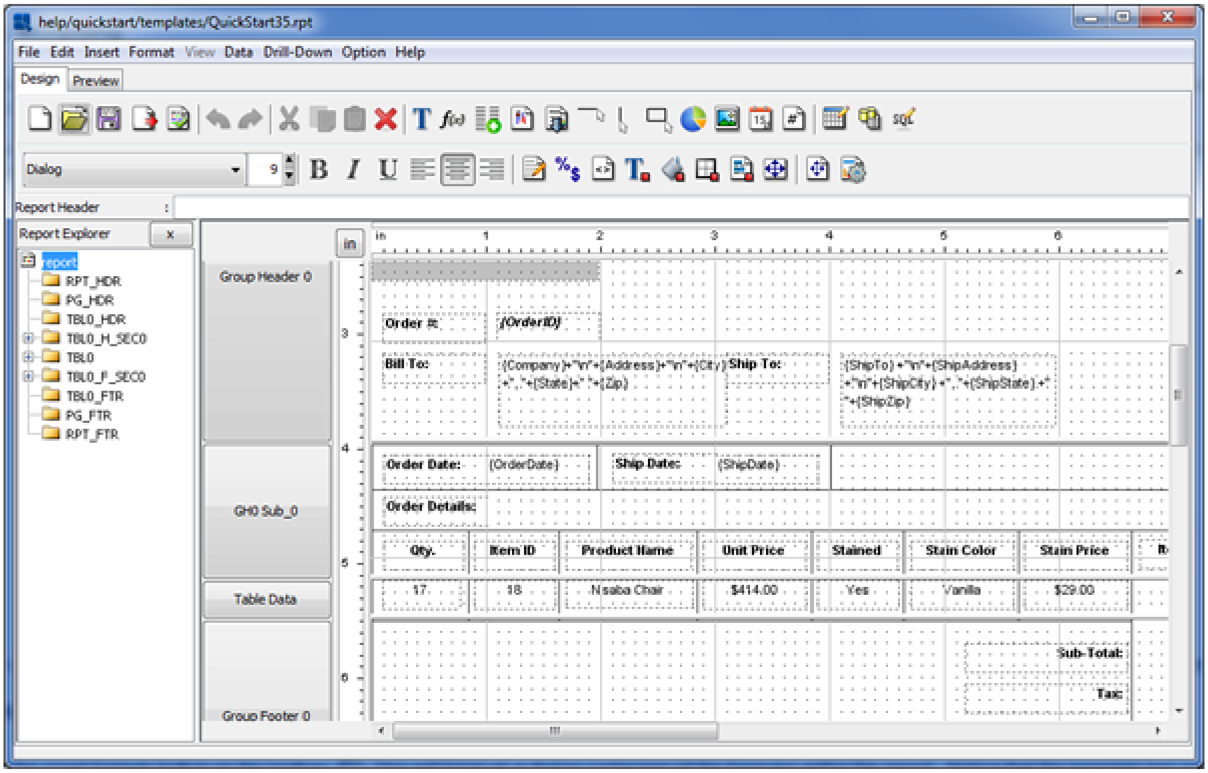
Product ID	ProductName	UnitPrice	StainPrice	UnitsInStock	Discontinued
1	Ishtar Chair	$339.00	$24.00	24	No
2	Shamash Chair	$449.00	$26.00	4	No
3	An Chair	$425.00	$22.00	14	No
4	Enlil Chair	$450.00	$27.00	12	No
5	Enki Chair	$425.00	$24.00	45	No
6	Ninhursag Chair	$369.00	$25.00	12	Yes
7	Nergal Chair	$335.00	$17.00	25	No
8	Zabada Chair	$312.00	$22.00	40	No
9	Ninurta Chair	$345.00	$19.00	14	No
10	Marduk Chair	$489.00	$31.00	12	No
11	Nabu Chair	$456.00	$31.00	25	No
12	Cula Chair	$468.00	$33.00	4	Yes
13	Adad Chair	$452.00	$35.00	16	No
14	Nusku Chair	$425.00	$31.00	36	No

Finished Report

Q.3.5. Formulas & Scripting

EspressReport provides a large built-in formula and scripting library, giving you many ways to manipulate and analyze report data. In the following section, we will take a template and use formulas and scripts to calculate/add some values to the report.

From the design window, click the *Open* button: . Then browse to the `<InstallDir>/help/quickstart/templates` directory. Once you're there, select `QuickStart35.rpt` file and open it. The report will open in the design window.

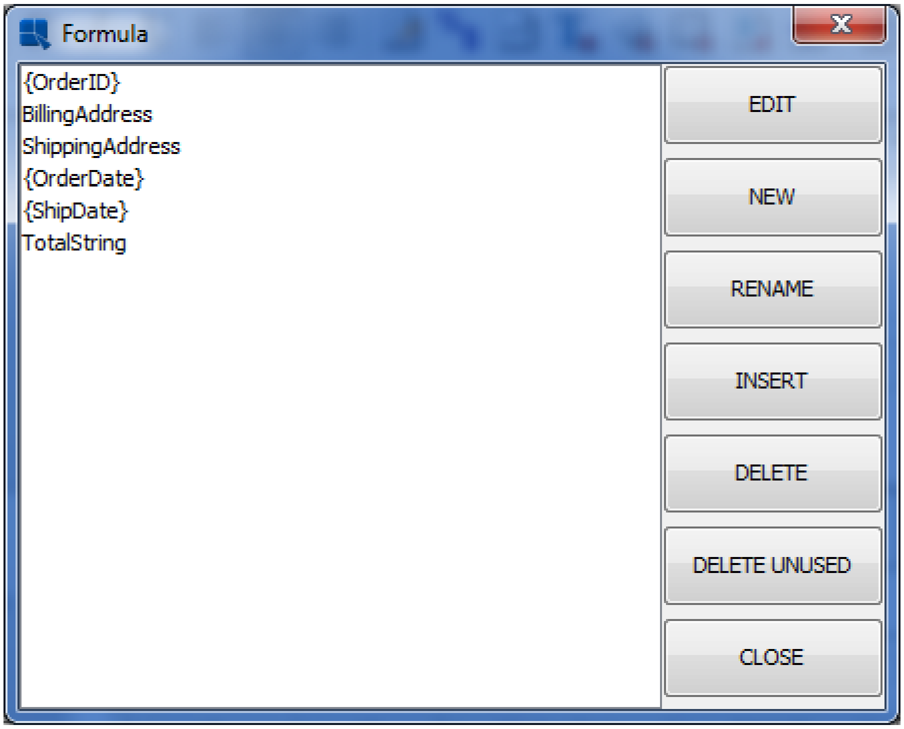


`QuickStart35.rpt` in Design Window

The report is an invoice created using the Master & Details layout. Notice that the Item Total, as well as the sub-totals are blank. We will add a few formulas to calculate these values.

Q.3.5.1. Add a Formula

To insert a formula, click the *Insert Formula* button on the toolbar: f(x). This will bring up a dialog containing all the formulas within the report. Notice that the template has several existing formulas. Click the *NEW* button to create a new formula and enter name **`ItemTotal`** at the prompt.



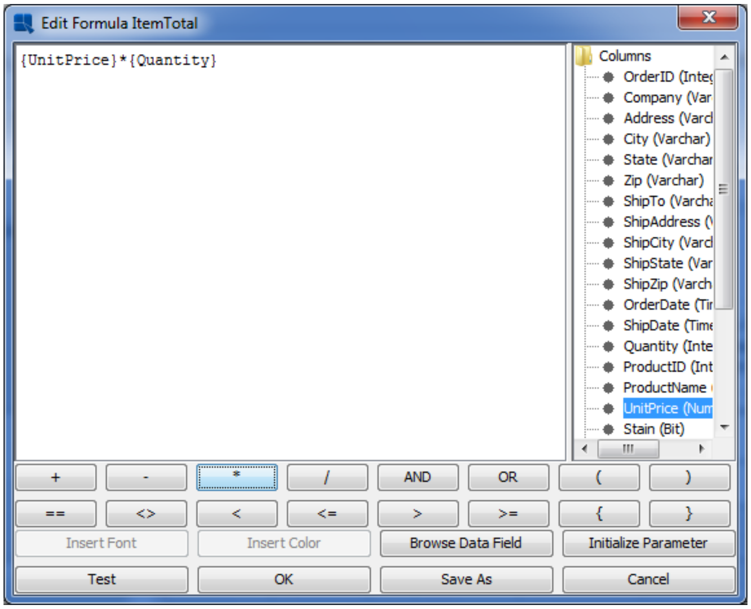
Formula

{OrderID}
BillingAddress
ShippingAddress
{OrderDate}
{ShipDate}
TotalString

Report Formula List

The Formula Builder window will then open. Double click on the *Columns* folder on the right side to expand it. Then double click on the *UnitPrice* column to add it to the formula. Next, click the multiply * button, then double click on the *Quantity* column to add it. The finished formula should look like this:

```
{UnitPrice}*{Quantity}
```



Edit Formula ItemTotal

```
{UnitPrice}*{Quantity}
```

Formula Builder Window

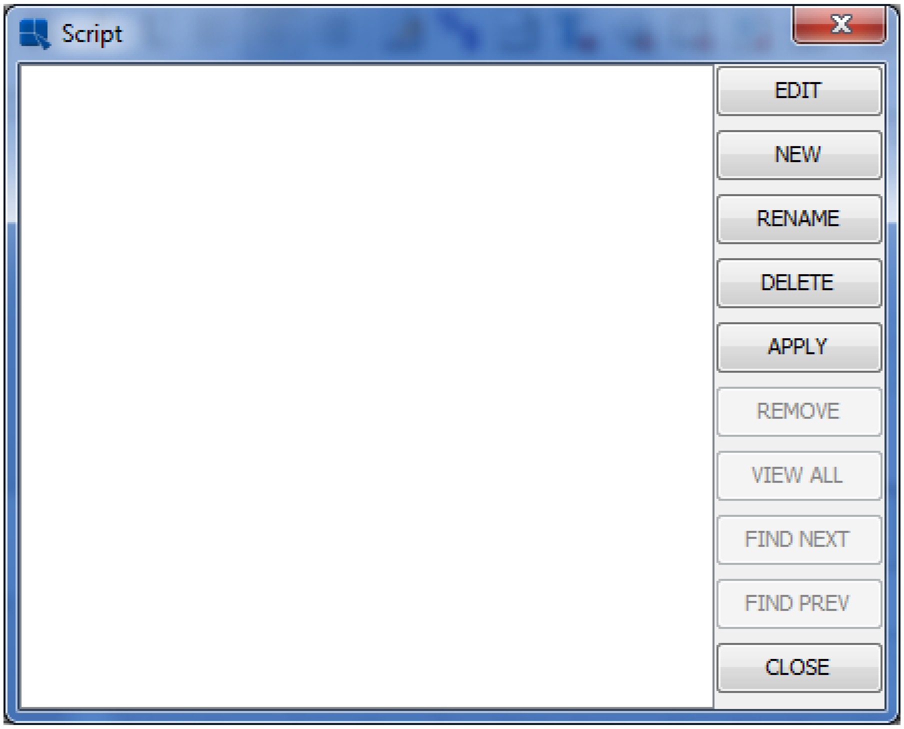
Click the *Test* button to ensure that the formula is entered correctly. Then click *OK*. You will be taken back to the formula list where your new formula has been added. From the formula list, select the *ItemTotal* formula that you just created and click the *Insert* button. The dialog will close and a small dotted rectangle will follow your pointer around the design window. Position the formula below the *Item Total* label and between the lines in the Table Data section, then click. The formula will then be added to the report.

Now preview the report and choose *Use Live Data* at the next dialog. Notice that because the formula was added to the Table data section, it now computes for each row of data. You also may have noticed that the formula data format was automatically set to currency.

Q.3.5.2. Add a Script

Notice that the formula you added in the previous section does not correctly calculate the line total for the invoice. By only multiplying the unit price and the quantity, the formula is ignoring whether an item was stained (which incurs an additional cost). To take care of this, we will use cell scripting.

To add a script, select the *ItemTotal* formula that you created in Section Q.3.5.1 - Add a Formula and click the scripting button on the toolbar: <>. This will bring up a dialog containing all the scripts in the report. Since no scripts have been previously added, the dialog will be blank.



Script List Window

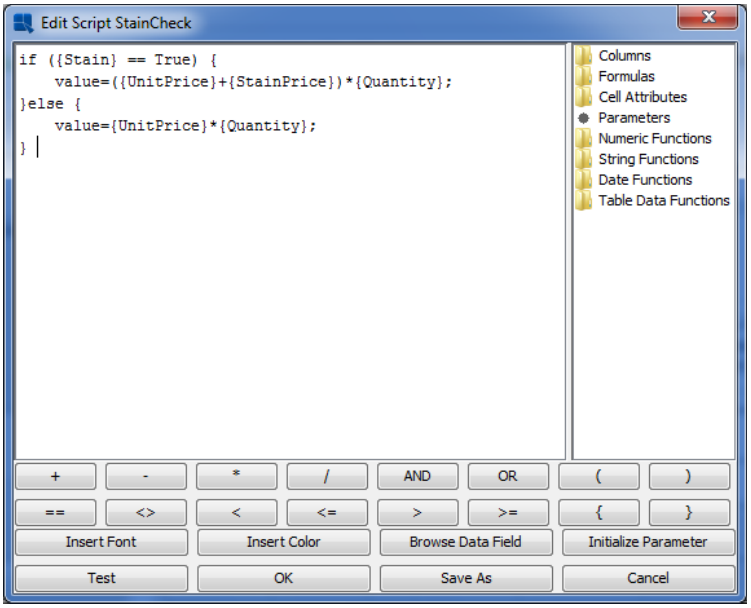
Click *NEW* to create a new script. In the prompt, type the name **`StainCheck`** for the script. Click *Ok* and the formula builder will open allowing you to add the script. Enter the following script:

```
if ({Stain} == True) {
value=({UnitPrice}+{StainPrice})*{Quantity};
 }else {
value={UnitPrice}*{Quantity};
 }
```

The script dynamically modifies the *ItemTotal* price depending on whether the item has been stained or not. If the item has been stained

```
{Stain} == True
```

, then the *ItemTotal* value includes the stain price. If the item hasn't been stained, then the price is calculated in the same manner as before.

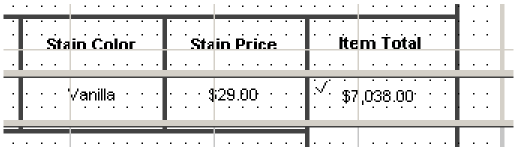


Edit Script StainCheck

```
if ({Stain} == True) {
    value=({UnitPrice}+{StainPrice})*{Quantity};
}else {
    value={UnitPrice}*{Quantity};
}
```

Formula Builder for Cell Scripts

Click *Test* to ensure that the script has been entered correctly. Then click *Ok*. You will return to the script list where the new script has been added. In the script list, select the script that you just created and click the *APPLY* button. The script will be applied to the column. Notice that a check mark now appears in the upper left corner of the *Item Total* cell.



Stain Color	Stain Price	Item Total
Vanilla	$29.00	✓ $7,038.00

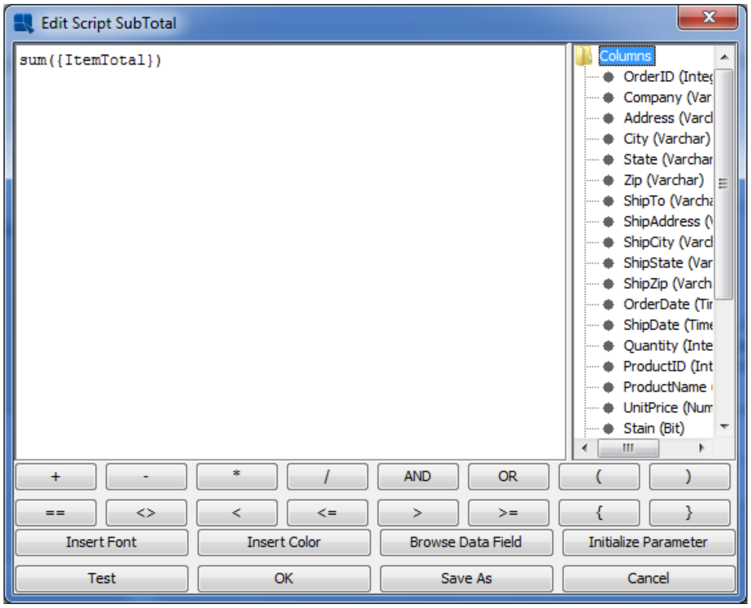
Cell with Applied Script in Design Window

Q.3.5.3. Add an Aggregation

In EspressReport, formulas can also be used to aggregate report columns. Adding an aggregation is the same as adding a formula like in Section Q.3.5.1 - Add a Formula. Click the *Insert Formula* button on the toolbar: to add a new formula. Name this formula **`SubTotal`**.

In the formula builder, double click on the *Numeric Functions* folder to expand it. Double click on the *Sum* function to insert it into the formula. Next, use the cursor to highlight the "field" portion of the sum function. Then double click on the *Columns* folder to expand it. At the end of the list there is "ItemTotal" that you created in Section Q.3.5.1 - Add a Formula. Double click on it to add it to the formula. The finished formula should look like this:

```
sum({ItemTotal})
```

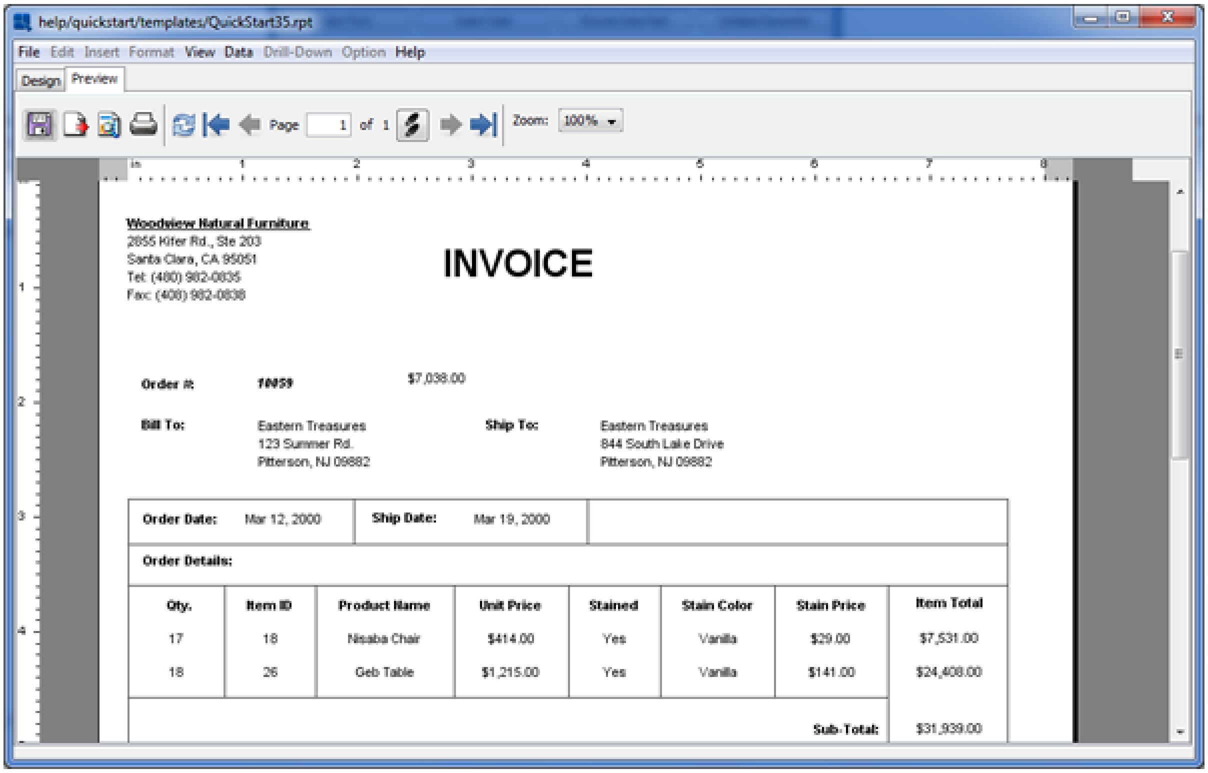


```
sum({ItemTotal})
```

Formula Builder with Aggregation Formula

Click *Test* to ensure that the formula is entered correctly. Then click *OK* to return to the formula list. In the formula list, select the formula that you have created and click the *INSERT* button. The dialog will close and a small dotted rectangle will follow your pointer around the design window. Position the formula in the Group Footer section of the report, below the Item Total column and next to Sub-Total, then click to add it.

Because the formula is in the Group Footer section, it will not calculate until the report is run and only displays the text of the formula. Now preview the report. Notice that the aggregation reflects the values that are modified by the cell script.


help/quickstart/templates/QuickStart35.rpt
File Edit Insert Format View Data Drill-Down Option Help
Design Preview
Page 1 of 1
Zoom: 100%

Woodview Natural Furniture
2855 Kifer Rd., Ste 203
Santa Clara, CA 95051
Tel: (480) 982-0835
Fax: (408) 982-0838

INVOICE

Order #: 10059 $7,038.00

Bill To: Eastern Treasures
123 Summer Rd.
Pitterson, NJ 09882

Ship To: Eastern Treasures
844 South Lake Drive
Pitterson, NJ 09882

Order Date: Mar 12, 2000 Ship Date: Mar 19, 2000

Order Details:

Qty.	Item ID	Product Name	Unit Price	Stained	Stain Color	Stain Price	Item Total
17	18	Nisaba Chair	$414.00	Yes	Vanilla	$29.00	$7,531.00
18	26	Geb Table	$1,215.00	Yes	Vanilla	$141.00	$24,408.00
						Sub-Total:	$31,939.00

Report With Formulas

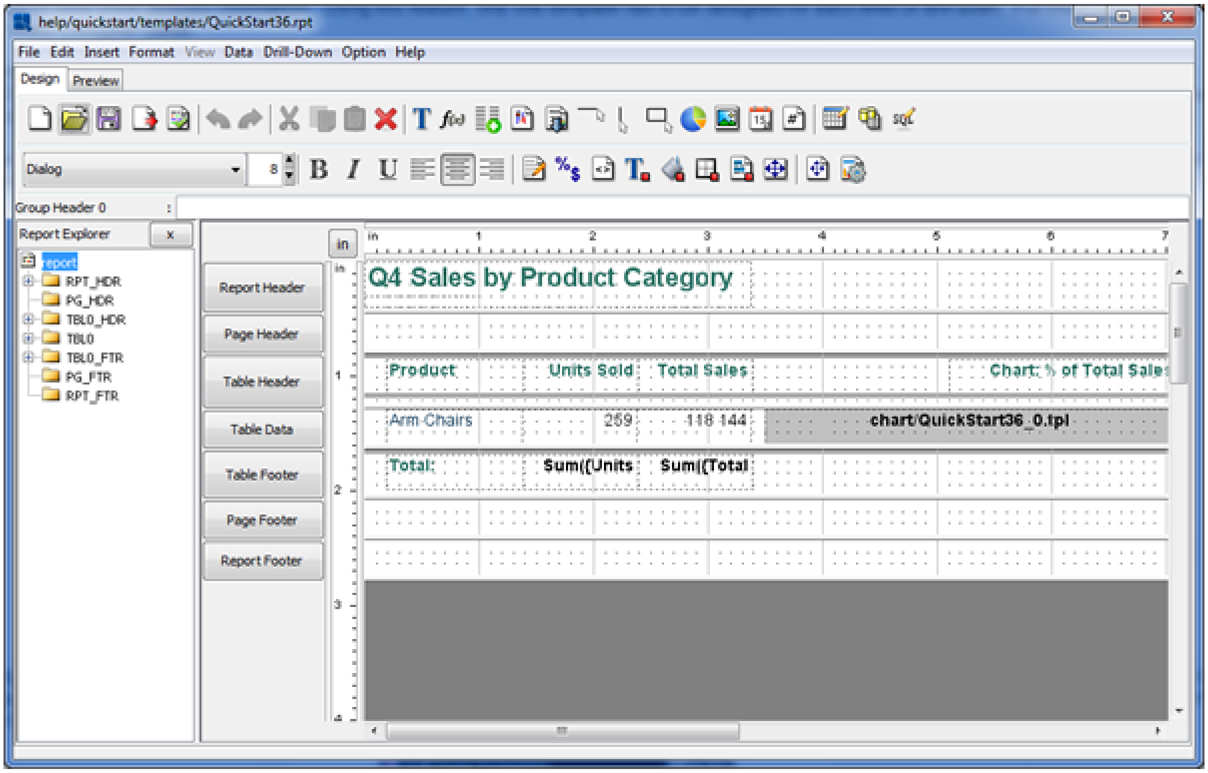
For additional tutorial, add two more formulas to the Group Footer section. One to calculate the sales tax and one to calculate a grand total for the order.

Q.3.6. Drill-Down

A unique feature of EspressReport is the ability to perform drill-down on reports automatically. Using drill-down, users can easily present summarized data in a top-layer report, and also click through to view more detailed information. Using this feature, only one template has to be designed for each level of drill-down. For more information about this feature, see Section 1.10 - Drill-Down in the Designer Guide.

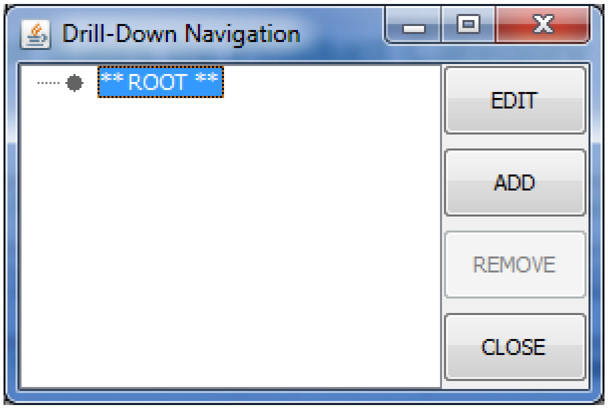
Because the drill-down feature uses parameterized queries, you need to have Woodview database set up (details are in Section Q.3.2.1.1 - Setup a JDBC Connection) in order to do this tutorial.

From the design window, click the *Open* button. Then browse to the `<InstallDir>/help/quickstart/templates` directory. Once you're there, select `QuickStart36.rpt` file and open it. The report will open in the design window.



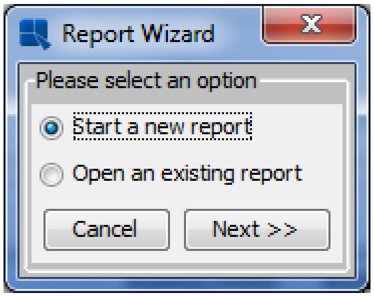
`QuickStart36.rpt` *in Design Window*

The report shows aggregated sales data grouped by product category. Next, we will link this template to another parameterized template, so that users can drill into each category and look at the sales figures for products in each category. To add a layer of drill-down, select Drill-Down → Navigate. This will bring up a dialog showing the hierarchy of all drill-down layers in this report. Because there aren't any layers defined, only ***ROOT*** will be displayed.



Drill-Down Navigation Dialog

To add a new layer of drill-down, click the *ADD* button. This will bring up a dialog asking you if you want to use an existing template or create a new one. If you select to create a new template, you will go back to the data registry where you can start designing a report. You can also use an existing report. In either case, the report you use must have a parameterized query or class as the data source. For more information about this, see Section 1.3.2.2.2 - Parameterized Queries and Section 1.3.5.1 - Parameterized Class Files of the Designer Guide. In this instance, select to open an existing report and click the *Next >>* button.



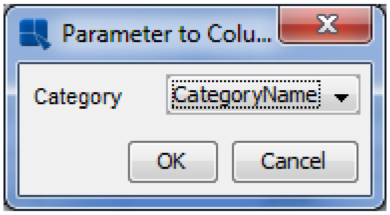
Please select an option

● Start a new report

○ Open an existing report

Report Options Dialog

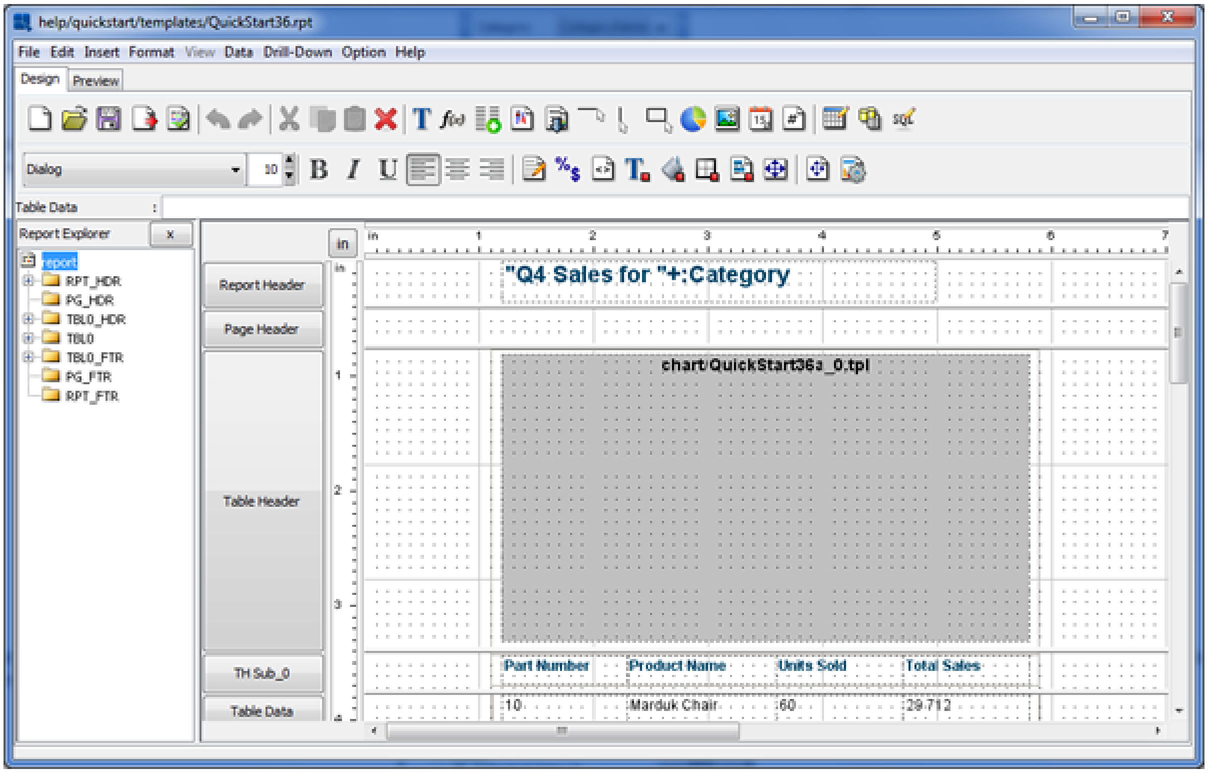
At the prompt, browse to `<InstallDir>/help/quickstart/templates` and select `QuickStart36a.rpt` file. (or `QuickStart36a_Acc.rpt` from the `/Access/` directory). A dialog will then open prompting you to select a column from the primary report to map to the drill-down layer.



Category: CategoryName

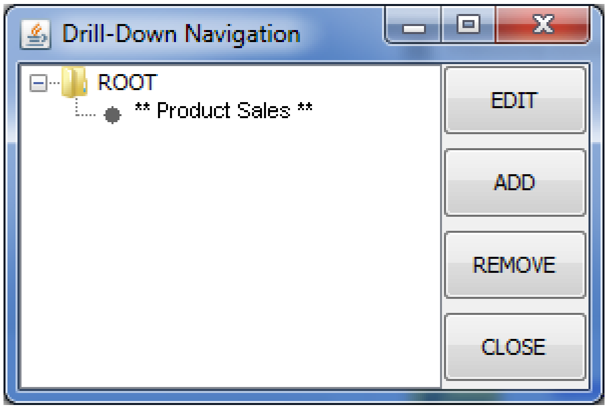
Column - Parameter Mapping Dialog

Select to map the Categoryname column to the parameter and click *OK*. A dialog will open prompting you to specify a display name for the report. Enter any name and click *Ok*. The drill-down layer will now open in the design window.



First Drill-Down Level in Report Designer

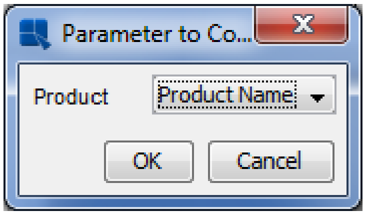
If you preview the report now, you will see how the report is parameterized based on Category name. Back in the design window, select Drill-Down → Navigate again. You will now see a new node for your added level under the ROOT node. The stars "**" indicate which level is currently open in the designer.



Drill-Down Navigation Dialog with Additional Layer

Now we will add one more layer of drill-down to this report that will allow users to drill through the product sales and to see the records of each order for a given product. To do this, select the node for the level you created and click *ADD*.

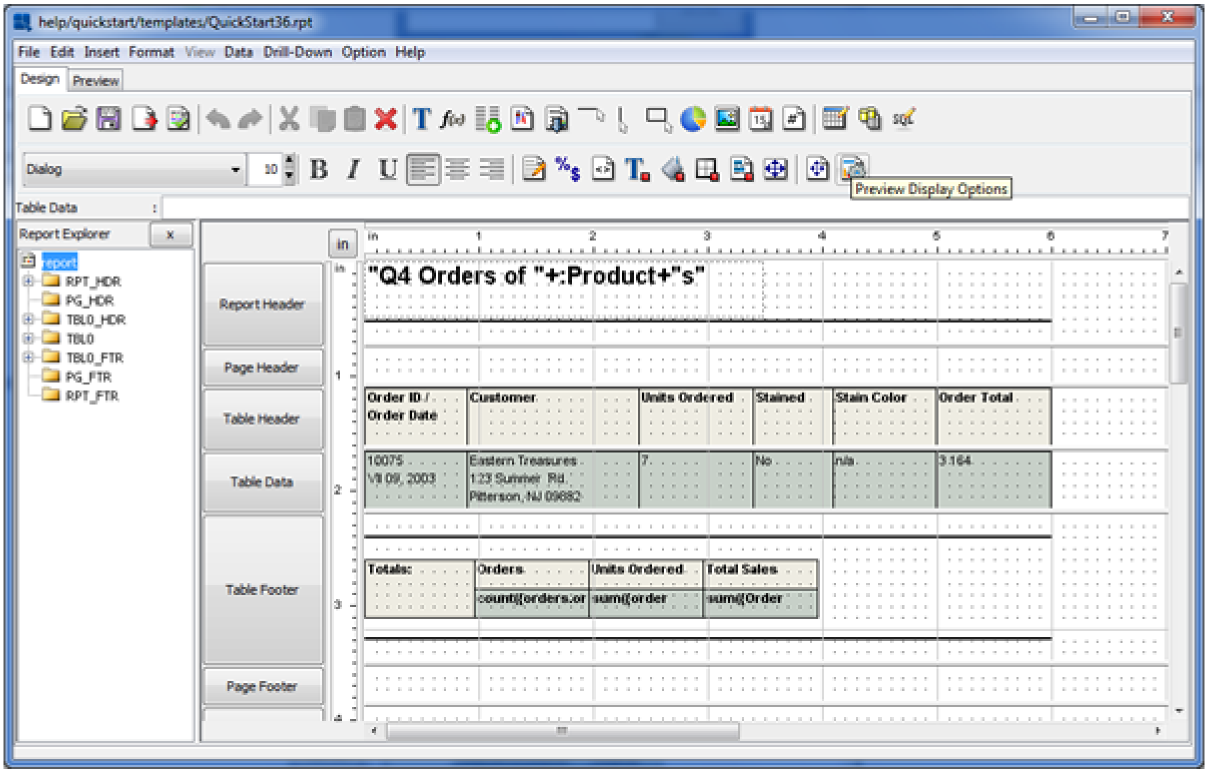
Again, select to open an existing template for the new drill-down layer. At the dialog, browse to `<InstallDir>/help/quickstart/templates` and select `QuickStart36b.rpt` file. (or `QuickStart36b_Acc.rpt` from the `/Access/` directory). This will bring up the parameter to column mapping dialog.



Product: Product Name

Second Parameter Mapping Dialog

Select to map the Product name column to the parameter and click *Ok*. Enter a display name for the new layer and click *Ok* again to open it in the design window.



"Q4 Orders of "+:Product+"s"

Order ID / Order Date	Customer	Units Ordered	Stained	Stain Color	Order Total
10075 VII 09, 2003	Eastern Treasures 123 Summer Rd. Paterson, NJ 09682	7	No	n/a	3,164

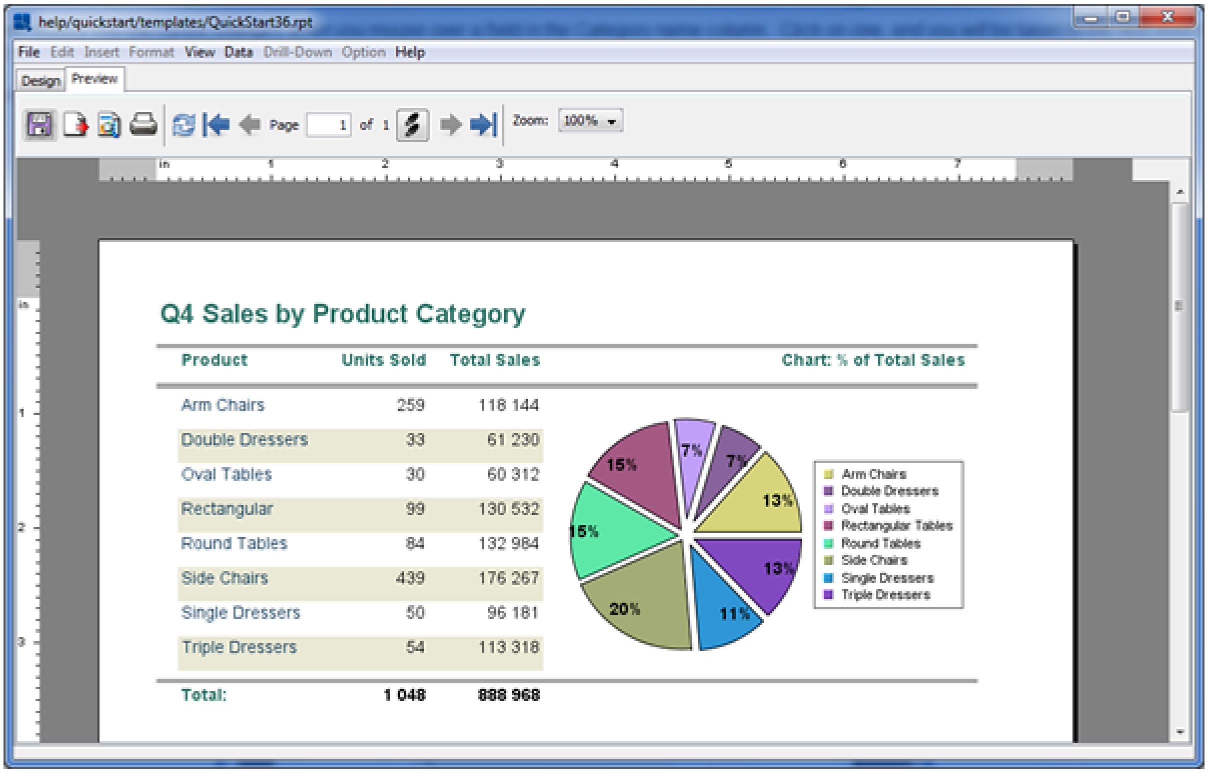
Totals:	Orders	Units Ordered	Total Sales
	count([orders.or	sum([order	sum([Order

Second Drill-Down Level in Report Designer

Now select Drill-Down → Navigate again. Notice that there is a new node for the layer you just added. Select the root report and click the *EDIT* button to open it in the design window. You will see the first report open in the designer. Then click *CLOSE* to dismiss the navigation dialog.

Now preview the report. Notice that the cursor changes when you mouse over a field in the Category name column. Click on one and you will be taken to the next level that shows sales for each product in that category.

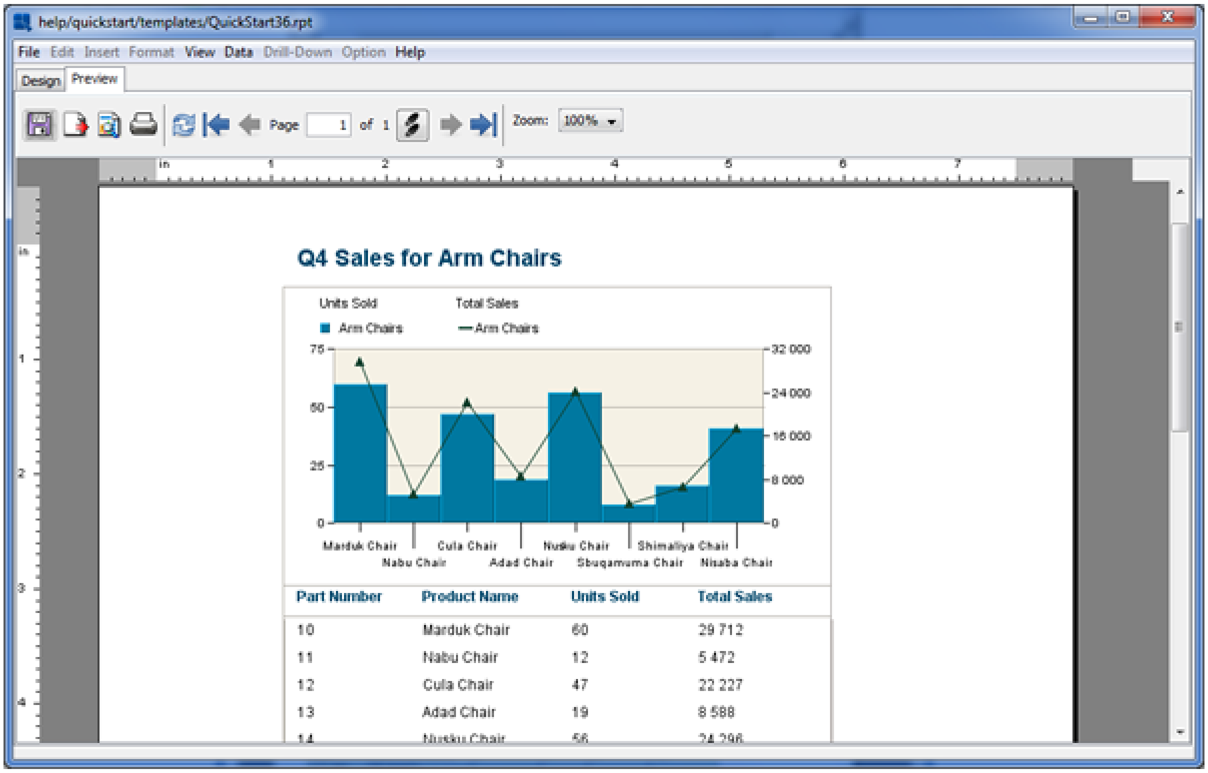
Within the product report, you can click on a field in the Product name column to go to the third level report showing the orders for that particular product.



Q4 Sales by Product Category

Product	Units Sold	Total Sales	Chart: % of Total Sales
Arm Chairs	259	118 144	
Double Dressers	33	61 230	
Oval Tables	30	60 312	
Rectangular	99	130 532	
Round Tables	84	132 984	
Side Chairs	439	176 267	
Single Dressers	50	96 181	
Triple Dressers	54	113 318	
Total:	1 048	888 968	

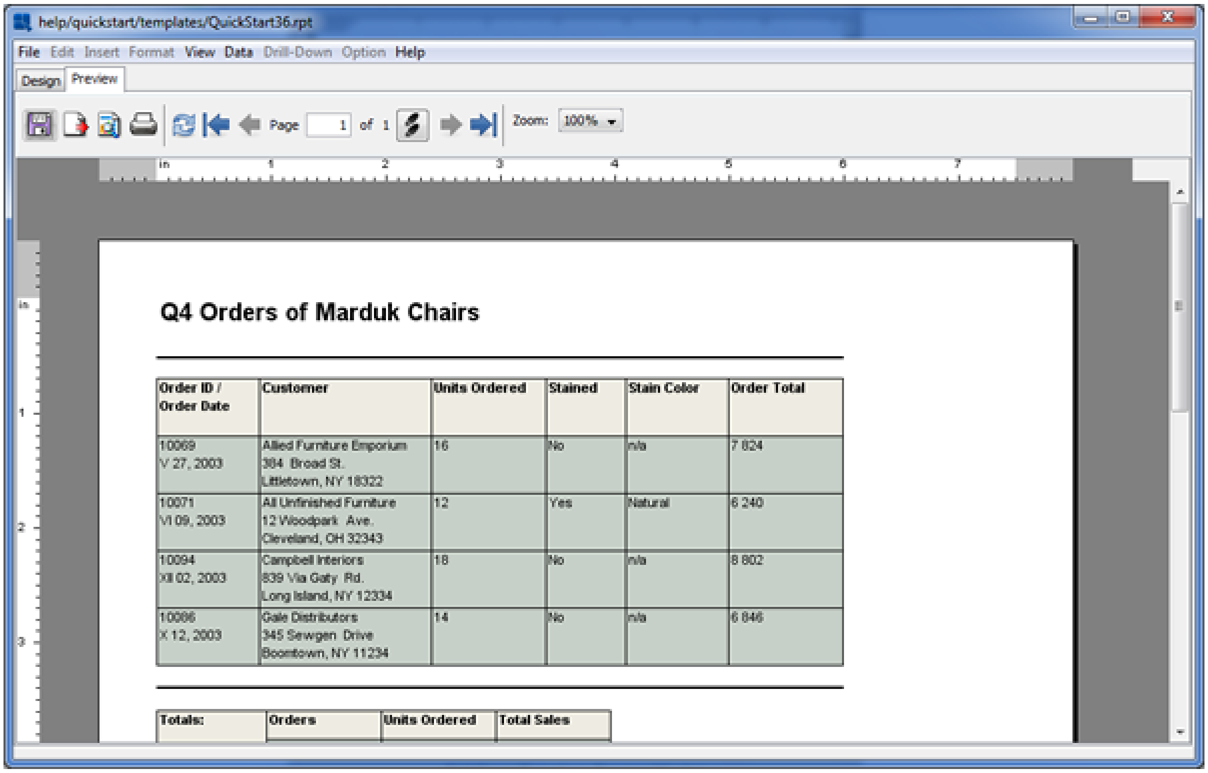
Drill-Down Level 1



Q4 Sales for Arm Chairs

Part Number	Product Name	Units Sold	Total Sales
10	Marduk Chair	60	29 712
11	Nabu Chair	12	5 472
12	Cula Chair	47	22 227
13	Adad Chair	19	8 588
14	Nusku Chair	56	24 296

Drill-Down Level 2


help/quickstart/templates/QuickStart36.rpt
File Edit Insert Format View Data Drill-Down Option Help
Design Preview
Page 1 of 1
Zoom: 100%

Q4 Orders of Marduk Chairs

Order ID / Order Date	Customer	Units Ordered	Stained	Stain Color	Order Total
10069 V 27, 2003	Allied Furniture Emporium 384 Broad St. Littletown, NY 18322	16	No	n/a	7 824
10071 VI 09, 2003	All Unfinished Furniture 12 Woodpark Ave. Cleveland, OH 32343	12	Yes	Natural	6 240
10094 XII 02, 2003	Campbell Interiors 839 Via Gaty Rd. Long Island, NY 12334	18	No	n/a	8 802
10086 X 12, 2003	Gale Distributors 345 Sewgen Drive Boomtown, NY 11234	14	No	n/a	6 846

Totals:	Orders	Units Ordered	Total Sales

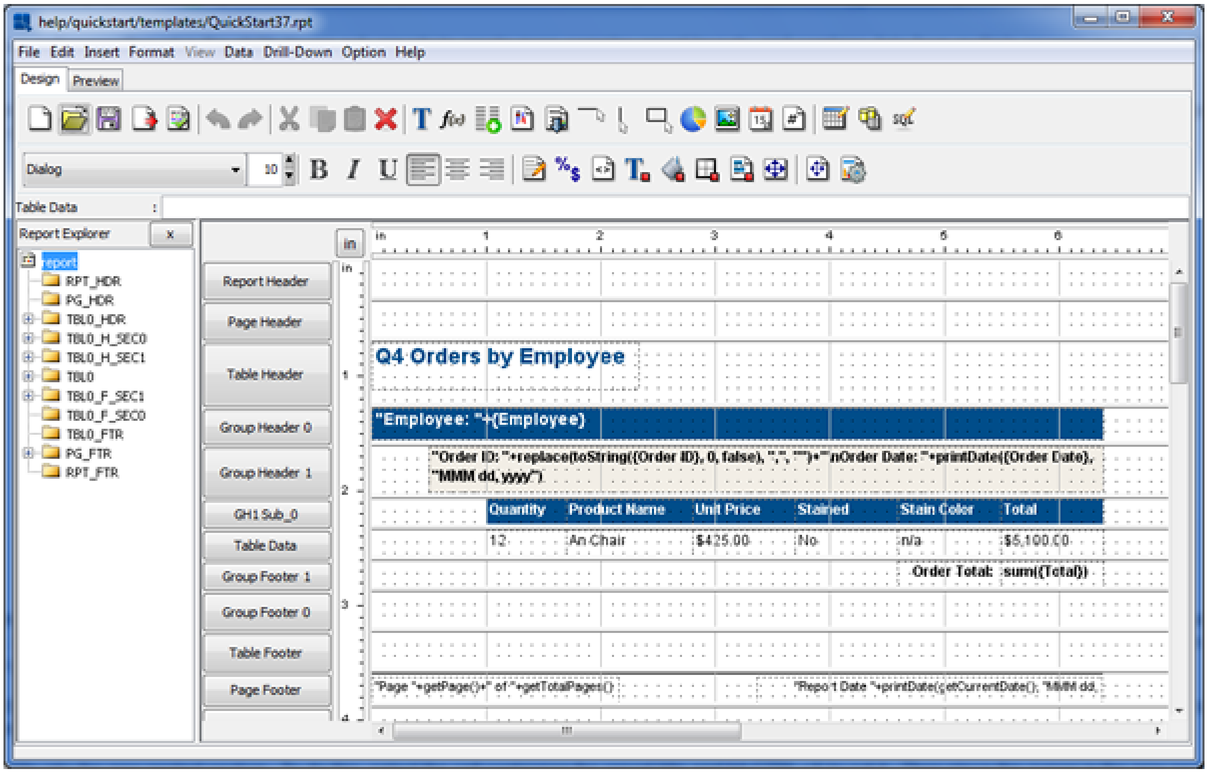
Drill-Down Level 3

To navigate back to higher layers, right click and select *Back* from the pop-up menu. For information on how to deploy drill-down reports, see Section Q.4.7 - Deploy/Export Drill-Down

Q.3.7. Sub-Reports

Another powerful feature in EspressReport is the ability to use sub-reports to create more complex report layouts and combine data from multiple sources in a report. For detailed information about sub-reports, see Section 1.11 - Sub-Reports in the Designer Guide.

From the design window, click the *Open* button. Then browse to `<InstallDir>/help/quickstart/templates` directory. Once there, select `QuickStart37.rpt` file and select to open it. The report will open in the design window.

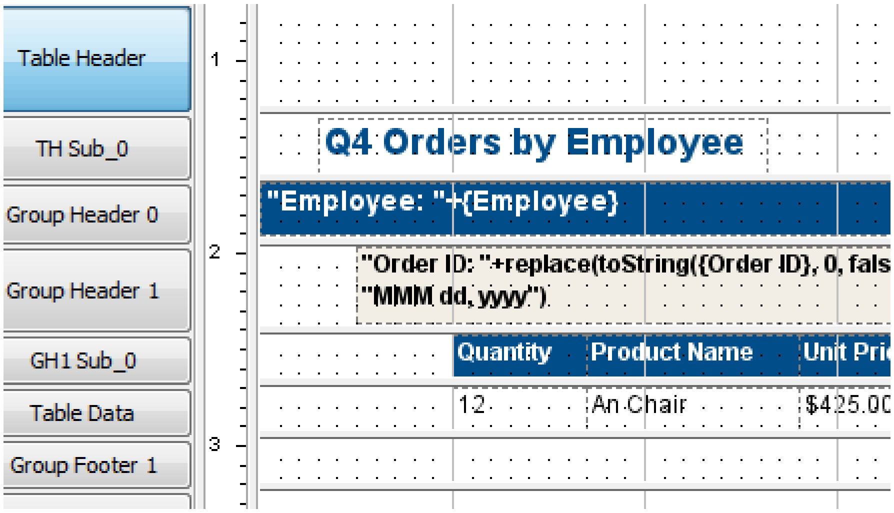


`QuickStart37.rpt` *in Designer*

The report uses two levels of nested grouping to show sales for each employee. We will now add a sub-report to the header that shows aggregated sales by category and employee (in a crosstab layout). Before adding the sub-report, we will create a new report section in which to place the sub-report. Although a sub-report can be placed anywhere in a report, it often makes sense to give a sub-report its own section, especially if the size of the sub-report is not fixed.

To insert a nested section, click the *Table Header* button on the left side to bring up the section options menu. Select *Insert Section*. This will spawn a nested section for the table header. For more about nested sections, see Section 1.5.1.1 - Nested Sections of the Designer Guide.

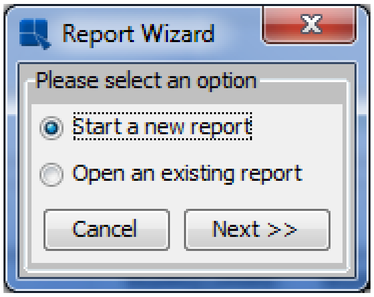
Next we will move the title into the new nested section. To do this, select the cell containing the report title and hit **CTRL**+**X** to cut it. Then place the cursor in the new section and hit **CTRL**+**V** to paste. The cursor will turn to a cross. Position it where you would like the field and click to add it. You may want to resize the new section a bit to fit the cell.



New Report Section

Next, click the *Insert Sub-Report* button on the toolbar: . You will be prompted to save the changes to your report before continuing. Once you do so, your mouse pointer will change to a cross. Press the left mouse button in the top-left corner of the `Table Header` section to insert the SubReport.

As with drill-down, you have the option of creating a new report for the sub-report, or using an existing template. Unlike drill-down, however, the template does not have to have a parameterized query as the data source (unless you want to create linked sub-reports. For more information about this, please see Section 1.11.4 - Linked Sub-Reports of the Designer Guide). Select to open an existing template and click *Next >>*.


Report Wizard

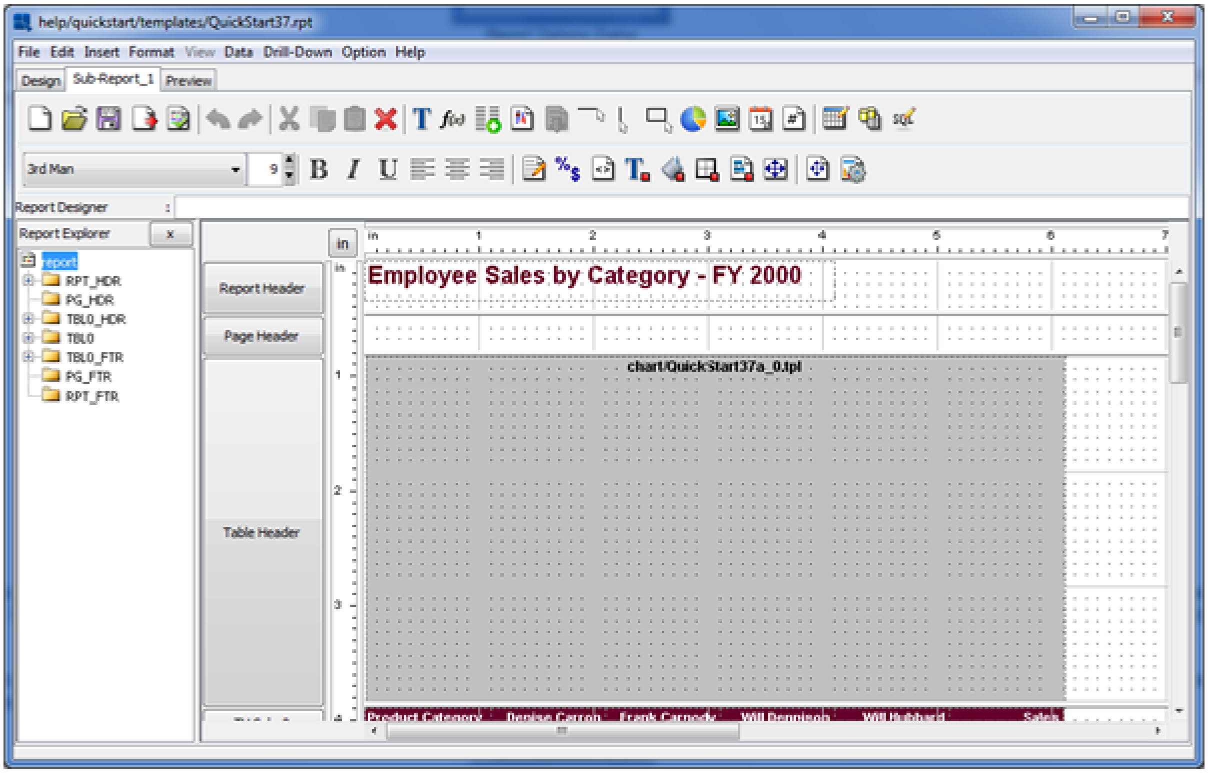
Please select an option

● Start a new report

○ Open an existing report

Report Options Dialog

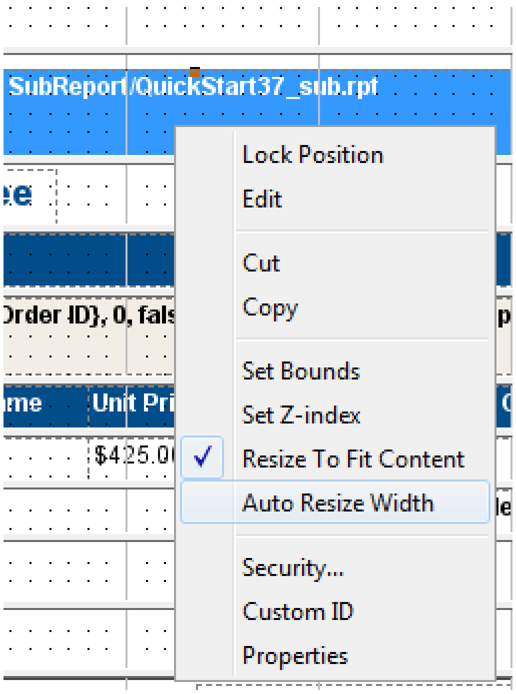
At the prompt, browse to `<InstallDir>/help/quickstart/templates` directory. Once there, select `QuickStart37a.rpt` file and click *Open*. The sub-report will then open in a new tab called Sub-Report_1 in the Report Designer.



Sub-Report in Designer

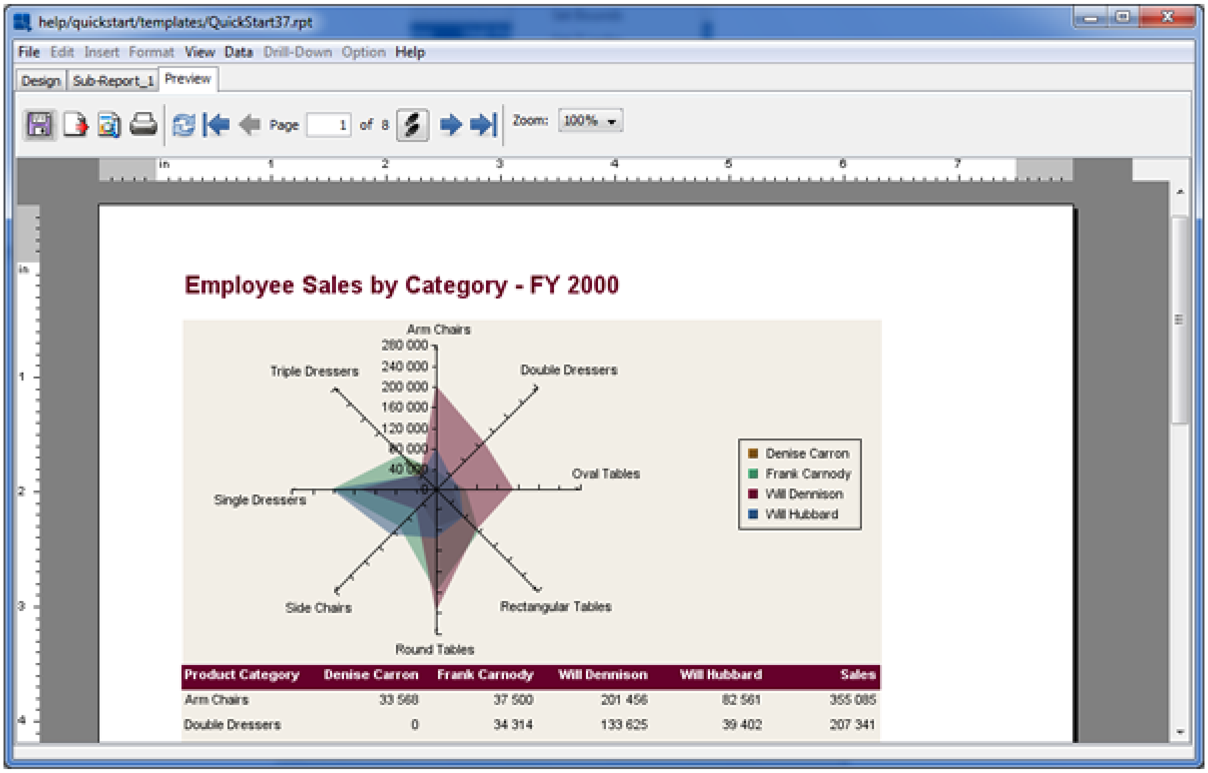
You can preview just the sub-report by toggling between the *Preview* and *Sub-Report* tab. Now go back to the main report by clicking the *Design* tab. The sub-report will appear as a small gray rectangle. Move the rectangle to the upper left corner of the section and click and drag on the horizontal ruler to increase the width of the sub report to about seven inches.

Next, we will set the sub-report to resize dynamically. Right click on the sub-report and make sure the *Resize to fit Content* from the pop-up menu is selected (there should be a ticker next to the option).



Resize to Fit Content Dialog

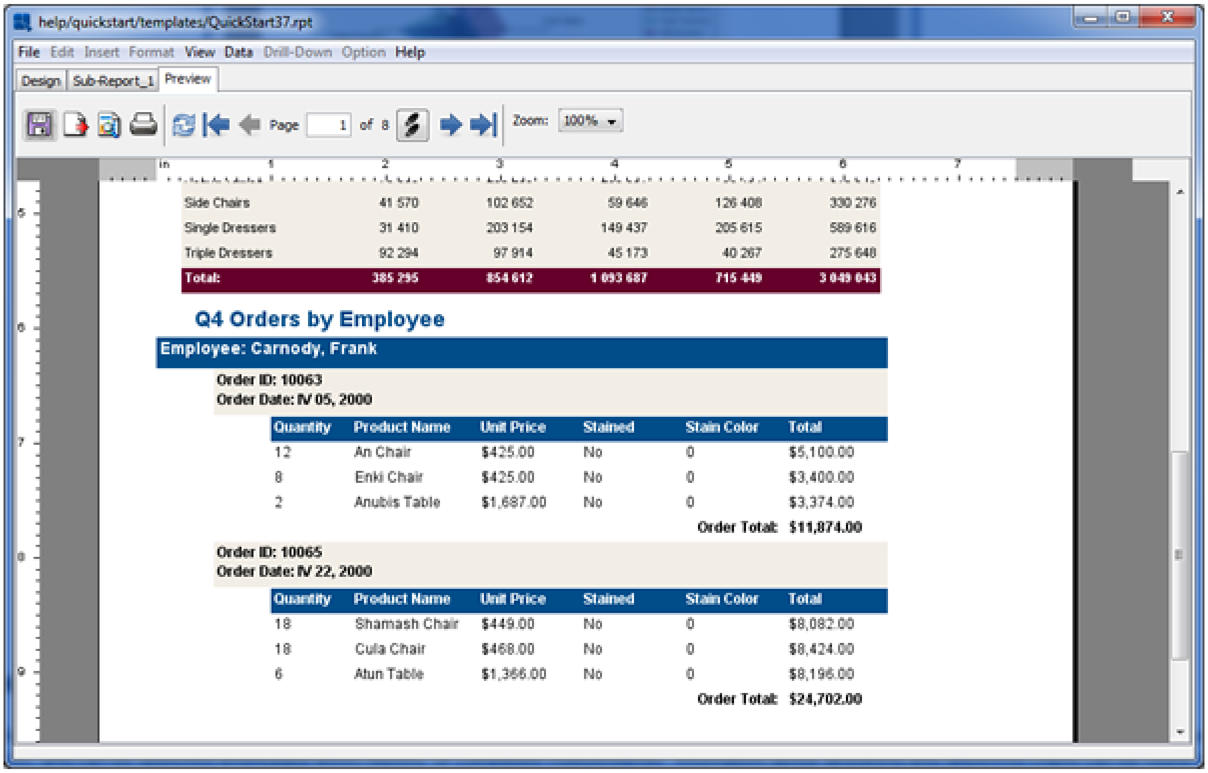
Now preview the report. You can see that the entire sub-report runs before the main report.



Employee Sales by Category - FY 2000

Product Category	Denise Carron	Frank Carnody	Will Dennison	Will Hubbard	Sales
Arm Chairs	33 568	37 500	201 456	82 561	355 085
Double Dressers	0	34 314	133 625	39 402	207 341

Main Report in Preview


help/quickstart/templates/QuickStart37.rpt
File Edit Insert Format View Data Drill-Down Option Help
Design Sub-Report_1 Preview
Page 1 of 8
Zoom: 100%

Side Chairs	41 570	102 652	59 646	126 408	330 276
Single Dressers	31 410	203 154	149 437	205 615	589 616
Triple Dressers	92 294	97 914	45 173	40 267	275 648
Total:	385 295	854 612	1 093 687	715 449	3 049 043

Q4 Orders by Employee

Employee: Carnody, Frank

Order ID: 10063
Order Date: IV 05, 2000

Quantity	Product Name	Unit Price	Stained	Stain Color	Total
12	An Chair	$425.00	No	0	$5,100.00
8	Enki Chair	$425.00	No	0	$3,400.00
2	Anubis Table	$1,687.00	No	0	$3,374.00
				Order Total:	$11,874.00

Order ID: 10065
Order Date: IV 22, 2000

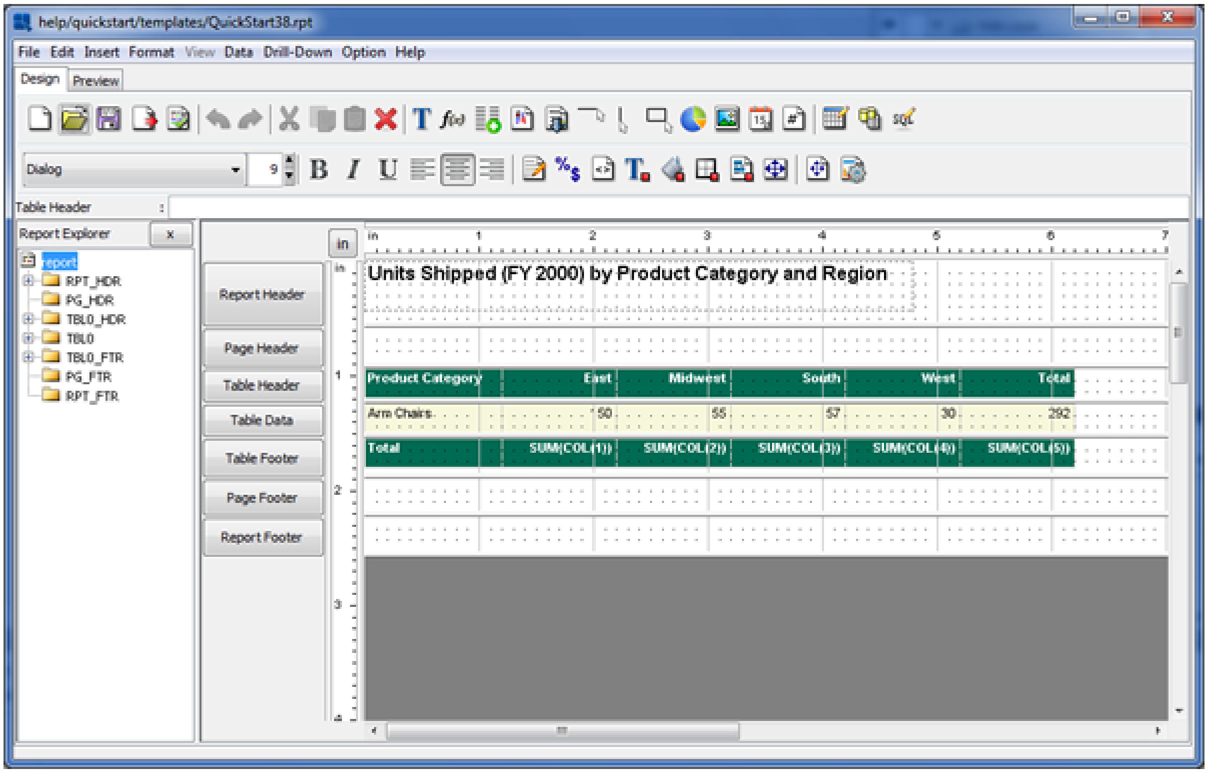
Quantity	Product Name	Unit Price	Stained	Stain Color	Total
18	Shamash Chair	$449.00	No	0	$8,082.00
18	Cula Chair	$468.00	No	0	$8,424.00
6	Atun Table	$1,366.00	No	0	$8,196.00
				Order Total:	$24,702.00

Sub-Report with Main Report in Preview

Q.3.8. Working with Charts

EspressReport includes an extensive charting library that allows you to plot data in over 30 different two-dimensional and three-dimensional chart types. Charts can use independent data sources and can be deployed/embedded within reports or completely independent. Information about designing and deploying charts can be found in the Chapter 3 - Charting Guide.

In this example, we will add a chart to a crosstab report. From the Design window, click the *Open* 📂 button on the toolbar. Then browse to `<InstallDir>/help/quickstart/templates` directory. Once there, select `QuickStart38.rpt` file and open it. The report will open in the Design window.



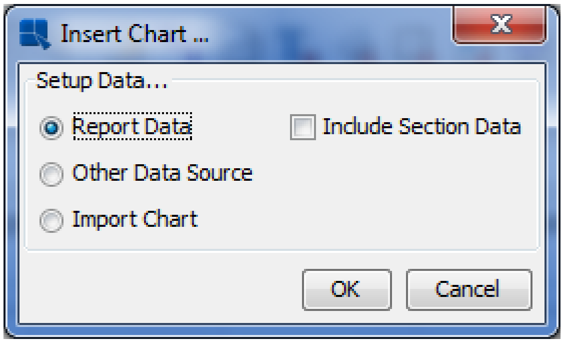
`QuickStart38.rpt` in Design Window

Q.3.8.1. Insert a Chart & Map Data

In this report, we will add a chart below the summaries in the Table Footer. First, you need to make some room for the chart. Right click on the Table Footer and select *Insert Section* from the pop-up menu to insert a new section.

Then enlarge it to add about four inches of space. Next, click the *Insert Chart* button on the toolbar: . A small dotted rectangle will follow your pointer around the Design window. Position it below the "Total" label in the Table Footer section and click. The Chart Designer will then load and guide you through the steps of generating the chart.

The first screen that you see prompts you to specify information about the chart's data source. Charts can retrieve their data from the report or they can have an independent data source.



Insert Chart ...

Setup Data...

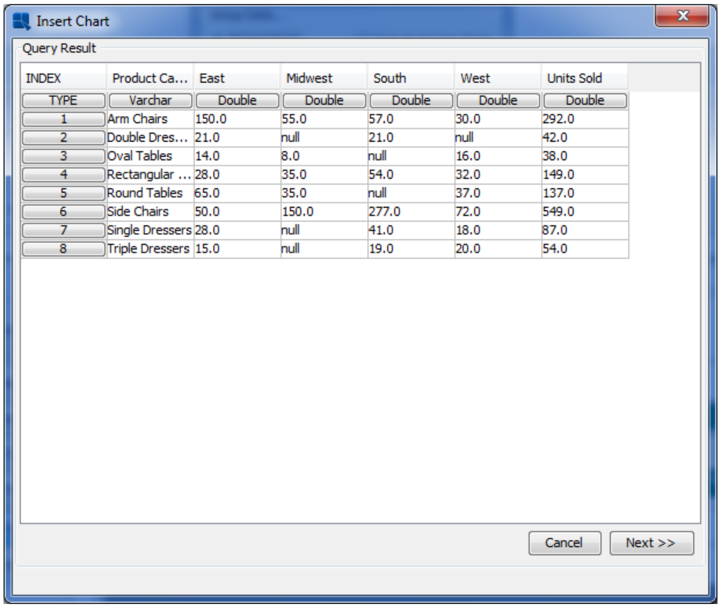
● Report Data □ Include Section Data

○ Other Data Source

○ Import Chart

Chart Data Options

In this case, select to use Report Data for the chart and click *OK*. This will bring up a table showing the report data from which the chart will be drawn.



INDEX	Product Ca...	East	Midwest	South	West	Units Sold
TYPE	Varchar	Double	Double	Double	Double	Double
1	Arm Chairs	150.0	55.0	57.0	30.0	292.0
2	Double Dres...	21.0	null	21.0	null	42.0
3	Oval Tables	14.0	8.0	null	16.0	38.0
4	Rectangular ...	28.0	35.0	54.0	32.0	149.0
5	Round Tables	65.0	35.0	null	37.0	137.0
6	Side Chairs	50.0	150.0	277.0	72.0	549.0
7	Single Dressers	28.0	null	41.0	18.0	87.0
8	Triple Dressers	15.0	null	19.0	20.0	54.0

Chart Data Dialog

Click the *Next >>* button to continue to the next dialog in the Chart Wizard. The next screen allows you to select a chart type. You can toggle between two-dimensional and three-dimensional chart types using the radio buttons. Select a two-dimensional column chart and click the *Next >>* button.

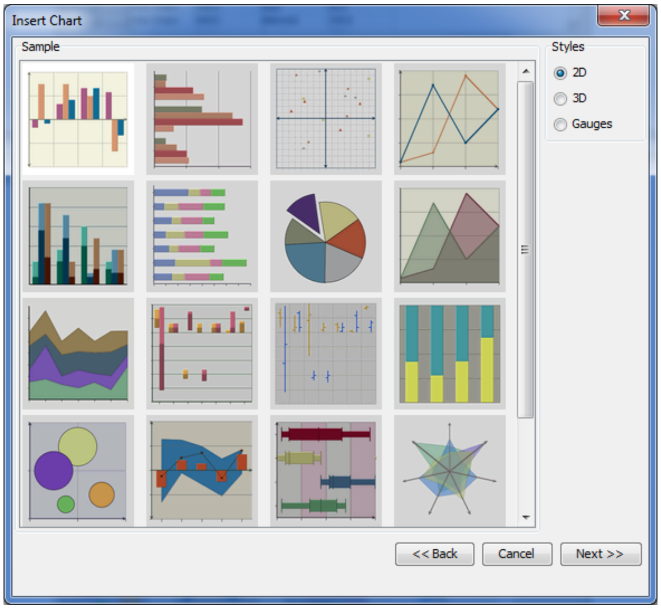
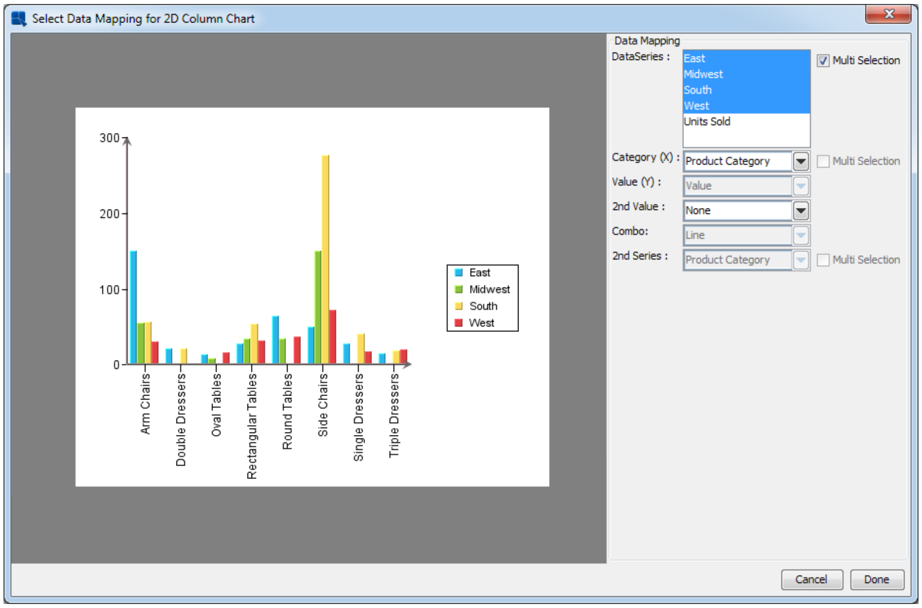


Chart Types Dialog

The next screen allows you to set data mapping for the chart. Data mapping is the process by which the columns of data from your data source are mapped to the elements of the chart. Select the *Multi Selection* option next to the *Data Series* field. The field should change from single-value to multi-value selection box. Use **Ctrl+click** method to select the following four columns as the data series:

```
East
Midwest
South
West
```

Now set the *Category (X)* option to “Product Category”. The dialog should now look like this:



Data Mapping

DataSeries : East, Midwest, South, West, Units Sold — ☑ Multi Selection

Category (X) : Product Category — □ Multi Selection

Value (Y) : Value

2nd Value : None

Combo: Line

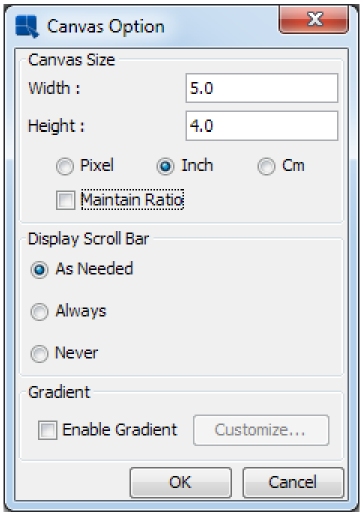
2nd Series : Product Category — □ Multi Selection

Data Mapping Dialog

Once you finish setting up the mapping options, click the *Done* button and you will go to the Chart Designer window where you can customize the chart.

Q.3.8.2. Customize Chart Properties

The first thing we will modify in the chart is the canvas size. This controls the size of the finished chart. To do this, select Format → Canvas. This will bring up a dialog allowing you to modify the chart canvas.



Canvas Size

Width : 5.0

Height : 4.0

○ Pixel ● Inch ○ Cm

□ Maintain Ratio

Display Scroll Bar

● As Needed

○ Always

○ Never

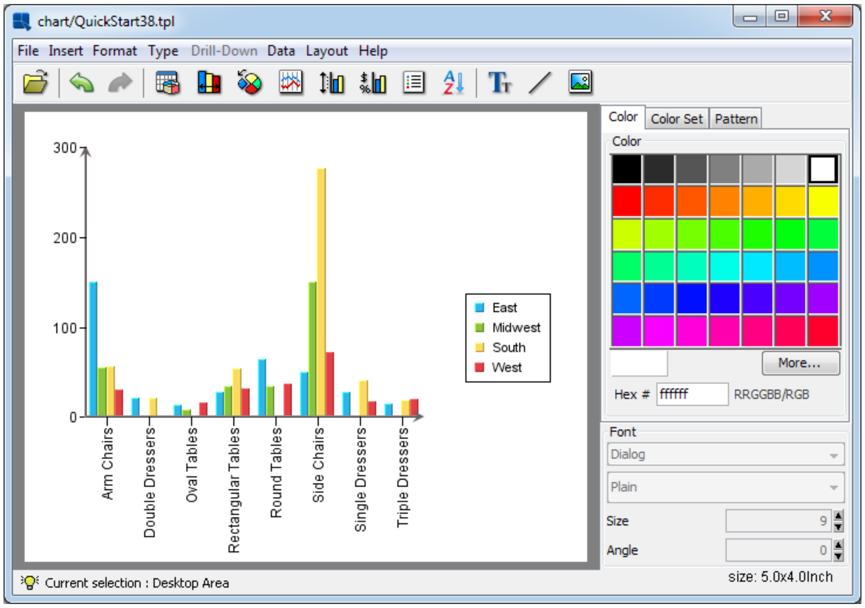
Gradient

□ Enable Gradient

Chart Canvas Dialog

Select the button to specify the measurements in inches, un-select *Maintain Ratio* option and set the new chart canvas size to 5 inches by 4 inches. Click *OK*. This will resize the canvas in the view part of the Chart Designer.

Now that the chart canvas has been resized, the chart plot will appear small and portions of the X-Axis labels may be truncated. This can be adjusted by resizing the chart. You can click and drag on the chart plot to move it or right click and drag to resize it. You can also click and drag to move the legend. Use these options to position the chart plot and legend on the canvas.



Hex # ffffff

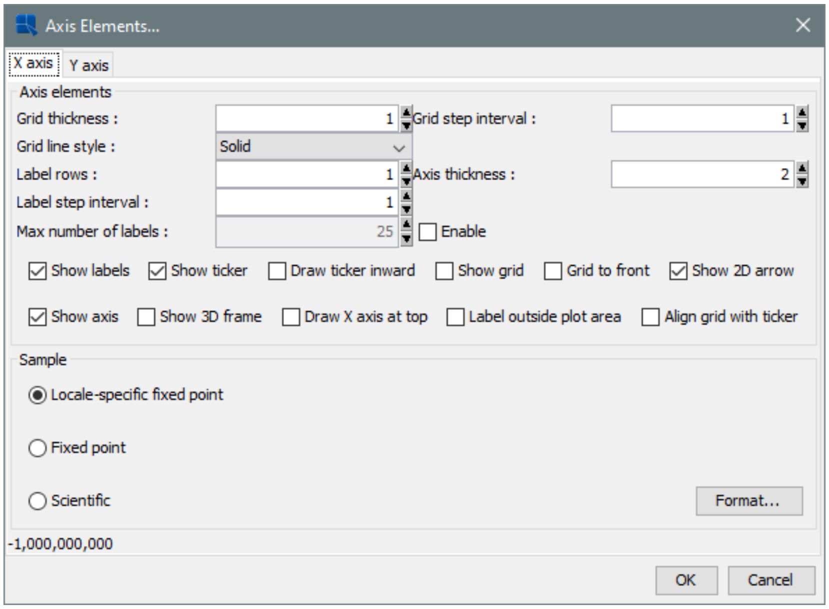
Font: Dialog, Plain

Size: 9

Angle: 0

Chart Designer with Positioned Chart and Legend

Next, you can modify the format of the axis labels. To do this, click the *Axis Elements* button on the toolbar. This will bring up a tabbed dialog allowing you to set different options for each chart axis. Click on the "Y Axis" tab to bring up options for the value axis.



Grid thickness : 1

Grid step interval : 1

Grid line style : Solid

Label rows : 1

Axis thickness : 2

Label step interval : 1

Max number of labels : 25 □ Enable

☑ Show labels ☑ Show ticker □ Draw ticker inward □ Show grid □ Grid to front ☑ Show 2D arrow

☑ Show axis □ Show 3D frame □ Draw X axis at top □ Label outside plot area □ Align grid with ticker

● Locale-specific fixed point

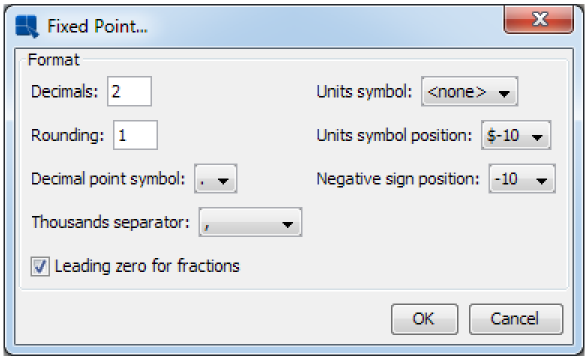
○ Fixed point

○ Scientific

-1,000,000,000

Axis Elements Dialog

Check the box marked *Show grid* to add grid lines to the Y-Axis. Then select *Fixed point* for the data format and click the *Format* button. This will bring up an additional dialog allowing you to set format options for numeric data. Enter number 2 as decimals and click *OK*.



Fixed Point...

Format

Decimals: 2

Units symbol: <none>

Rounding: 1

Units symbol position: $-10

Decimal point symbol: .

Negative sign position: -10

Thousands separator: ,

☑ Leading zero for fractions

Value Axis Fixed Point Formatting Dialog

Click *OK* again to dismiss the axis elements dialog and you will see the specified changes reflected in the chart.

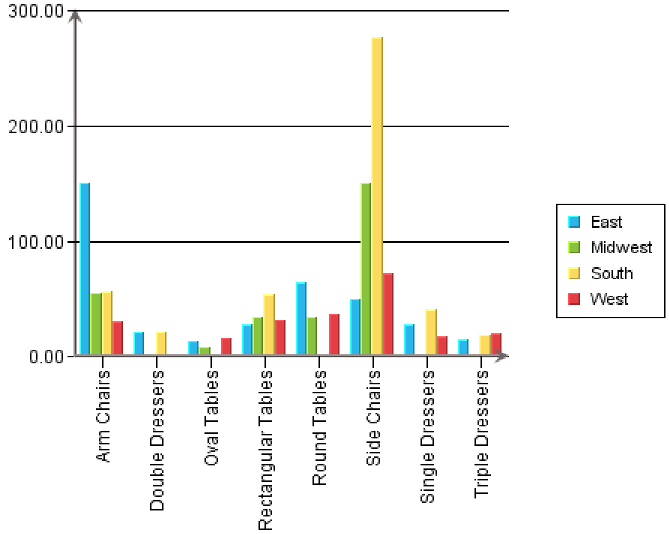
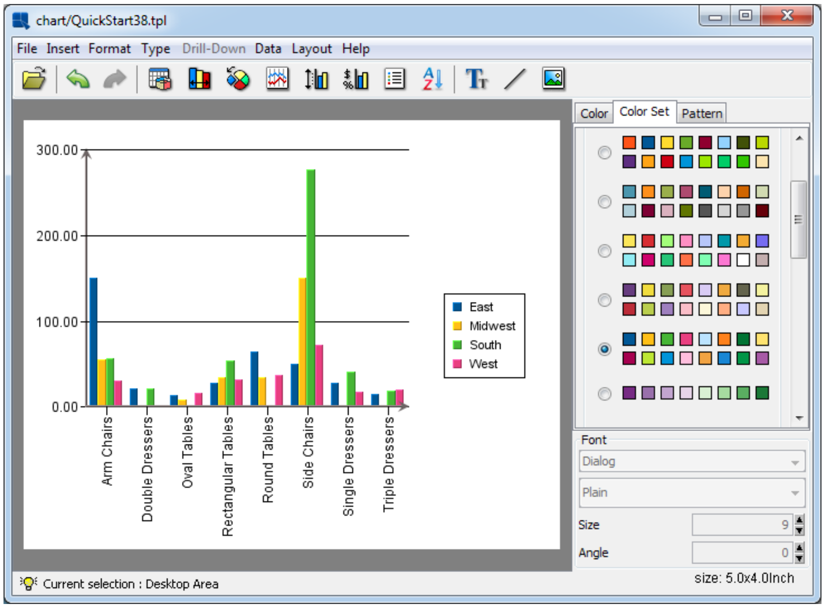


Chart After Axis Element Formatting

To modify colors of the chart elements, you can use one of the preset “Color Sets”. Click on the *Color Set* tab in the right side of the Chart Designer window and choose any color set. This will apply the color set for the data elements. You can also change the color for any element by clicking on it to select it (hint: the Design window will indicate the currently selected element at the lower left corner) and picking a new color from the color panel.



Font

Dialog

Plain

Size 9

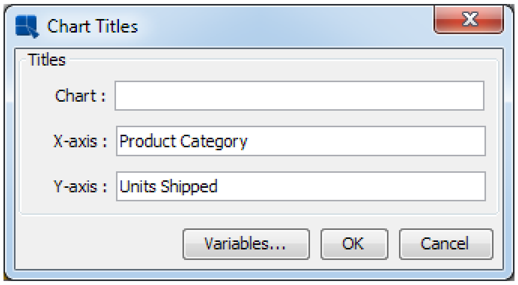
Angle 0

Current selection : Desktop Area

size: 5.0x4.0Inch

Picking a color set

Next, you can add titles to the chart. To do this, select Insert → Titles. This will bring up a dialog allowing you to enter titles for the chart, as well as for each of the axes.



Titles

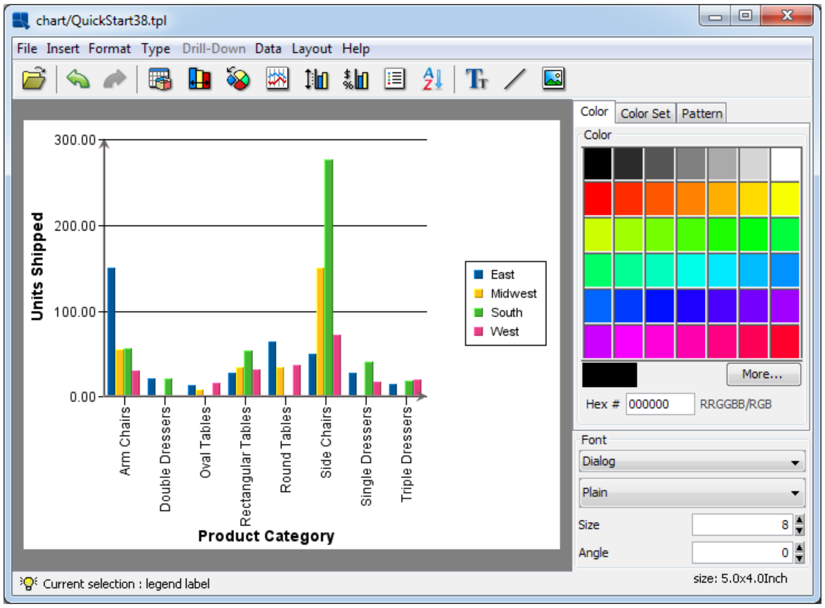
Chart :

X-axis : Product Category

Y-axis : Units Shipped

Chart Titles Dialog

Leave the main title blank and enter titles for both the X and Y axes. The titles will then be added to the chart. Titles are placed automatically but you may need to manually adjust their positions by clicking and dragging the text on the chart canvas.



Hex # 000000 RRGGBB/RGB

Font: Dialog, Plain

Size 8

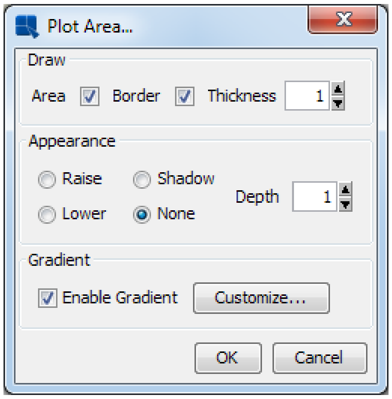
Angle 0

Current selection : legend label

size: 5.0x4.0Inch

Chart with Axis Titles

Finally, you can customize the appearance of the plot area by adding a background and border to the plot. To do this, Format → Plot Area. This will bring up a dialog allowing you to set display options for the chart plot.



Draw

Area ☑ Border ☑ Thickness 1

Appearance

○ Raise ○ Shadow

○ Lower ● None

Depth 1

Gradient

☑ Enable Gradient

Plot Area Dialog

Select to draw both the plot area and the border with a thickness of 1. Specify **None** for the appearance. Choose the *Enable Gradient* option. Once you specify all the options, click *OK* and the plot area for the chart will be modified. You can change the background color of the plot area by clicking on it to select it and then modifying the color in the color panel.

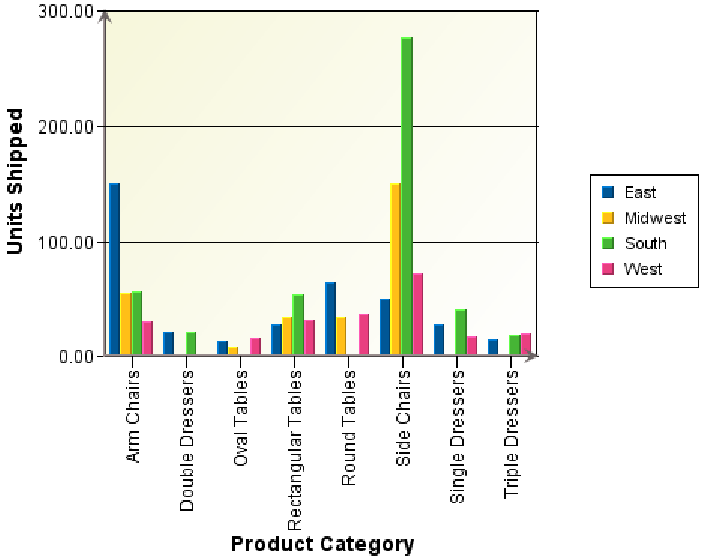
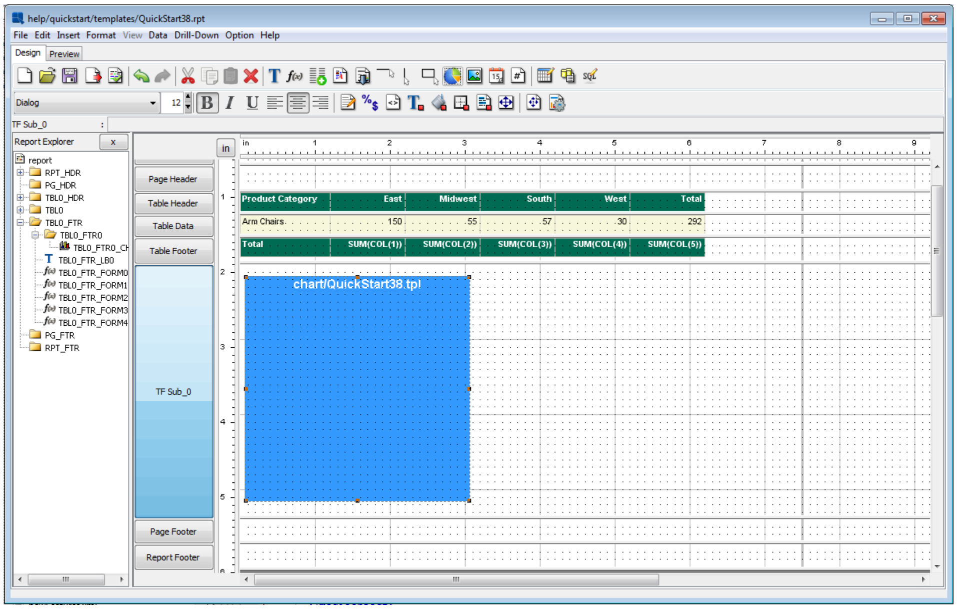


Chart with Plot Area

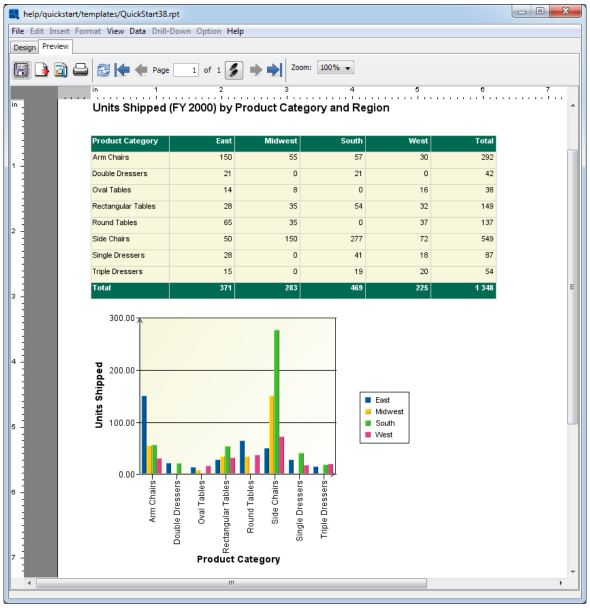
Next, save the changes you have made by clicking the *Save* button on the toolbar (because the chart uses report data, it will be saved automatically in the `/chart/` directory). Then exit the Chart Designer by selecting File → Exit. This will return you to the Design window where a gray rectangle now represents your chart.



Product Category	East	Midwest	South	West	Total
Arm Chairs	150	55	57	30	292
Total	SUM(COL(1))	SUM(COL(2))	SUM(COL(3))	SUM(COL(4))	SUM(COL(5))

Chart in Report Designer

Now preview the report. You will see the chart displayed below the table.



help/quickstart/templates/QuickStart38.rpt

Units Shipped (FY 2000) by Product Category and Region

Product Category	East	Midwest	South	West	Total
Arm Chairs	150	55	57	30	292
Double Dressers	21	0	21	0	42
Oval Tables	14	8	0	16	38
Rectangular Tables	28	35	54	32	149
Round Tables	65	35	0	37	137
Side Chairs	50	150	277	72	549
Single Dressers	28	0	41	18	87
Triple Dressers	15	0	19	20	54
Total	371	283	469	225	1 348

Report with Chart in Preview

Q.4. API Quick Start

This section contains a series of short tutorials designed to illustrate some of the basic features of the Report API. For more details about any of the features described in this section, please see the Programming Guide portion of the documentation.

Q.4.1. Set Up Environment

The code, given in this guide, has been created with the following in mind:

- EspressManager is up and running;

- Tomcat is being used as the servlet container and the web-server;

- The MySQL Woodview database has been set up. The Woodview MySQL dump file named `"Woodview_mysql.sql"` (without the double quotes) located in `<EspressReport installation directory>/help/examples/DataSources/database` directory. Users can reload the dump flie to create a MySQL Woodview database in their environment. This is only necessary for users who wish to use the template with MySQL connection;

Any code given in this guide is also under `<EspressReport installation directory>/help/quickstart/src` directory and any java application class file will be under `<EspressReport installation directory>/help/quickstart/classes` directory. Any HTML files will be under `<EspressReport installation directory>/help/quickstart/html` directory. Similarly, any templates will be under `help/quickstart/templates` directory.

Most files doesn't have to be moved in order to run the tutorials. Instructions will be given in the few instances where the files have to be moved and/or modified.

In order to successfully use the QuickStart API Guide:

- You will need to add `hsqldb.jar` to the CLASSPATH in your EspressManager (Section Q.3.2.1.1 - Setup a JDBC Connection).

- You will need to add `ReportAPIWithChart.jar`, `barbecue.jar`, `javaws.jar`, `jsqlparser.jar`, `ReportDesigner.jar`, `qblicense.jar` and `ExportLib.jar` and `hsqldb.jar` (located in `<EspressReport installation directory>/lib` directory) to your CLASSPATH. In addition, you will also need `servlet.jar` (located in `<Tomcat installation directory>/common/lib` directory) in the CLASSPATH.

- The `ReportAPIWithChart.jar`, `ReportDesigner.jar`, `qblicense.jar`, `ExportLib.jar` and `hsqldb.jar` must also be added in the CLASSPATH of the Tomcat server.

Templates (referenced in this guide) that use a database as a data source, use the HSQL java database. There are alternate templates available for users who use the MySQL Woodview database. These templates can be found under `<EspressReport installation directory>/help/quickstart/templates/MySQL` directory. The templates will again follow the naming convention specified above along with "_mysql" (without the double quotes) before the ".rpt" (without the double quotes) extension.

Depending on the version of the Tomcat server you are using, the servlet context is defined in different ways:

- In Tomcat 6.x or lower version, the servlet context `/servlet/` may be commented out. Please check `<Tomcat installation directory>/conf/web.xml` and uncomment the following lines:

  ```
  <servlet-mapping>
   <servlet-name>invoker</servlet-name>
   <url-pattern>/servlet/*</url-pattern>
  </servlet-mapping>
  ```

  and

  ```
  <servlet>
   <servlet-name>invoker</servlet-name>
   <servlet-class>
    org.apache.catalina.servlets.InvokerServlet
   </servlet-class>
   <init-param>
    <param-name>debug</param-name>
    <param-value>0</param-value>
   </init-param>
   <load-on-startup>2</load-on-startup>
  </servlet>
  ```

  The block of statements is commented if there is <!-- at the beginning and --> at the end. So if the above blocks are commented out, please uncomment and restart your Tomcat server.

- In Tomcat 7.x or higher version, the servlet, i.e., QuickStart423 in this example, needs be explicitly mapped in web.xml:

  ```
  <servlet>
  <servlet-name>QuickStart423</servlet-name>
  <servlet-class>QuickStart423</servlet-class>
  </servlet>

  <servlet-mapping>
  <servlet-name>QuickStart423</servlet-name>
  <url-pattern>/servlet/QuickStart423</url-pattern>
  </servlet-mapping>
  ```

The "classes" (without the double quotes) must also exist in your `<Tomcat installation directory>/webapps/ROOT/WEB-INF` directory. If the directory does not exist, please create it and restart your Tomcat server.

While running the code, you may see a message "Failed to read `espressmanager.cfg`. Use default host and port no. 22071." (without the double quotes). This is not an error message. This message simply states that `espressmanager.cfg` was not found and that it will look for and connect to EspressManager running on the same machine, as the code, and using port number 22071.

Please note that while EspressReport API can be used without EspressManager and in other application servers, the code in this guide was designed with EspressManager and Tomcat in mind. Please refer to the Programming Guide (Section 2.7.3 - Deploying without EspressManager and Section 2.5 - Servlets and Java Server Pages) to switch to other configurations.

The code has also been implemented with the idea that everything (i.e. EspressManager, Tomcat server, and the client) is on the same machine. The code is single-threaded and set to use the same machine for the client and server for demonstration purposes. You can generate multi-threaded code using EspressReport API in a multi-client server environment. To use it in a multi-machine environment, you will have to edit the code (and/or accompanying HTML files). Please refer to the Programming Guide (Section 2.7.2 - Deploying with EspressManager) for further details.

Q.4.2. Run a Report

The following sections show how to run a pre-existing template (`QuickStart42.rpt` located in `<EspressReport installation directory>/help/quickstart/templates` directory) in an application, applet and servlet.

Each section shows the code to generate the report and any steps necessary to deploy.

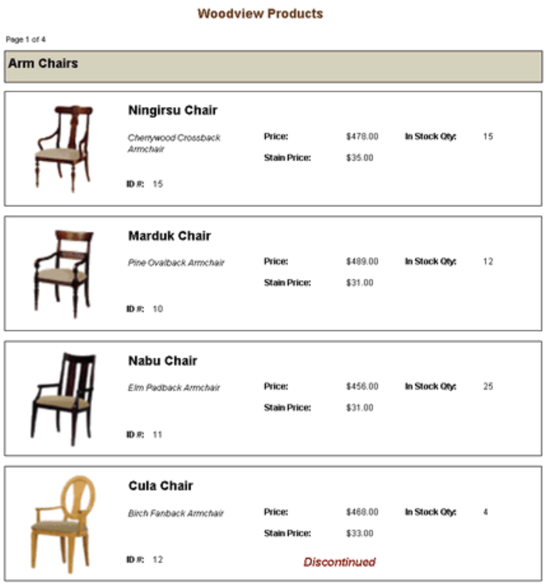
Q.4.2.1. Application

The following code shows how to display an existing report template (in this case `QuickStart42.rpt`) in an application:

```
import java.awt.*;
import java.io.*;
import java.applet.*;
import quadbase.reportdesigner.ReportAPI.*;
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.ReportViewer.*;
import quadbase.reportdesigner.util.*;
import quadbase.reportdesigner.lang.*;

public class QuickStart421 extends Applet {

 public static void main(java.lang.String[] args) {
  try {
   QuickStart421 doReport = new QuickStart421();
   Frame frame = new Frame();
   frame.setLayout(new BorderLayout());
   frame.add("Center", doReport.doQuickStart421(frame));
   frame.setSize(600, 600);
   frame.setVisible(true);
  } catch (Exception ex) {
   ex.printStackTrace();
  }
 }

 public void init() {
  setLayout(new BorderLayout());
  add("Center", doQuickStart421(this));
 }

 Component doQuickStart421(Object object) {
  // Connect to EspressManager
  QbReport.setEspressManagerUsed(true);
  // Create new Report object using specified report
  QbReport report = new QbReport(object,
    "help/quickstart/templates/QuickStart42.rpt");
  // Show the Report
  return (new Viewer().getComponent(report));
 }
}
```

The class file for the above source is located in `<EspressReport installation directory>/help/quickstart/classes` directory.

When the class file is run, the following report is shown using the command **`java QuickStart421`**:



Woodview Products

Page 1 of 4

Arm Chairs

Ningirsu Chair

Cherrywood Crossback Armchair

Price: $478.00 In Stock Qty: 15

Stain Price: $35.00

ID #: 15

Marduk Chair

Pine Ovalback Armchair

Price: $489.00 In Stock Qty: 12

Stain Price: $31.00

ID #: 10

Nabu Chair

Elm Padback Armchair

Price: $456.00 In Stock Qty: 25

Stain Price: $31.00

ID #: 11

Cula Chair

Birch Fanback Armchair

Price: $468.00 In Stock Qty: 4

Stain Price: $33.00

ID #: 12

Discontinued

Report Generated

The main part of the code is in the `doQuickStart421` component. There, a `QbReport` object called `report` is created using the `QuickStart42.rpt` template. The following constructor is used:

```
QbReport(Object parent, String reportTemplatename);
```

Q.4.2.2. Java Web Start Application (JNLP)

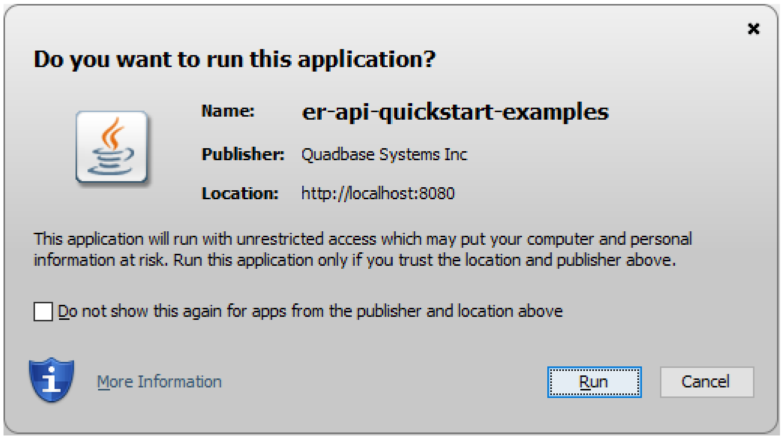
The following JNLP (`QuickStart422.jnlp` in `<EspressReport installation directory>/help/quickstart/html` directory) shows how to display an existing report template (in this case `QuickStart42.rpt`) with launched JNLP file:

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.5" codebase="http://
localhost:8080/" href="QuickStart422.jnlp">
   <information>
      <title>QuickStart422</title>
      <vendor>Quadbase Systems Inc.</vendor>
      <description>QuickStart422 JNLP</description>
      <offline-allowed/>
   </information>
   <security>
      <all-permissions/>
   </security>
   <resources>
 <j2se version="1.7+" max-heap-size="780m"/>
 <jar href="ReportAPIWithChart.jar"/>
 <jar href="SwingReportAPIWithChart.jar"/>
 <jar href="QuickStart.jar" main="true"/>
   </resources>
   <applet-desc
      name="Espress Report QuickStart examples"
      main-class="QuickStart421"
      width="800"
      height="700">
  <param name="filename" value=""/>
     </applet-desc>
</jnlp>
```

The same code used for the application (given in the previous section) can be used to show the report in a launched JNLP file as well. To deploy the JNLP file, copy:

- `QuickStart.jar` from `<EspressReport installation directory>/help/quickstart/classes` directory to `<Tomcat installation>/webapps/ROOT` directory;
- `ReportAPIWithChart.jar`, `SwingReportAPIWithChart.jar` and `qblicense.jar` from `<EspressReport installation directory>/lib` directory to `<Tomcat installation>/webapps/ROOT` directory;
- `QuickStart422.jnlp` from `<EspressReport installation directory>/help/quickstart/html` directory to `<Tomcat installation>/webapps/ROOT` directory.

Type URL "http://localhost:8080/QuickStart422.jnlp" (without the quotes) into a browser and a window pops up asking you whether to run this application:



Do you want to run this application?

Name: er-api-quickstart-examples

Publisher: Quadbase Systems Inc

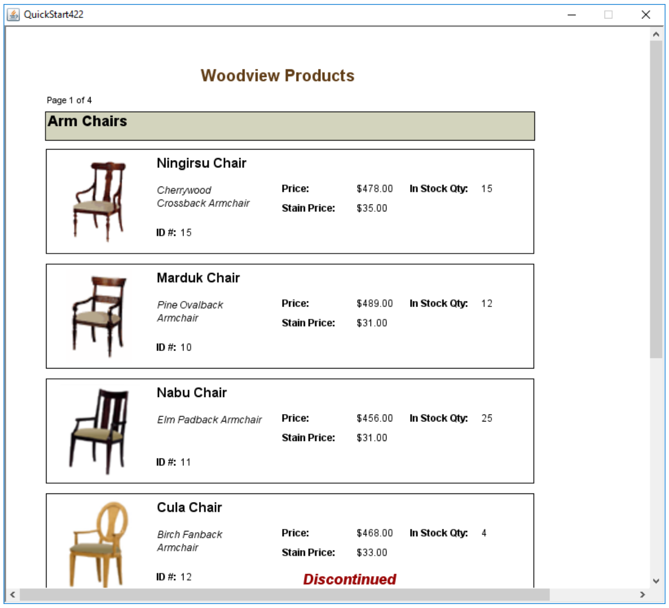
Location: http://localhost:8080

This application will run with unrestricted access which may put your computer and personal information at risk. Run this application only if you trust the location and publisher above.

□ Do not show this again for apps from the publisher and location above

Whether to run JNLP

Click the *Run* button, the jnlp file will run and the following report will be displayed:



QuickStart422

Woodview Products

Page 1 of 4

Arm Chairs

Ningirsu Chair

Cherrywood Crossback Armchair

Price: $478.00 **In Stock Qty:** 15

Stain Price: $35.00

ID #: 15

Marduk Chair

Pine Ovalback Armchair

Price: $489.00 **In Stock Qty:** 12

Stain Price: $31.00

ID #: 10

Nabu Chair

Elm Padback Armchair

Price: $456.00 **In Stock Qty:** 25

Stain Price: $31.00

ID #: 11

Cula Chair

Birch Fanback Armchair

Price: $468.00 **In Stock Qty:** 4

Stain Price: $33.00

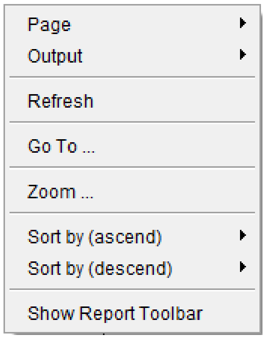
ID #: 12

Discontinued

Report Generated

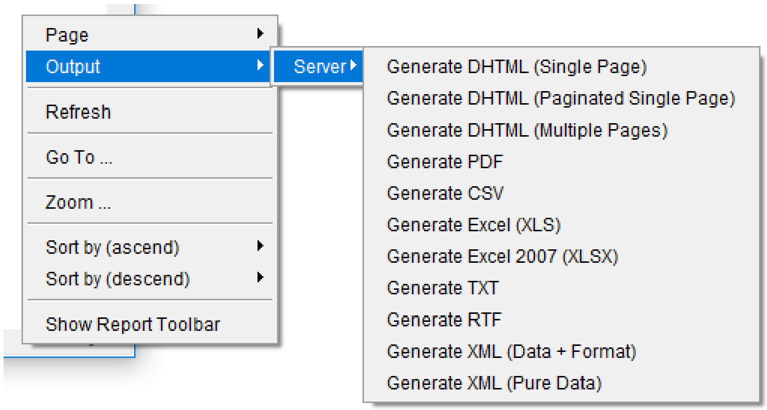
If your server is not localhost, you need to modify `codebase` (in the 2nd line of QuickStart422.jnlp) value with the correct server IP. For your convenience, you can also copy `QuickStart422.html` and `QuickStart422.jsp` from `<EspressReport installation directory>/help/quickstart/html` directory to `<Tomcat installation>/webapps/ROOT` directory. Then type URL "http://localhost:8080/QuickStart422.html" (without the quotes) into a browser, the jnlp will download to your machine, and you can save and open it.

Right clicking on the report displays a Pop-Up Menu:



ReportViewer: Right-Click Pop-Up Menu

Mouse over to the option of "Output" and "Server":



ReportViewer: Export Report in Some Format

In order to Generate Exel(XLS), `poi.jar`, `poi-ooxml.jar`, and `commons-codec.jar` need to be included in the resource section of the jnlp file. Correspondingly, all these jar files need to copy from `<EspressReport installation directory>/lib` directory to `<Tomcat installation>/webapps/ROOT` directory.

In order to Generate Exel 2007(XLSX), `poi.jar`, `poi-ooxml.jar`, `xmlbeans.jar` and `poi-ooxml-schemas.jar` need to be inlcuded in the jnlp file. Also need copy these jars from `<EspressReport installation directory>/lib` directory to `<Tomcat installation>/webapps/ROOT` directory.

Note

The `SwingReportAPIWithChart.jar` is necessary when doing export. It is for dir/file browse dialog to popup.

Q.4.2.3. Servlet

The following code shows how to display an existing report template (in this case `QuickStart42.rpt`) in a servlet:

```
import quadbase.reportdesigner.ReportAPI.*;
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.ReportViewer.*;
import quadbase.reportdesigner.util.*;
import quadbase.reportdesigner.lang.*;
import java.awt.*;
import java.applet.*;
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class QuickStart423 extends HttpServlet {

 public void doGet(HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException {
  System.out.println("Calling QuickStart423....");
  // Set the "content type" header of the response
  res.setContentType("text/html");
  // Get the response's OutputStream to return content to the client.
  OutputStream toClient = res.getOutputStream();
  try {
   // Use EspressManager
```

```
     QbReport.setEspressManagerUsed(true);
     // Open up specified Report
     QbReport report = new QbReport((Applet) null,
       "help/quickstart/templates/QuickStart42.rpt");
     // Export (Stream) the report to DHTML
     report.export(QbReport.DHTML, toClient);
    } catch (Exception e) {
     e.printStackTrace();
    }
    // Flush the outputStream
    toClient.flush();
    // Close the writer; the response is done.
    toClient.close();
   }

   public String getServletInfo() {
    return "QuickStart423 servlet for EspressReport";
   }
  }
```

To deploy the servlet:

- Move `QuickStart423.class` from `<EspressReport installation directory>/help/quickstart/classes` directory to `<Tomcat installation>/webapps/ROOT/WEB-INF/classes` directory.

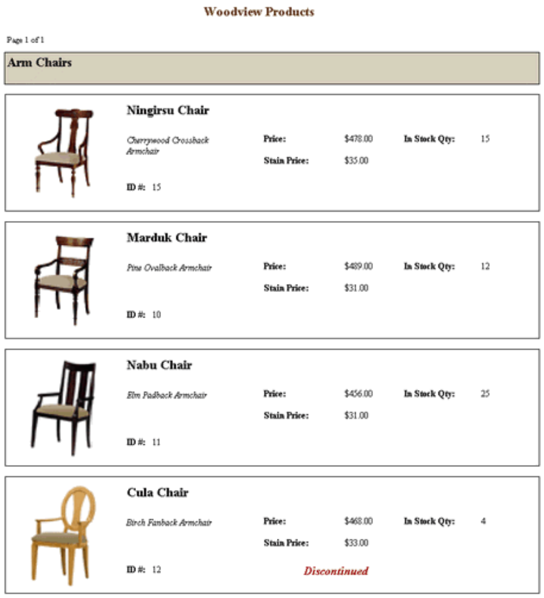
- For Tomcat 7.x or higher version, the servlet, i.e. `QuickStart423` in this example, needs to be explicitly mapped in web.xml:

  ```
  <servlet>
    <servlet-name>QuickStart423</servlet-name>
    <servlet-class>QuickStart423</servlet-class>
  </servlet>
  ```

  and

  ```
  <servlet-mapping>
    <servlet-name>QuickStart423</servlet-name>
    <url-pattern>/servlet/QuickStart423</url-pattern>
  </servlet-mapping>
  ```

When the servlet, using the URL "http://localhost:8080/servlet/QuickStart423" (without the quotes) is run, the following report will be displayed:



Woodview Products

Page 1 of 1

Arm Chairs

Ningirsu Chair

Cherrywood Crossback Armchair

Price: $478.00 In Stock Qty: 15

Stain Price: $35.00

ID #: 15

Marduk Chair

Pine Ovalback Armchair

Price: $489.00 In Stock Qty: 12

Stain Price: $31.00

ID #: 10

Nabu Chair

Elm Padback Armchair

Price: $456.00 In Stock Qty: 25

Stain Price: $31.00

ID #: 11

Cula Chair

Birch Fanback Armchair

Price: $468.00 In Stock Qty: 4

Stain Price: $33.00

ID #: 12 *Discontinued*

Report Generated

The main part of the code is in the `QuickStart423`. There, a `QbReport` object called `report` is created using the `QuickStart42.rpt` template. The following constructor is used:

```
QbReport(Object parent, String reportTemplatename);
```

The `QbReport` object, report, was then exported as a DHTML document to the servlet's response stream using the `export` method:

```
<QbReport object>.export(int exportFormat, OutputStream out);
```

Q.4.2.4. JSP

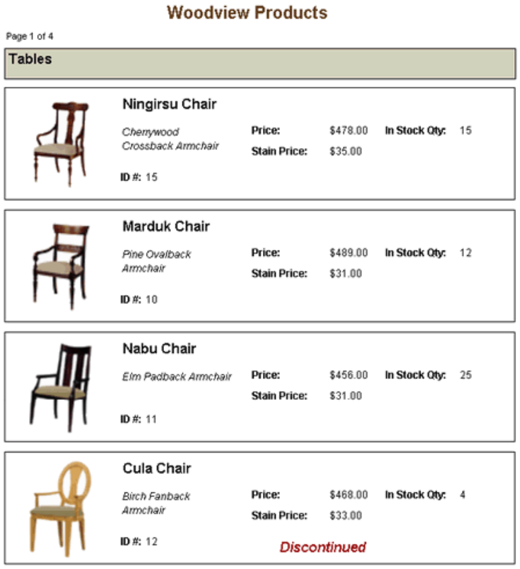
Besides deploying the report using sevlets, it is also possible to deploy it using JSP. An example of deploying reports using JSP/Java Bean technology can be found in `<EspressReport installation directory>/help/examples/jsp/deploy` folder. The same folder contains a `README.txt` file that explains how to setup this example on your servlet container.

Q.4.2.5. Page Viewer

The following piece of code shows how to display the report in an application/applet using Page Viewer instead of Report Viewer.

```
import java.applet.*;
import quadbase.reportdesigner.ReportAPI.*;
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.PageViewer.*;
import quadbase.reportdesigner.util.*;
import quadbase.reportdesigner.lang.*;

public class QuickStart424 extends Applet {
 public static void main(java.lang.String[] args) {
  try {
   QuickStart424 doReport = new QuickStart424();
   Frame frame = new Frame();
   frame.setLayout(new BorderLayout());
   frame.add("Center", doReport.doQuickStart424(frame));
   frame.setSize(600, 600);
   frame.setVisible(true);
  } catch (Exception ex) {
   ex.printStackTrace();
  }
 }

 public void init() {
  setLayout(new BorderLayout());
  add("Center", doQuickStart424(this));
 }

 Component doQuickStart424(Object parent) {
  // Connect to EspressManager
  QbReport.setEspressManagerUsed(true);

  // Location of the report
  String report = "help/quickstart/templates/QuickStart42.rpt";

  // Show the Report
  if (parent instanceof Applet)
   return (new Viewer().getComponent((Applet) parent, report, 0));
  else
   return (new Viewer().getComponent((Frame) parent, report, 0));
 }
}
```

The class file for the above source is located in `<EspressReport installation directory>/help/quickstart/classes` directory. When the class file is run using the command **`java QuickStart424`**, the following report will be displayed:



Woodview Products

Page 1 of 4

Tables

Ningirsu Chair

Cherrywood Crossback Armchair

Price: $478.00 **In Stock Qty:** 15

Stain Price: $35.00

ID #: 15

Marduk Chair

Pine Ovalback Armchair

Price: $489.00 **In Stock Qty:** 12

Stain Price: $31.00

ID #: 10

Nabu Chair

Elm Padback Armchair

Price: $456.00 **In Stock Qty:** 25

Stain Price: $31.00

ID #: 11

Cula Chair

Birch Fanback Armchair

Price: $468.00 **In Stock Qty:** 4

Stain Price: $33.00

ID #: 12

Discontinued

Generated Report

Notice that a `QbReport` object is not created. Instead, the .rpt file name is passed directly into the component:

```
return (new Viewer().getComponent((Applet)parent, report, 0));
```

In the back-end, the .rpt will be exported to VIEW and PAGE files that will be loaded into the viewer. When running without EspressManager, the report will first have to be exported in VIEW format. The resulting VIEW file can then be passed into the Page Viewer component.

The Page Viewer code can also run as a Java Web Start Application. The following jnlp (`QuickStart424.jnlp` in `<EspressReport installation directory>/help/quickstart/html` directory) shows how to display an existing report template (in this case `QuickStart42.rpt`) in the Page Viewer JavaWS application:

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.5" codebase="http://
localhost:8080/" href="QuickStart424.jnlp">
   <information>
      <title>QuickStart424</title>
      <vendor>Quadbase Systems Inc.</vendor>
      <description>QuickStart424 JNLP</description>
      <offline-allowed/>
   </information>
   <security>
      <all-permissions/>
   </security>
   <resources>
 <j2se version="1.7+" max-heap-size="780m"/>
 <jar href="ReportAPIWithChart.jar"/>
 <jar href="PageViewer.jar"/>
 <jar href="QuickStart.jar" main="true"/>
   </resources>
   <applet-desc
      name="Espress Report QuickStart examples"
      main-class="QuickStart424"
      width="800"
      height="700">
  <param name="filename" value=""/>
     </applet-desc>
</jnlp>
```

The same code used for the application (given in the previous section) can also be used to show the report in Java Web Start (JavaWS) Application. To deploy the JavaWS application, copy:

- `QuickStart.jar` from `<EspressReport installation directory>/help/quickstart/classes` directory to `<Tomcat installation>/webapps/ROOT` directory;
- `PageViewer.jar` from `<EspressReport installation directory>/lib` directory to `<Tomcat installation>/webapps/ROOT` directory;
- `ReportAPIWithChart.jar` from `<EspressReport installation directory>/lib` directory to `<Tomcat installation>/webapps/ROOT` directory;
- `QuickStart424.jnlp` from `<EspressReport installation directory>/help/quickstart/html` directory to `<Tomcat installation>/webapps/ROOT` directory.

If your server is not localhost, you need to modify `codebase` (in the 2nd line of QuickStart424.jnlp) value with the correct server IP. For your convenience, you can also copy `QuickStart424.html` and `QuickStart424.jsp` from `<EspressReport installation directory>/help/quickstart/html` directory to `<Tomcat installation>/webapps/ROOT` directory. Then type URL "http://localhost:8080/QuickStart422.html" (without the quotes) into a browser, the jnlp will download to your machine, and you can save and open it.

Q.4.3. Create a Report Programmatically

The following sections show how to create a report programmatically and apply a template `QuickStart43.rpt` (located in `<EspressReport installation directory>/help/quickstart/templates` directory) to the report in an application.

Each section shows the code to generate the report and any steps necessary to deploy.

Q.4.3.1. Map Summary Break ColInfo

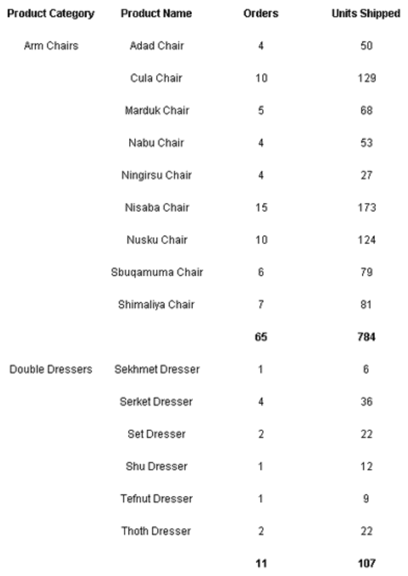
The following code shows how to create a Summary Break report programmatically (using `QuickStart43.txt` as the data source):

```
import java.awt.*;
import java.io.*;
import java.applet.*;
import quadbase.reportdesigner.ReportAPI.*;
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.ReportViewer.*;
import quadbase.reportdesigner.util.*;
import quadbase.reportdesigner.lang.*;

public class QuickStart431 extends Applet {

 // main method for application
 public static void main(java.lang.String[] args) {
  try {
   QuickStart431 doReport = new QuickStart431();
   Frame frame = new Frame();
```

```
    frame.setLayout(new BorderLayout());
    frame.add("Center", doReport.doQuickStart431(frame));
    frame.setSize(600, 600);
    frame.setVisible(true);
   } catch (Exception ex) {
    ex.printStackTrace();
   }
  }

  // init method for applet
  public void init() {
   setLayout(new BorderLayout());
   add("Center", doQuickStart431(this));
  }

  // creates report and return it
  Component doQuickStart431(Object object) {
   // Connect to EspressManager
   QbReport.setEspressManagerUsed(true);
   // Specify Column Mapping
   ColInfo colInfo[] = new ColInfo[4];
   colInfo[0] = new ColInfo(0);
   colInfo[0].setRowBreak(true);
   colInfo[1] = new ColInfo(1);
   colInfo[1].setAggregation(false, ColInfo.NONE);
   colInfo[2] = new ColInfo(2);
   colInfo[2].setAggregation(false, ColInfo.SUM);
   colInfo[3] = new ColInfo(3);
   colInfo[3].setAggregation(false, ColInfo.SUM);
   // Create the QbReport object
   QbReport report = new QbReport(object, QbReport.SUMMARY,
     "help/quickstart/templates/data/QuickStart43.txt", colInfo,
     null);
   // Show the generated report in the Viewer
   return (new Viewer().getComponent(report));
  }
 }
```

The class file for the above source is located in `<EspressReport installation directory>/help/quickstart/classes` directory.

When the class file is run, the following report will be displayed using the command **`java QuickStart431`**:



Product Category	Product Name	Orders	Units Shipped
Arm Chairs	Adad Chair	4	50
	Cula Chair	10	129
	Marduk Chair	5	68
	Nabu Chair	4	53
	Ningirsu Chair	4	27
	Nisaba Chair	15	173
	Nusku Chair	10	124
	Sbuqamuma Chair	6	79
	Shimaliya Chair	7	81
		65	**784**
Double Dressers	Sekhmet Dresser	1	6
	Serket Dresser	4	36
	Set Dresser	2	22
	Shu Dresser	1	12
	Tefnut Dresser	1	9
	Thoth Dresser	2	22
		11	**107**

Report Generated

Please note that when the above code is run, the generated report has default look and no additional formatting.

The main part of the code is in the `doQuickStart431` component. There, a `QbReport` object called `report` is created using specified Column Mapping, specified Report Type, and specified Data Source. The following constructor is used:

```
QbReport(Object parent, int reportType, String dataSource, ColInfo[]
 columnMapping, String reportTemplate);
```

Q.4.3.2. Apply Template

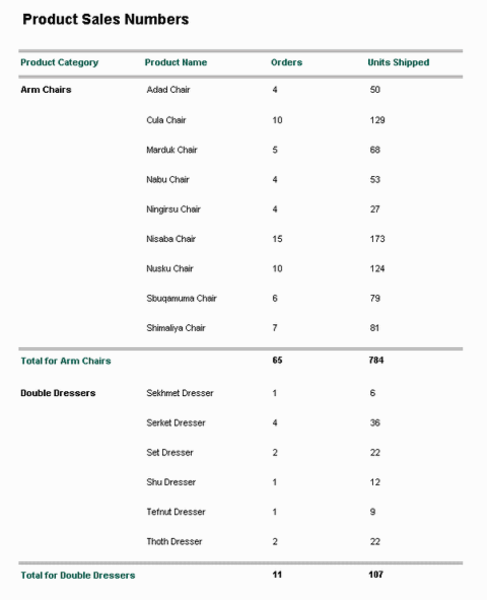
You can apply different formatting by specifying a template during the creation of the `QbReport` object.

The following code shows how to create a Summary Break report programmatically (using `QuickStart43.txt as the data source and QuickStart43.rpt` as the template):

```
import java.awt.*;
import java.io.*;
import java.applet.*;
import quadbase.reportdesigner.ReportAPI.*;
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.ReportViewer.*;
import quadbase.reportdesigner.util.*;
import quadbase.reportdesigner.lang.*;

public class QuickStart432 extends Applet {

 // main method for application
 public static void main(java.lang.String[] args) {
  try {
   QuickStart432 doReport = new QuickStart432();
   Frame frame = new Frame();
   frame.setLayout(new BorderLayout());
   frame.add("Center", doReport.doQuickStart432(frame));
   frame.setSize(600, 600);
   frame.setVisible(true);
  } catch (Exception ex) {
   ex.printStackTrace();
  }
 }

 // init method for applet
 public void init() {
  setLayout(new BorderLayout());
  add("Center", doQuickStart432(this));
 }

 // creates report and return it
 Component doQuickStart432(Object object) {
  // Connect to EspressManager
  QbReport.setEspressManagerUsed(true);
  // Specify Column Mapping
  ColInfo colInfo[] = new ColInfo[4];
  colInfo[0] = new ColInfo(0);
  colInfo[0].setRowBreak(true);
  colInfo[1] = new ColInfo(1);
  colInfo[1].setAggregation(false, ColInfo.NONE);
  colInfo[2] = new ColInfo(2);
  colInfo[2].setAggregation(false, ColInfo.SUM);
  colInfo[3] = new ColInfo(3);
  colInfo[3].setAggregation(false, ColInfo.SUM);
  // Create the QbReport object
  QbReport report = new QbReport(object, QbReport.SUMMARY,
    "help/quickstart/templates/data/QuickStart43.txt", colInfo,
    "help/quickstart/templates/QuickStart43.rpt");
  // Show the generated report in the Viewer
  return (new Viewer().getComponent(report));
 }
}
```

The class file for the above source is located in `<EspressReport installation directory>/help/quickstart/classes` directory.

When the class file is run, the following report will be displayed using the command **`java QuickStart432`**:



Product Sales Numbers

Product Category	Product Name	Orders	Units Shipped
Arm Chairs	Adad Chair	4	50
	Cula Chair	10	129
	Marduk Chair	5	68
	Nabu Chair	4	53
	Ningirsu Chair	4	27
	Nisaba Chair	15	173
	Nusku Chair	10	124
	Sbuqamuma Chair	6	79
	Shimaliya Chair	7	81
Total for Arm Chairs		**65**	**784**
Double Dressers	Sekhmet Dresser	1	6
	Serket Dresser	4	36
	Set Dresser	2	22
	Shu Dresser	1	12
	Tefnut Dresser	1	9
	Thoth Dresser	2	22
Total for Double Dressers		**11**	**107**

Report Generated

The main part of the code is in the `doQuickStart432` component. There, a `QbReport` object called `report` is created using specified Column Mapping, specified Report Type, specified Data Source, and specified Report Template. The following constructor is used:

```
QbReport(Object parent, int reportType, String dataSource, ColInfo[]
 columnMapping, String reportTemplate);
```

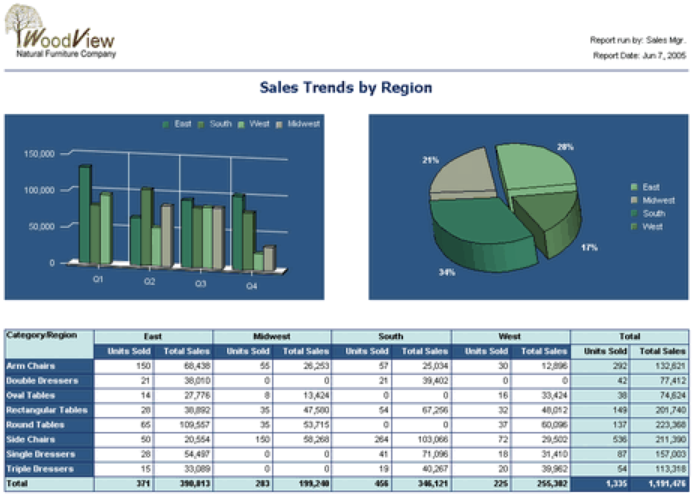
Q.4.4. Modify Data Source of a Report

The following code shows how to modify report's (and its accompanying sub-reports, drill-downs and independent charts) data source without having to create a new `QbReport` object. The `QbReport` object (created from `QuickStart44.rpt`) is opened with backup data (to avoid unnecessary load to the database). The data source is then changed to MySQL Woodview database.

```
import java.awt.*;
import java.io.*;
import java.applet.*;
import quadbase.reportdesigner.ReportAPI.*;
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.ReportViewer.*;
import quadbase.reportdesigner.util.*;
import quadbase.common.util.*;
import quadbase.reportdesigner.lang.*;
import java.sql.*;

public class QuickStart44 extends Applet {

 public static void main(java.lang.String[] args) {

  try {

   QuickStart44 doReport = new QuickStart44();
   Frame frame = new Frame();
   frame.setLayout(new BorderLayout());
   frame.add("Center", doReport.doQuickStart44(frame));
   frame.setSize(600, 600);
   frame.setVisible(true);
  } catch (Exception ex) {
   ex.printStackTrace();
  }
 }

 public void init() {

  setLayout(new BorderLayout());
  add("Center", doQuickStart44(this));
 }

 Component doQuickStart44(Object object) {

  // Connect to EspressManager
  QbReport.setEspressManagerUsed(true);

  // Create the report using the two rows of back-up data
  QbReport report = new QbReport(object,
    "help/quickstart/templates/QuickStart44.rpt", false, false,
    false, true);

  // Begin Code : Specification for new Database
  String newDatabaseURL = "jdbc:mysql://localhost:3306/woodview";
  String newDatabaseDriver = "com.mysql.jdbc.Driver";
  String newDatabaseUserid = "root";
  String newDatabasePassword = "root";
  // End Code : Specification for new Database

  try {

   // Begin Code : Change the data source of the main report and all
   // ancillary templates
   report.getInputData().setAllDatabaseInfo(newDatabaseURL,
     newDatabaseDriver, newDatabaseUserid, newDatabasePassword);
   // End Code : Change the data source of the main report and all
   // ancillary templates
  } catch (Exception ex) {

   ex.printStackTrace();
  }

  return (new Viewer().getComponent(report));
 }
}
```

The class file for the above source is located in `<EspressReport installation directory>/help/quickstart/classes` directory.

When the class file is run, the following report will be displayed using the command **`java QuickStart44`**:



Report run by: Sales Mgr.
Report Date: Jun 7, 2005

Sales Trends by Region

Category/Region	East		Midwest		South		West		Total	
	Units Sold	Total Sales	Units Sold	Total Sales	Units Sold	Total Sales	Units Sold	Total Sales	Units Sold	Total Sales
Arm Chairs	150	68,438	55	26,253	57	25,034	30	12,896	292	132,621
Double Dressers	21	38,010	0	0	21	39,402	0	0	42	77,412
Oval Tables	14	27,776	8	13,424	0	0	16	33,424	38	74,624
Rectangular Tables	28	38,892	35	47,580	54	67,256	32	48,012	149	201,740
Round Tables	65	109,557	35	53,715	0	0	37	60,096	137	223,368
Side Chairs	50	20,554	150	58,268	264	103,066	72	29,502	536	211,390
Single Dressers	28	54,497	0	0	41	71,096	18	31,410	87	157,003
Triple Dressers	15	33,089	0	0	19	40,267	20	39,962	54	113,318
Total	371	390,813	283	199,240	456	346,121	225	255,382	1,335	1,191,476

Report Generated

The main part of the code is in the `doQuickStart44` component. There, a `QbReport` object called `report` is created. The data source for the report, sub-report, drill-down, and chart (with independent data source) is changed using the following method in quadbase.reportdesigner.util.IInputData [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/reportdesigner/util/IInputData.html] interface:

```
setAllDatabaseInfo(String url, String driver, String userid, String
 password);
```

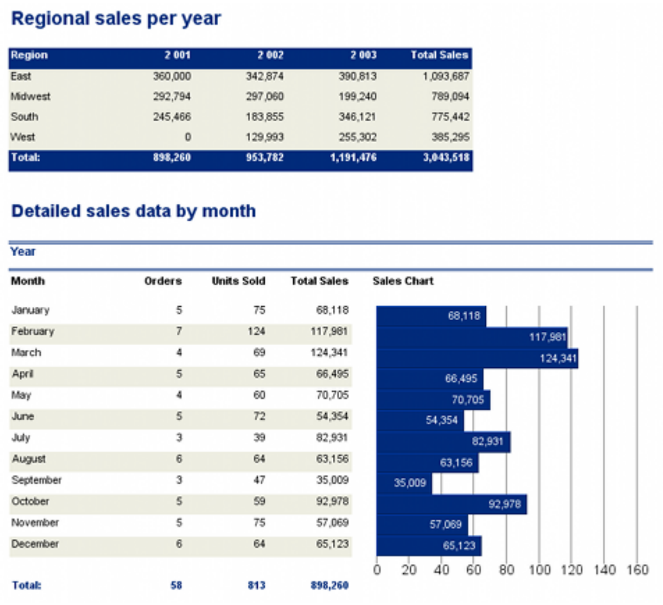
Q.4.4.1. Modify Data Source and Query of a Report (with SubReport)

The following code shows how to modify a report's (and its accompanying sub-reports) data source without having to create a new `QbReport` object. The data source is then changed from MySQL Woodview database to HSQLDB Woodview database. The `QbReport` object (created from `QuickStart441_mysql.pak`) will be opened with backup data, if the HSQLDB database is not hit.

```
import java.awt.*;
import java.io.*;
import java.applet.*;
import quadbase.reportdesigner.ReportAPI.*;
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.ReportViewer.*;
import quadbase.reportdesigner.util.*;
import quadbase.common.util.*;
import quadbase.reportdesigner.lang.*;
import java.sql.*;

public class QuickStart441 extends Applet {

 public static void main(java.lang.String[] args) {

  try {

   QuickStart441 doReport = new QuickStart441();
   Frame frame = new Frame();
   frame.setLayout(new BorderLayout());
   frame.add("Center", doReport.doQuickStart441(frame));
   frame.setSize(600, 600);
   frame.setVisible(true);
  } catch (Exception ex) {
   ex.printStackTrace();
  }
 }

 public void init() {

  setLayout(new BorderLayout());
  add("Center", doQuickStart441(this));
 }

 Component doQuickStart441(Object object) {

  // Connect to EspressManager
  QbReport.setEspressManagerUsed(true);

  // Create the report using the two rows of back-up data
  QbReport report = new QbReport(object,
    "help/quickstart/templates/MySQL/QuickStart441_mysql.pak",
    false, false, false, true);

  // Begin Code : Specification for new Database
  String newDatabaseURL = "jdbc:hsqldb:help/examples/DataSources/database/
woodview";
  String newDatabaseDriver = "org.hsqldb.jdbcDriver";
  String newDatabaseUserid = "sa";
  String newDatabasePassword = "";

  String newDatabaseReportQuery = "select year(o.orderdate) as \"Year
\", month(o.orderdate) as \"Month\", count(o.orderid) as \"Orders\",
 sum(od.quantity) as \"Units Sold\", sum((p.unitprice + od.staincost) *
 od.quantity) as \"Total Sales\" from orders o, order_details od, products
 p where o.orderid = od.orderid and p.productid = od.productid group
 by year(o.orderdate), month(o.orderdate) order by year(o.orderdate),
 month(o.orderdate);";

  String newDatabaseSubReportQuery = "select c.region as \"Region\",
 year(o.orderdate) as \"Year\", sum((p.unitprice + od.staincost) *
 od.quantity) as \"Total Sales\" from customers c, orders o, products
 p, order_details od where c.customerid = o.customerid and o.orderid
 = od.orderid and od.productid = p.productid group by c.region,
 year(o.orderdate);";
  try {
   // Begin Code : Get a handle to the Sub-Report and change its data
   // source
   SubReportObject subReportObject = report.getSubReports()[0];
   QbReport subReport = (QbReport) subReportObject.getSubReport(false,
     false, true, report);

   // Get the query and pass in new database info
   DBInfo newSubReportDatabaseInfo = new DBInfo(newDatabaseURL,
     newDatabaseDriver, newDatabaseUserid, newDatabasePassword,
     newDatabaseSubReportQuery);

   subReport.getInputData().setDatabaseInfo(newSubReportDatabaseInfo);
   // End Code : Get a handle to the Sub-Report and change its data
   // source

   // Begin Code : Change the data source of the main report
   // Get the query and pass in new database info
   DBInfo newReportDatabaseInfo = new DBInfo(newDatabaseURL,
     newDatabaseDriver, newDatabaseUserid, newDatabasePassword,
     newDatabaseReportQuery);

   report.getInputData().setDatabaseInfo(newReportDatabaseInfo);
   // End Code : Change the data source of the main report
  } catch (Exception ex) {

   ex.printStackTrace();
  }

  return (new Viewer().getComponent(report));
 }
}
```

The class file for the above source is located in `<EspressReport installation directory>/help/quickstart/classes` directory.

When the class file is run, the following report will be displayed using the command **`java QuickStart441`**:



Regional sales per year

Region	2 001	2 002	2 003	Total Sales
East	360,000	342,874	390,813	1,093,687
Midwest	292,794	297,060	199,240	789,094
South	245,466	183,855	346,121	775,442
West	0	129,993	255,302	385,295
Total:	898,260	953,782	1,191,476	3,043,518

Detailed sales data by month

Year

Month	Orders	Units Sold	Total Sales
January	5	75	68,118
February	7	124	117,981
March	4	69	124,341
April	5	65	66,495
May	4	60	70,705
June	5	72	54,354
July	3	39	82,931
August	6	64	63,156
September	3	47	35,009
October	5	59	92,978
November	5	75	57,069
December	6	64	65,123
Total:	58	813	898,260

Report Generated

The main part of the code is in the `doQuickStart441` component. There, a `QbReport` object called `report` is created. The data source for the report and the sub-report is changed using the following method in quadbase.reportdesigner.util.IInputData [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/reportdesigner/util/IInputData.html] interface:

```
setDatabaseInfo(IDatabaseInfo db);
```

The sub-report is obtained by getting a handle to the cell containing the sub-report and then calling the sub-report. The sub-report is also opened using its backup data. This is done by:

```
SubReportObject subReportObject = report.getSubReports()[index of particular
 SubReport];
(QbReport)subreport = (QbReport)subReportObject.getSubReport(boolean
 isEnterpriseServer, boolean optimizeMemory, boolean useBackupData, IReport
 report);
```

Please note that if you wish to change the data source from HSQLDB Woodview database to MySQL Woodview database, you will need to change the following lines of code from:

```
QbReport report = new QbReport(object, "help/quickstart/templates/MySQL/
QuickStart441_.pak", false, false, false, true);

// Begin Code : Specification for new Database
String newDatabaseURL = "jdbc:hsqldb:help/examples/DataSources/database/
woodview";
String newDatabaseDriver = "org.hsqldb.jdbcDriver";
String newDatabaseUserid = "sa";
String newDatabasePassword = "";

String newDatabaseReportQuery = "select year(o.orderdate) as \"Year
\", month(o.orderdate) as \"Month\", count(o.orderid) as \"Orders\",
 sum(od.quantity) as \"Units Sold\", sum((p.unitprice + od.staincost) *
 od.quantity) as \"Total Sales\" from orders o, order_details od, products
 p where o.orderid = od.orderid and p.productid = od.productid group
 by year(o.orderdate), month(o.orderdate) order by year(o.orderdate),
 month(o.orderdate);";

String newDatabaseSubReportQuery = "select c.region as \"Region\",
 year(o.orderdate) as \"Year\", sum((p.unitprice + od.staincost) *
 od.quantity) as \"Total Sales\" from customers c, orders o, products
 p, order_details od where c.customerid = o.customerid and o.orderid
 = od.orderid and od.productid = p.productid group by c.region,
 year(o.orderdate);";
```

to

```
QbReport report = new QbReport(object, "help/quickstart/templates/
QuickStart441.pak", false, false, false, true);

// Begin Code : Specification for new Database
String newDatabaseURL = "jdbc:mysql://localhost:3306/woodview";
String newDatabaseDriver = "com.mysql.jdbc.Driver";
String newDatabaseUserid = "root";
String newDatabasePassword = "root";

String newDatabaseReportQuery = "select year(o.orderdate) as \"Year
\", month(o.orderdate) as \"Month\", count(o.orderid) as \"Orders\",
 sum(od.quantity) as \"Units Sold\", sum((p.unitprice + od.staincost)
 * od.quantity) as \"Total Sales\" from orders o, \"order_details\" od,
 products p where o.orderid = od.orderid and p.productid = od.productid
 group by year(o.orderdate), month(o.orderdate) order by year(o.orderdate),
 month(o.orderdate);";
```

```
String newDatabaseSubReportQuery = "select c.region as \"Region\",
 year(o.orderdate) as \"Year\", sum((p.unitprice + od.staincost) *
 od.quantity) as \"Total Sales\" from customers c, orders o, products
 p, \"order_details\" od where c.customerid = o.customerid and o.orderid
 = od.orderid and od.productid = p.productid group by c.region,
 year(o.orderdate);";
```

Q.4.4.2. Modify Data Source and Query of a Parameterized Report (with Parameterized SubReport)

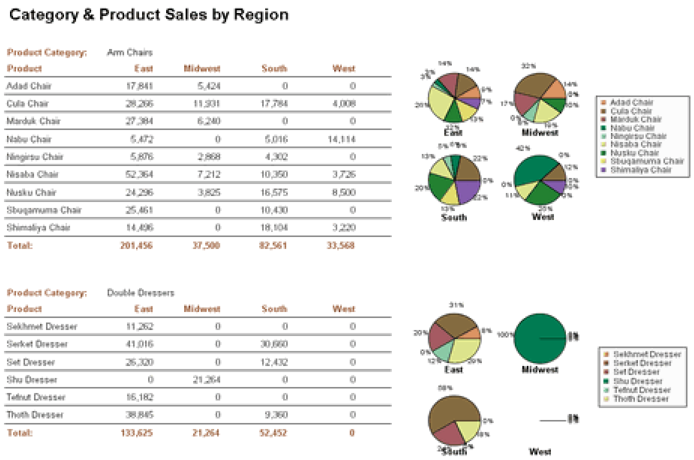
The following code shows how to modify a report's and its accompanying sub-reports parameterized data sources without having to create a new `QbReport` object. The `QbReport` object (created from `QuickStart442_Acc.rpt`) is opened with backup data (to avoid unnecessary load to the database). The datasource is then changed from Access Woodview database to HSQLDB Woodview database.

```
import java.awt.*;
import java.util.*;
import java.io.*;
import java.applet.*;
import java.sql.*;
import quadbase.reportdesigner.ReportAPI.*;
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.ReportViewer.*;
import quadbase.reportdesigner.util.*;
import quadbase.reportdesigner.lang.*;

public class QuickStart442 extends Applet {

 public static void main(java.lang.String[] args) {

  try {
   QuickStart442 doReport = new QuickStart442();
   Frame frame = new Frame();
   frame.setLayout(new BorderLayout());
   frame.add("Center", doReport.doQuickStart442(frame));
   frame.setSize(600, 600);
   frame.setVisible(true);

  } catch (Exception ex) {
   ex.printStackTrace();

  }

 }

 public void init() {

  setLayout(new BorderLayout());
  add("Center", doQuickStart442(this));

 }

 Component doQuickStart442(Object parent) {

  // Connect to EspressManager
  QbReport.setEspressManagerUsed(true);
  // Open the report using the two rows of backup data
  QbReport report = new QbReport(parent,
    "help/quickstart/templates/MySQL/QuickStart442_mysql.rpt",
    false, false, false, true);

  // Begin Code : Specification for new Database
  String URL = "jdbc:hsqldb:help/examples/DataSources/database/woodview";
  String driver = "org.hsqldb.jdbcDriver";
  String userid = "sa";
  String passwd = "";

  String newDatabaseSubReportQuery = "select c.categoryname as \"Category\",
 p.productname as \"Product\", cu.region as \"Region\", sum((od.staincost
 + p.unitprice) * od.quantity) as \"Sales\" from categories c, products p,
 customers cu, orders o, order_details od where c.categoryid = p.categoryid
 and p.productid = od.productid and cu.customerid = o.customerid and
 o.orderid = od.orderid and c.categoryname IN (:category) group by
 c.categoryname, p.productname, cu.region";

  try {

   // Begin Code : Get a handle to the SubReport and change its
   // datasource
   SubReportObject subReportObject = report.getSubReports()[0];
   QbReport subReport = (QbReport) subReportObject.getSubReport(false,
     false, true, report);

   // Begin Code : Get the parameter information of the subreport and
   // pass in the new database information along with the parameter
   // information
   IQueryInParam[] subReportParameters = ((IQueryFileInfo) subReport
     .getInputData().getDatabaseInfo()).getInParam();
   SimpleQueryFileInfo subReportInfo = new SimpleQueryFileInfo(URL,
     driver, userid, passwd, newDatabaseSubReportQuery);
   subReportInfo.setInParam(subReportParameters);

   subReport.getInputData().setDatabaseInfo(subReportInfo);
   // End Code : Get the parameter information of the subreport and
   // pass in the new database information along with the parameter
   // information
   // End Code : Get a handle to the SubReport and change its
   // datasource

   // Begin Code : Get the parameter information of the main report and
   // pass in the new database information along with the parameter
   // information
   // Begin Code : Pass in the parameter values for the main report
   // (which is then picked up by the subreport)
   Vector<String> paramValues = new Vector<String>();
   paramValues.addElement(new String("Arm Chairs"));
   paramValues.addElement(new String("Double Dressers"));
   paramValues.addElement(new String("Round Tables"));
   // End Code : Pass in the parameter values for the main report
   // (which is then picked up by the subreport)

   // Begin Code : Get the parameter properties information and pass in
   // the value of the parameter
```

```
   IQueryInParam[] reportParameters = (IQueryInParam[]) ((IQueryFileInfo)
 report
     .getInputData().getDatabaseInfo()).getInParam();
   ((IQueryMultiValueInParam) reportParameters[0])
     .setValues(paramValues);
   // End Code : Get the parameter properties information and pass in
   // the value of the parameter

   // Get the query for the main report from the report template itself
   // instead of passing in a new query
   SimpleQueryFileInfo reportInfo = new SimpleQueryFileInfo(URL,
     driver, userid, passwd, report.getInputData()
       .getDatabaseInfo().getQuery());
   reportInfo.setInParam(reportParameters);

   report.getInputData().setDatabaseInfo(reportInfo);
   // End Code : Get the parameter information of the main report and
   // pass in the new database information along with the parameter
   // information

  } catch (Exception ex) {
   ex.printStackTrace();

  }

  return (new Viewer().getComponent(report));

 }

}
```

The class file for the above source is located in `<EspressReport installation directory>/help/quickstart/classes` directory.

When the class file is run, the following report will be displayed using the command **`java QuickStart442`**:



Category & Product Sales by Region

Product Category: Arm Chairs

Product	East	Midwest	South	West
Adad Chair	17,841	5,424	0	0
Cula Chair	28,266	11,931	17,784	4,008
Marduk Chair	27,384	6,240	0	0
Nabu Chair	5,472	0	5,016	14,114
Ningirsu Chair	5,876	2,868	4,302	0
Nisaba Chair	52,364	7,212	10,350	3,726
Nusku Chair	24,296	3,825	16,575	8,500
Sbuqamuma Chair	25,461	0	10,430	0
Shimaliya Chair	14,496	0	18,104	3,220
Total:	201,456	37,500	82,561	33,568

Product Category: Double Dressers

Product	East	Midwest	South	West
Sekhmet Dresser	11,262	0	0	0
Serket Dresser	41,016	0	30,660	0
Set Dresser	26,320	0	12,432	0
Shu Dresser	0	21,264	0	0
Tefnut Dresser	16,182	0	0	0
Thoth Dresser	38,845	0	9,360	0
Total:	133,625	21,264	52,452	0

Report Generated

The main part of the code is in the `doQuickStart442` component. There, a `QbReport` object called `report` is created. The data source for the report and the sub-report is changed using the following method in quadbase.reportdesigner.util.IInputData [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/reportdesigner/util/IInputData.html] interface:

```
setDatabaseInfo(IDatabaseInfo db);
```

However, because both reports use a parameterized report, a separate class file (QueryFileInfo) was created, which used the interface quadbase.reportdesigner.util.IQueryFileInfo [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/reportdesigner/util/IQueryFileInfo.html] to pass in the database connection information and the parameter properties information as well. An object of the class QueryFileInfo (with the required information) is then passed as the argument for setDatabaseInfo() method.

The parameter information is obtained by getting the database connection information (by calling the `getDatabaseInfo()` method), casting it to `IQueryFileInfo` and using the following method in `IQueryFileInfo`:

```
public IQueryInParam[] getInParam();
```

The above method call returns complete parameter properties information for all parameters defined in the report.

The code also passes in values for the parameters directly into the report, rather than have the user be prompted to specify the values. The values are entered by going to each parameter and specifying them before passing them to the report object. The following method, in `IQueryInParam`, is used to specify the value:

```
public void setValue(Object value);
```

The sub-report is obtained by getting a handle to the cell containing the sub-report and then calling the sub-report. The sub-report is also opened using its backup data. This is done by:

```
SubReportObject subReportObject = report.getSubReports()[index of particular
 SubReport];

(QbReport)subreport = (QbReport)subReportObject.getSubReport(boolean
 isEnterpriseServer, boolean optimizeMemory, boolean useBackupData, IReport
 report);
```

Please note that if you wish to change the data source from HSQLDB Woodview database to MySQL Woodview database, you will need to change the following lines of code from:

```
QbReport report = new QbReport(object, "help/quickstart/templates/MySQL/
QuickStart442_mysql.rpt", false, false, false, true);

// Begin Code : Specification for new Database
String newDatabaseURL = "jdbc:hsqldb:help/examples/DataSources/database/
woodview";
String newDatabaseDriver = "org.hsqldb.jdbcDriver";
String newDatabaseUserid = "sa";
String newDatabasePassword = "";

String newDatabaseSubReportQuery = "select c.categoryname as \"Category\",
 p.productname as \"Product\", cu.region as \"Region\", sum((od.staincost
 + p.unitprice) * od.quantity) as \"Sales\" from categories c, products p,
 customers cu, orders o, order_details od where c.categoryid = p.categoryid
 and p.productid = od.productid and cu.customerid = o.customerid and
 o.orderid = od.orderid and c.categoryname IN (:category) group by
 c.categoryname, p.productname, cu.region;";
```

to

```
QbReport report = new QbReport(object, "help/quickstart/templates/
QuickStart442.rpt", false, false, false, true);

// Begin Code : Specification for new Database
String newDatabaseURL = "jdbc:mysql://localhost:3306/woodview";
String newDatabaseDriver = "com.mysql.jdbc.Driver";
String newDatabaseUserid = "root";
String newDatabasePassword = "root";

String newDatabaseSubReportQuery = "select c.categoryname as \"Category\",
 p.productname as \"Product\", cu.region as \"Region\", sum((od.staincost
 + p.unitprice) * od.quantity) as \"Sales\" from categories c, products p,
 customers cu, orders o, order_details od where c.categoryid = p.categoryid
 and p.productid = od.productid and cu.customerid = o.customerid and
 o.orderid = od.orderid and c.categoryname IN (:category) group by
 c.categoryname, p.productname, cu.region;";
```

You can also pass in a `java.sql.Connection object` instead of passing in the URL, driver, userid and password of the database. For example, if you wish to pass in a `Connection` object, `conn`, you would need to change the following lines of code:

```
SimpleQueryFileInfo subReportInfo = new SimpleQueryFileInfo(URL, driver,
 userid, passwd, newDatabaseSubReportQuery);
SimpleQueryFileInfo reportInfo = new SimpleQueryFileInfo(URL, driver,
 userid, passwd, report.getInputData().getDatabaseInfo().getQuery());
```

to

```
SimpleQueryFileInfo subReportInfo = new SimpleQueryFileInfo(conn,
 newDatabaseSubReportQuery);
SimpleQueryFileInfo reportInfo = new SimpleQueryFileInfo(conn,
 report.getInputData().getDatabaseInfo().getQuery());
```

Q.4.5. Modify Report Elements

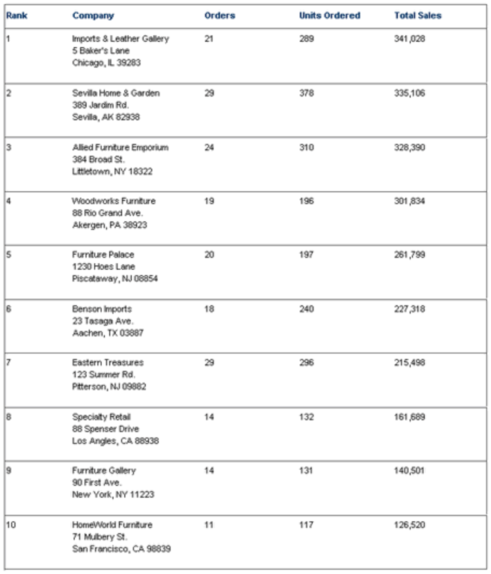
The following code shows how to modify certain elements of the report programmatically. The `QbReport` object is created from `QuickStart45.rpt`. When the code is run, the original report is first shown and when the *Change* button (located at the bottom) is clicked, some of the report elements will change.

```
import java.awt.*;
import java.io.*;
import java.applet.*;
import java.util.*;
import quadbase.reportdesigner.ReportAPI.*;
```

```java
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.ReportViewer.*;
import quadbase.reportdesigner.util.*;
import quadbase.reportdesigner.lang.*;
import quadbase.reportdesigner.report.Formula;

// Set up Frame
public class QuickStart45 extends Frame {
 QbReport report;
 Viewer viewer;
 Component reportComponent;
 Button b = new Button("Change");

 // Start Frame
 public QuickStart45() {
  start();
 }

 // Create Empty Report and set up initial data and wipe out empty report
 // data
 public void start() {

  // Connect to EspressManager
  QbReport.setEspressManagerUsed(true);
  setLayout(new BorderLayout());
  // Create QbReport object
  report = new QbReport((Applet) null,
    "help/quickstart/templates/QuickStart45.rpt");
  viewer = new Viewer();
  reportComponent = viewer.getComponent(report);
  add("Center", reportComponent);
  add("South", b);
 }

 // What to do with the button
 public boolean action(Event e, Object o) {
  // Begin Code : Dual color
  int numberOfColumns = report.getTable().getColumnCount();
  for (int i = 0; i < numberOfColumns; i++) {
   ReportColumn column = report.getTable().getColumn(i);
   column.setAlternateRow(1);
   column.setBgColor2(new Color(245, 245, 238));
   column.setFontColor2(new Color(0, 0, 0));
   column.setFont2(new Font("Dialog", Font.PLAIN, 8));
  }
  // End Code : Dual color

  // Begin Code : Add title to Report Header
  ReportCell title = new ReportCell();
  title.setText("Top 10 Customers");
  title.setBgColor(new Color(255, 255, 255));
  title.setFontColor(new Color(0, 54, 100));
  title.setFont(new Font("Dialog", Font.BOLD, 14));
  title.setAlign(IAlignConstants.ALIGN_LEFT);
  title.setWidth(2.1);
  title.setHeight(0.4);
  title.setX(0);
  title.setY(0);
  report.getReportHeader().addData(title);
  // End Code : Add title to Report Header

  // Begin Code :Add a formula
  ReportCell formulaCell = new ReportCell();
  Formula formula = new Formula("totalSales", "sum({Total Sales})");
  report.addFormula(formula);
  formulaCell.setFormulaObj(formula);
  formulaCell.setBgColor(new Color(255, 255, 255));
  formulaCell.setFontColor(new Color(0, 0, 0));
  formulaCell.setFont(new Font("Dialog", Font.BOLD, 8));
  formulaCell.setAlign(IAlignConstants.ALIGN_LEFT);
  formulaCell.setWidth(1.0);
  formulaCell.setHeight(0.25);
  formulaCell.setX(4.1);
  formulaCell.setY(0);
  report.getReportFooter().addData(formulaCell);
  // End Code :Add a formula

  // Begin Code : Add a label
  ReportCell label = new ReportCell();
  label.setText("Total sales for top 10 customers:");
  label.setBgColor(new Color(255, 255, 255));
  label.setFontColor(new Color(0, 54, 100));
  label.setFont(new Font("Dialog", Font.BOLD, 8));
  label.setAlign(IAlignConstants.ALIGN_RIGHT);
  label.setWidth(2.1);
  label.setHeight(0.4);
  label.setX(1.9);
  label.setY(0);
  report.getReportFooter().addData(label);
  // End Code : Add a label
  remove(reportComponent);
  viewer = new Viewer();
  reportComponent = viewer.getComponent(report);
  add("Center", reportComponent);
  pack();
  return true;
 }

 public Dimension getPreferredSize() {
  return new Dimension(600, 600);
 }

 // Start
 public static void main(String[] args) {
  QuickStart45 t = new QuickStart45();
  t.setSize(600, 600);
  t.setVisible(true);
 }
}
```

The class file for the above source is located in `<EspressReport installation directory>/help/quickstart/classes` directory.

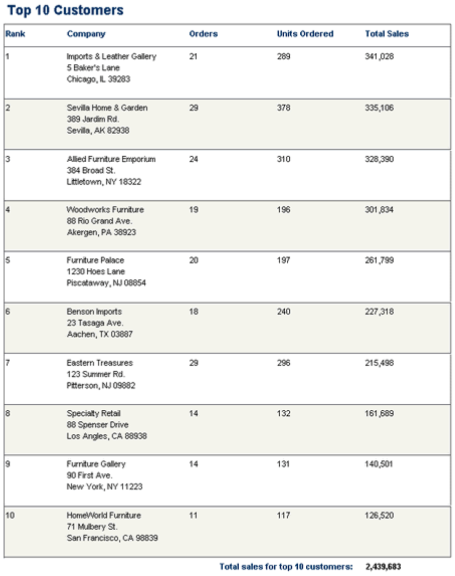
When the class file is run, the following report will be displayed using the command **`java QuickStart45`**:



Rank	Company	Orders	Units Ordered	Total Sales
1	Imports & Leather Gallery 5 Baker's Lane Chicago, IL 39283	21	289	341,028
2	Sevilla Home & Garden 389 Jardim Rd. Sevilla, AK 82938	29	378	335,106
3	Allied Furniture Emporium 384 Broad St. Littletown, NY 18322	24	310	328,390
4	Woodworks Furniture 88 Rio Grand Ave. Akergen, PA 38923	19	196	301,834
5	Furniture Palace 1230 Hoes Lane Piscataway, NJ 08854	20	197	261,799
6	Benson Imports 23 Tasaga Ave. Aachen, TX 03887	18	240	227,318
7	Eastern Treasures 123 Summer Rd. Pitterson, NJ 09882	29	296	215,498
8	Specialty Retail 88 Spenser Drive Los Angles, CA 88938	14	132	161,689
9	Furniture Gallery 90 First Ave. New York, NY 11223	14	131	140,501
10	HomeWorld Furniture 71 Mulbery St. San Francisco, CA 98839	11	117	126,520

Report Generated

When the *Change* button is clicked, the following report will be displayed:



Top 10 Customers

Rank	Company	Orders	Units Ordered	Total Sales
1	Imports & Leather Gallery 5 Baker's Lane Chicago, IL 39283	21	289	341,028
2	Sevilla Home & Garden 389 Jardim Rd. Sevilla, AK 82938	29	378	335,106
3	Allied Furniture Emporium 384 Broad St. Littletown, NY 18322	24	310	328,390
4	Woodworks Furniture 88 Rio Grand Ave. Akergen, PA 38923	19	196	301,834
5	Furniture Palace 1230 Hoes Lane Piscataway, NJ 08854	20	197	261,799
6	Benson Imports 23 Tasaga Ave. Aachen, TX 03887	18	240	227,318
7	Eastern Treasures 123 Summer Rd. Pitterson, NJ 09882	29	296	215,498
8	Specialty Retail 88 Spenser Drive Los Angles, CA 88938	14	132	161,689
9	Furniture Gallery 90 First Ave. New York, NY 11223	14	131	140,501
10	HomeWorld Furniture 71 Mulbery St. San Francisco, CA 98839	11	117	126,520

Total sales for top 10 customers: 2,439,683

Report Generated After Change Button is Clicked

The main part of the code is in the action class. Report properties such as Dual Color are turned on and formula, label and title are added to the `QbReport` object.

Dual colors are set using the following code. The dual color property is set for each table column and alternate background color, font color, and font are specified.

```
<Desired Report Column>.setAlternateRow(int
 numberOfRowsBeforeAlternateColor);
<Desired Report Column>.setBgColor2(Color alternateBackgroundColor);
<Desired Report Column>.setFontColor2(Color alternateFontCOlor);
<Desired Report Column>.setFont2(Font alternateFont);
```

Adding a title to the Report Header is done using the following code. The `ReportCell` object is first created, the title text then set and the `ReportCell` properties such as height, width, x-position, and y-position are specified before adding them to the Report Header section.

```
ReportCell title = new ReportCell();
title.setText(String text);
title.setBgColor(Color backgroundColor);
title.setFontColor(Color fontColor);
title.setFont(Font font);
title.setAlign(int alignment);
title.setWidth(double width);
title.setHeight(double height);
title.setX(double xPosition);
title.setY(double yPosition);
<Handle to desired Report Section>.addData(title);
```

A formula and a label are specified in the same way. A `ReportCell` object is first created, the formula or label set, and the ReportCell properties specified before adding the newly created formula or label ReportCell object to the appropriate section.

```
ReportCell formulaCell = new ReportCell();
Formula formula = new Formula(String formulaname, String formulaText);
report.addFormula(formula);
formulaCell.setFormulaObj(formula);
formulaCell.setBgColor(Color backgroundColor);
formulaCell.setFontColor(Color fontColor);
formulaCell.setFont(Font font);
formulaCell.setAlign(int alignment);
formulaCell.setWidth(double width);
formulaCell.setHeight(double height);
formulaCell.setX(double xPosition);
formulaCell.setY(double yPosition);
<Handle to desired Report Section>.addData(formulaCell);

ReportCell label = new ReportCell();
label.setText(String text);
label.setBgColor(Color backgroundColor);
label.setFontColor(Color fontColor);
label.setFont(Font font);
label.setAlign(int alignment);
label.setWidth(double width);
label.setHeight(double height);
label.setX(double xPosition);
label.setY(double yPosition);
<Handle to desired Report Section>.addData(label);
```

Q.4.6. Parameterized Reports

The following sections show how to run a pre-existing template (`QuickStart46.rpt` located in `<Espress-Report installation directory>/help/quickstart/templates` directory) which uses a parameterized query as the data source.

Each section shows the code to generate the report and any steps necessary to deploy.

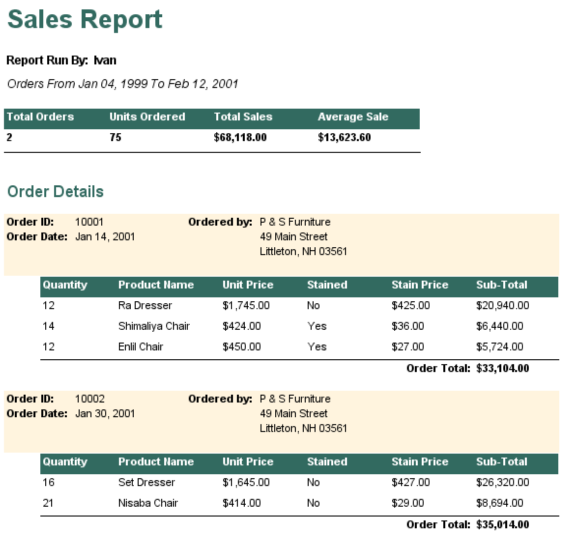
Q.4.6.1. Pass in Parameter values

The following code shows how to display an existing report template (in this case `QuickStart46.rpt`), which uses a parameterized query, in an application. The parameter values are passed when creating the report.

```
import java.io.*;
import java.applet.*;
import java.awt.*;
import java.sql.*;
import java.util.Vector;
import quadbase.reportdesigner.ReportAPI.*;
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.ReportViewer.swing.*;
import quadbase.reportdesigner.util.*;
import quadbase.reportdesigner.lang.*;

public class QuickStart461 extends Applet {

 public QuickStart461() {
 };

 public static void main(java.lang.String[] args) {
  try {
   QuickStart461 doReport = new QuickStart461();
   Frame frame = new Frame();
   frame.setLayout(new BorderLayout());
   frame.add("Center", doReport.createReport(doReport));
   frame.setSize(600, 600);
   frame.setVisible(true);
  } catch (Exception ex) {
   ex.printStackTrace();
  }
 }

 Component createReport(Object parent) {
  // Connect to EspressManager
  QbReport.setEspressManagerUsed(true);
  // Begin Code : Set query parameters
  Vector vec = new Vector();
  vec.addElement("CA");
  vec.addElement("NH");
  Object queryParams[] = new Object[3];
  queryParams[0] = vec;
  queryParams[1] = new Date(99, 0, 4);
  queryParams[2] = new Date(101, 01, 12);
  // End Code : Set query parameters
  // Begin Code : Set formula parameter
  Object formulaParams[] = new Object[1];
  formulaParams[0] = "Ivan";
  // End Code : Set formula parameter
  // Create new Report object using specified report and parameter values
  QbReport report = new QbReport(parent,
    "help/quickstart/templates/QuickStart46.rpt", queryParams,
    formulaParams);

  // Show the Report
  Viewer viewer = new Viewer();
  Component comp = viewer.getComponent(report);
  return comp;

 }

}
```

The class file for the above source is located in `<EspressReport installation directory>/help/quickstart/classes` directory.

When the class file is run, the following report will be displayed using the command **`java QuickStart461`**:



Sales Report

Report Run By: Ivan

Orders From Jan 04, 1999 To Feb 12, 2001

Total Orders	Units Ordered	Total Sales	Average Sale
2	**75**	**$68,118.00**	**$13,623.60**

Order Details

Order ID: 10001 **Ordered by:** P & S Furniture
Order Date: Jan 14, 2001 49 Main Street
Littleton, NH 03561

Quantity	Product Name	Unit Price	Stained	Stain Price	Sub-Total
12	Ra Dresser	$1,745.00	No	$425.00	$20,940.00
14	Shimaliya Chair	$424.00	Yes	$36.00	$6,440.00
12	Enlil Chair	$450.00	Yes	$27.00	$5,724.00
				Order Total:	**$33,104.00**

Order ID: 10002 **Ordered by:** P & S Furniture
Order Date: Jan 30, 2001 49 Main Street
Littleton, NH 03561

Quantity	Product Name	Unit Price	Stained	Stain Price	Sub-Total
16	Set Dresser	$1,645.00	No	$427.00	$26,320.00
21	Nisaba Chair	$414.00	No	$29.00	$8,694.00
				Order Total:	**$35,014.00**

Report Generated

The main part of the code is in the `createReport` component. There, a `QbReport` object called `report` is created using the `QuickStart46.rpt` template. The parameter values are passed using the following constructor:

```
QbReport(Object parent, String reportTemplatename, Object[]
 queryParameterValues, Qbject[] formulaParamterValues);
```

Both query parameter and formula parameter values are placed in the same order as the parameters were defined. Query parameters, which take in multiple values, are declared as `Vector` object which then contains different multiple values.

Please note that if you are using Access templates, you will need to change the following lines of code from:

```
queryParams[1] = new Date(99, 0, 4);
queryParams[2] = new Date(101, 01, 12);
```

to

```
queryParams[1] = new Timestamp(99, 0, 4, 0, 0, 0, 0);
queryParams[2] = new Timestamp(101, 01, 12, 0, 0, 0, 0);
```

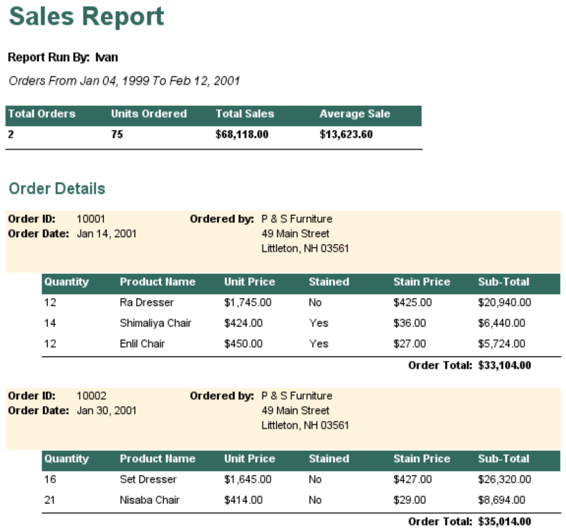
Q.4.6.2. Pass in Parameter values using getAllParameters

In addition to the above, you can also pass in parameter values using `getAllParameters` method in `QbReport`. The following code shows how to display an existing report template (in this case `QuickStart46.rpt`), which uses a parameterized query, in an application. The report is opened with backup data (to avoid unnecessary load to the database) and the parameter values are set. The report is then refreshed with the data.

```
import java.io.*;
import java.applet.*;
import java.awt.*;
import java.sql.*;
import java.util.Vector;
import quadbase.reportdesigner.ReportAPI.*;
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.ReportViewer.swing.*;
import quadbase.reportdesigner.util.*;
import quadbase.reportdesigner.lang.*;

public class QuickStart462 extends Applet {

 public QuickStart462() {
 };

 public static void main(java.lang.String[] args) {
  try {
   QuickStart462 doReport = new QuickStart462();
   Frame frame = new Frame();
   frame.setLayout(new BorderLayout());
   frame.add("Center", doReport.createReport(doReport));
   frame.setSize(600, 600);
   frame.setVisible(true);
  } catch (Exception ex) {
   ex.printStackTrace();
  }
 }

 Component createReport(Object parent) {
  // Connect to EspressManager
  QbReport.setEspressManagerUsed(true);
  // Begin Code : Set query parameters
  Vector vec = new Vector();
  vec.addElement("CA");
  vec.addElement("NH");
  Object queryParams[] = new Object[3];
  queryParams[0] = vec;
  queryParams[1] = new Date(99, 0, 4);
  queryParams[2] = new Date(101, 01, 12);
  // End Code : Set query parameters
  // Begin Code : Set formula parameter
  Object formulaParams[] = new Object[1];
  formulaParams[0] = "Ivan";
  // End Code : Set formula parameter
  // Create new Report object using backup data
  QbReport report = new QbReport(parent,
    "help/quickstart/templates/QuickStart46.rpt", false, false,
    false, true);

  try {

   report.getAllParameters().get(0).setValues((Vector) queryParams[0]);
   report.getAllParameters().get(1).setValue(queryParams[1]);
   report.getAllParameters().get(2).setValue(queryParams[2]);
   report.getAllParameters().get(3).setValue(formulaParams[0]);

   report.refreshWithOriginalData();

  } catch (Exception ex) {
   ex.printStackTrace();
  }

  // Show the Report
  Viewer viewer = new Viewer();
  Component comp = viewer.getComponent(report);
  return comp;
 }
}
```

The class file for the above source is located in `<EspressReport installation directory>/help/quickstart/classes` directory.

When the class file is run, the following report will be displayed using the command **`java QuickStart462`**:



Sales Report

Report Run By: Ivan

Orders From Jan 04, 1999 To Feb 12, 2001

Total Orders	Units Ordered	Total Sales	Average Sale
2	**75**	**$68,118.00**	**$13,623.60**

Order Details

Order ID: 10001 **Ordered by:** P & S Furniture
Order Date: Jan 14, 2001 49 Main Street
Littleton, NH 03561

Quantity	Product Name	Unit Price	Stained	Stain Price	Sub-Total
12	Ra Dresser	$1,745.00	No	$425.00	$20,940.00
14	Shimaliya Chair	$424.00	Yes	$36.00	$6,440.00
12	Enlil Chair	$450.00	Yes	$27.00	$5,724.00
				Order Total:	**$33,104.00**

Order ID: 10002 **Ordered by:** P & S Furniture
Order Date: Jan 30, 2001 49 Main Street
Littleton, NH 03561

Quantity	Product Name	Unit Price	Stained	Stain Price	Sub-Total
16	Set Dresser	$1,645.00	No	$427.00	$26,320.00
21	Nisaba Chair	$414.00	No	$29.00	$8,694.00
				Order Total:	**$35,014.00**

Report Generated

The main part of the code is in the `createReport` component. There, a `QbReport` object called `report` is created using the `QuickStart46.rpt` template. The parameter values are passed using the following interface in `QbReport`:

```
getAllParameters();
```

Both query parameter and formula parameter values are placed in the same order as the parameters were defined. Query parameters which take in multiple values are declared as `Vector` object which then contains different multiple values.

Please note that if you are using Access templates, you will need to change the following lines of code from:

```
queryParams[1] = new Date(99, 0, 4);
queryParams[2] = new Date(101, 01, 12);
```

to

```
queryParams[1] = new Timestamp(99, 0, 4, 0, 0, 0, 0);
queryParams[2] = new Timestamp(101, 01, 12, 0, 0, 0, 0);
```

Q.4.6.3. Use getParameterPage() and ParamReportGeneratorServlet

The following code shows how to display an existing report template (in this case `QuickStart46.rpt`), which uses a parameterized query, in a servlet. The servlet creates the `QbReport` object by opening the template using backup data. A HTML page is then streamed asking for the parameter values. These values are passed to another servlet (the `ParamReportGeneratorServlet` servlet) and the `QbReport` object is created from the given template with the specified parameter values.

```
import quadbase.reportdesigner.ReportAPI.QbReport;
import quadbase.common.param.*;
import java.applet.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.awt.Font;

public class QuickStart463 extends HttpServlet {

 public void doGet(HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException {

  System.out.println("QuickStart463 Servlet: Do Get....");
  System.out
    .println("QuickStart463 Servlet: Generate paramter html page...");
  res.setContentType("text/html");

  OutputStream toClient = res.getOutputStream();
  try {
   // Connect to EspressManager
   QbReport.setEspressManagerUsed(true);
   String reportLocation = "help/quickstart/templates/QuickStart46.rpt";
   // Create the ObReport object using back-up data
   QbReport report = new QbReport((Applet) null, reportLocation,
     false, false, false, true, false);

   // Specify the parameters for connecting to
   // ParamReportGeneratorServlet
   report.setDynamicExport(true, "127.0.0.1", 8080);

   // Specify report template location and export format desired
   ParameterPage paramPage = report.getParameterPage(reportLocation,
     null, QbReport.DHTML, null);
   Writer writer = new PrintWriter(toClient);
   HtmlParameterPageWriter paramPageWriter = new HtmlParameterPageWriter(
     paramPage, writer);
   paramPageWriter.writePage();
   writer.flush();
   writer.close();
  } catch (Exception e) {
   e.printStackTrace();
  }
  toClient.flush();
  toClient.close();

 }

 public String getServletInfo() {
  return "QuickStart463 servlet for EspressReport";
 }
}
```

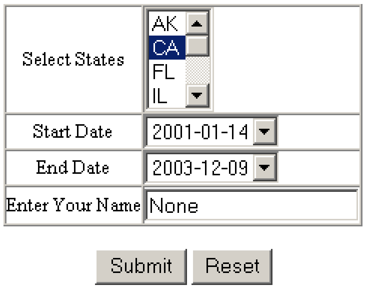
To deploy the servlet:

- Move `QuickStart463.class` from `<EspressReport installation directory>/help/quickstart/classes` directory to `<Tomcat installation>/webapps/ROOT/WEB-INF/classes` directory;

- Compile `ParamReportGeneratorServlet <EspressReport installation directory>/ParamReportGenerator`

- Move `ParamReportGeneratorServlet.class` to `<Tomcat installation>/webapps/ROOT/WEB-INF/classes` directory.

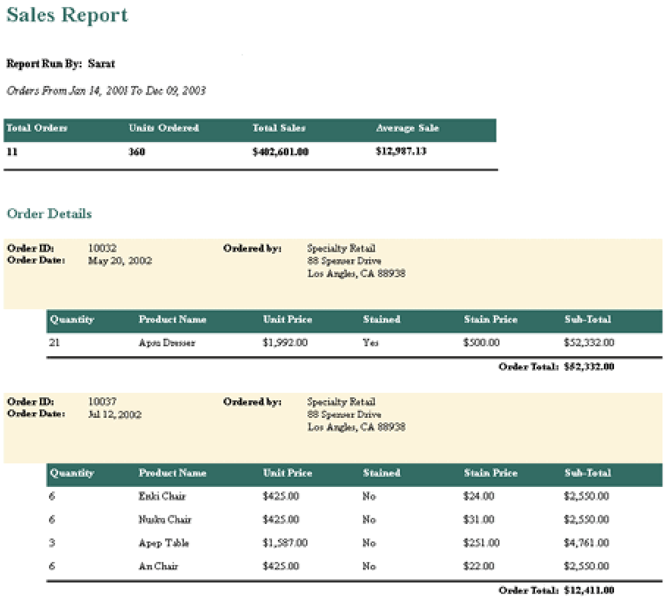
When the servlet is run using the URL **`http://localhost:8080/servlet/QuickStart463`**, the following HTML page will be displayed:



Select States	AK CA FL IL
Start Date	2001-01-14
End Date	2003-12-09
Enter Your Name	None

HTML Prompt Page Generated

Depending on what parameter values you pass in, a report similar to the following will be displayed:



Sales Report

Report Run By: Sarat

Orders From Jan 14, 2001 To Dec 09, 2003

Total Orders	Units Ordered	Total Sales	Average Sale
11	**360**	**$402,601.00**	**$12,987.13**

Order Details

Order ID: 10032 **Ordered by:** Specialty Retail
Order Date: May 20, 2002 88 Spencer Drive
Los Angles, CA 88938

Quantity	Product Name	Unit Price	Stained	Stain Price	Sub-Total
21	Apsu Dresser	$1,992.00	Yes	$500.00	$52,332.00

Order Total: $52,332.00

Order ID: 10037 **Ordered by:** Specialty Retail
Order Date: Jul 12, 2002 88 Spencer Drive
Los Angles, CA 88938

Quantity	Product Name	Unit Price	Stained	Stain Price	Sub-Total
6	Enki Chair	$425.00	No	$24.00	$2,550.00
6	Nusku Chair	$425.00	No	$31.00	$2,550.00
3	Apep Table	$1,587.00	No	$251.00	$4,761.00
6	An Chair	$425.00	No	$22.00	$2,550.00

Order Total: $12,411.00

Report Generated After Passing in Parameter Values

The main part of the code is in the `QuickStart463`. There, a `QbReport` object called `report` is created using the `QuickStart46.rpt` template. The dynamic export is then called to use the `ParamReportGeneratorServlet` (provided with EspressReport) using the following lines:

```
<QbReport object>.setDynamicExport(boolean isDynamicExport, String
 servername, int servletRunnerPort);
String htmlParamPage = <QbReport object>.getHTMLParamPage(String
 reportLocation, int exportFormat);
```

The HTML file asking for the parameter values is then generated and passed to the `OutputStream`.

For more information about the parameter page writer classes, see Section 2.3.5.7.10.3 - Generating HTML Parameter Page. For an example of generating parameter page using CssHtmlParameterPageWriter which utilize CSS, go to <EspressReport>/help/examples/servlet/CssParamReport and follow the javadoc instructions.

Q.4.7. Deploy/Export Drill-Down

The following code shows how to display an existing report template (in this case `QuickStart47.rpt`), which is a drill-down report, in a servlet. The servlet creates the `QbReport` object by opening the template. A DHTML page, showing the contents of the first level, is then streamed. The next level report is obtained by clicking on the links. These links point to the `DrillDownReportServlet`. The values of the link (clicked on) are passed to the `DrillDownReportServlet` and the next level report (based on those values) is then generated and then streamed to the client.

```
import quadbase.reportdesigner.ReportAPI.QbReport;
import java.applet.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.awt.Font;

public class QuickStart47 extends HttpServlet {

 public void doGet(HttpServletRequest req, HttpServletResponse res)
```

```
   throws ServletException, IOException {

  System.out.println("QuickStart47 Servlet: Do Get....");
  res.setContentType("text/html");

  OutputStream toClient = res.getOutputStream();
  try {
   // Connect to EspressManager
   QbReport.setEspressManagerUsed(true);
   String reportLocation = "help/quickstart/templates/QuickStart47.rpt";
   // Create the ObReport object using back-up data
   QbReport report = new QbReport((Applet) null, reportLocation);
   // Specify the parameters for connecting to DrillDownReportServlet
   report.setDynamicExport(true, "127.0.0.1", 8080);
   // Export the report to DHTML
   report.export(QbReport.DHTML, toClient);
  } catch (Exception e) {
   e.printStackTrace();
  }

  toClient.flush();
  toClient.close();

 }

 public String getServletInfo() {
  return "QuickStart47 servlet for EspressReport";
 }
}
```

To deploy the servlet:

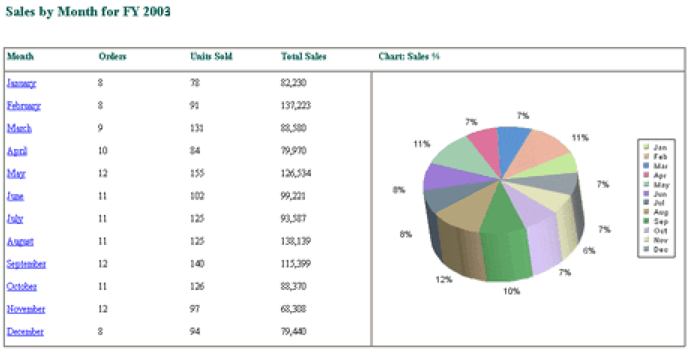
- Move `QuickStart47.class` from `<EspressReport installation directory>/help/quickstart/classes` directory to `<Tomcat installation>/webapps/ROOT/WEB-INF/classes` directory;
- Compile `DrillDownReportServlet` `<EspressReport installation directory>/DrillDownLinkGenerator`;
- Move `DrillDownReportServlet.class` to the `<Tomcat installation>/webapps/ROOT/WEB-INF/classes` directory;
- Compile `RPTImageGenerator` `<EspressReport installation directory>/ImageGenerator`;
- Move `RPTImageGenerator.class` to `<Tomcat installation>/webapps/ROOT/WEB-INF/classes` directory.
- For Tomcat 7.x or higher version, the servlet, i.e. `QuickStart482` in this example needs to be explicitly mapped in web.xml:

```
<servlet>
  <servlet-name>QuickStart47</servlet-name>
  <servlet-class>QuickStart47</servlet-class>
</servlet>
```

and

```
<servlet-mapping>
  <servlet-name>QuickStart47</servlet-name>
  <url-pattern>/servlet/QuickStart47</url-pattern>
</servlet-mapping>
```

When the servlet is run using the URL **`http://localhost:8080/servlet/QuickStart47`**, the following HTML page will be displayed:

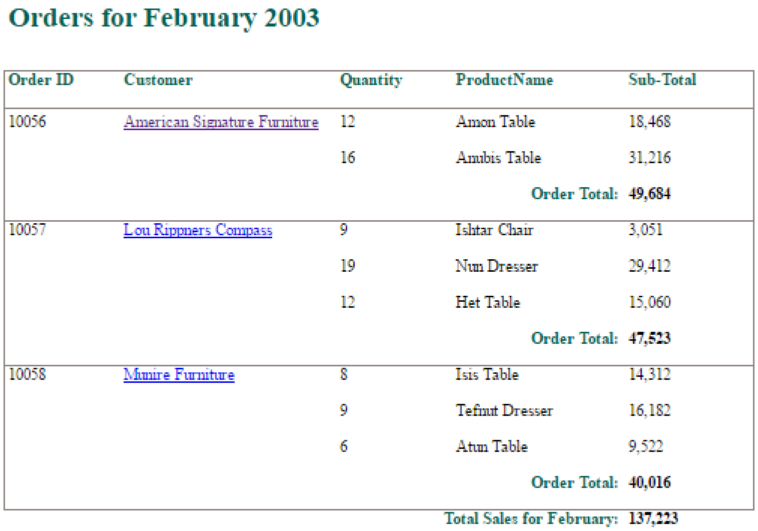


Sales by Month for FY 2003

Month	Orders	Units Sold	Total Sales	Chart: Sales %
January	8	78	82,230	
February	8	91	137,223	
March	9	131	88,580	
April	10	84	79,970	
May	12	155	126,534	
June	11	102	99,221	
July	11	125	93,587	
August	11	125	138,139	
September	12	140	115,399	
October	11	126	88,370	
November	12	97	68,308	
December	8	94	79,440	

Report Generated

Depending on what link you click, a report similar to the following will be displayed:

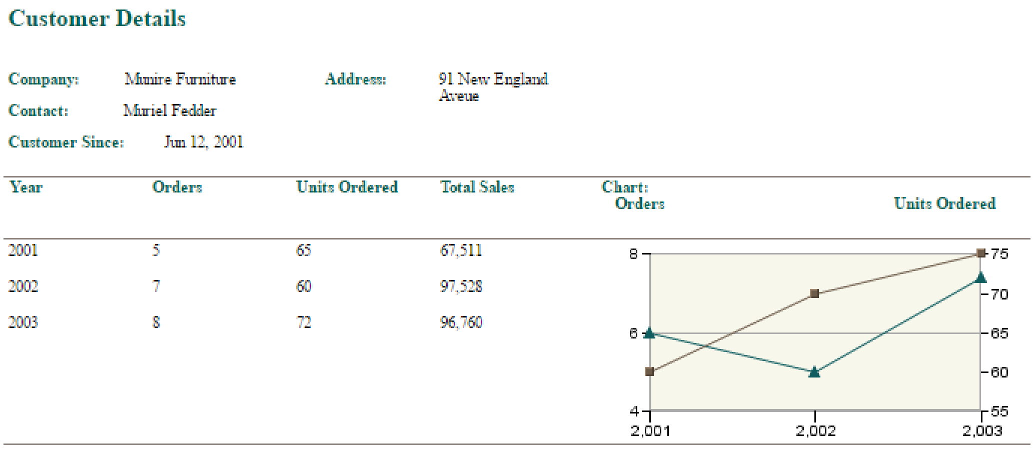


Orders for February 2003

Order ID	Customer	Quantity	ProductName	Sub-Total
10056	American Signature Furniture	12	Amon Table	18,468
		16	Anubis Table	31,216
			Order Total:	**49,684**
10057	Lou Rippners Compass	9	Ishtar Chair	3,051
		19	Nun Dresser	29,412
		12	Het Table	15,060
			Order Total:	**47,523**
10058	Munire Furniture	8	Isis Table	14,312
		9	Tefnut Dresser	16,182
		6	Atun Table	9,522
			Order Total:	**40,016**
			Total Sales for February:	**137,223**

Report Generated After Clicking on a Link

Again, depending on what link you click, a report similar to the following will be displayed:



Customer Details

Company: Munire Furniture

Address: 91 New England Aveue

Contact: Muriel Fedder

Customer Since: Jun 12, 2001

Year	Orders	Units Ordered	Total Sales	Chart: Orders Units Ordered
2001	5	65	67,511	
2002	7	60	97,528	
2003	8	72	96,760	

Report Generated After Clicking on a Link

The main part of the code is in the `QuickStart47`. There, a `QbReport` object called report is created using the `QuickStart47.rpt` template. The dynamic export is then called to use the `DrillDownReportServlet` (provided with EspressReport) using the following lines:

```
<QbReport object>.setDynamicExport(boolean isDynamicExport, String
 servername, int servletRunnerPort);
```

The DHTML file for the top level report is then generated and passed to the `OutputStream`.

```
<QbReport object>.export(int exportFormat, OutputStream out);
```

Q.4.8. More on Servlets

The following sections show how to run a pre-existing template (`QuickStart48.rpt` located in `<EspressReport installation directory>/help/quickstart/templates` directory) in a servlet and have the generated report saved to a file or streamed to the client browser.

Each section shows the code to generate the report and any steps necessary to deploy.

Q.4.8.1. Write to File (PDF)

The following code shows how to display an existing report template (in this case `QuickStart48.rpt`) in an servlet. The servlet then exports the report to a PDF file.

```
import quadbase.reportdesigner.ReportAPI.*;
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.ReportViewer.*;
import quadbase.reportdesigner.util.*;
import quadbase.reportdesigner.lang.*;
import java.awt.*;
import java.applet.*;
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
```

```
public class QuickStart481 extends HttpServlet {

 public void doGet(HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException {
  System.out.println("Calling QuickStart481....");
  // Set the "content type" header of the response
  res.setContentType("text/html");
  // Get the response's PrintWriter to return content to the client.
  PrintWriter toClient = res.getWriter();
  try {
   // Use EspressManager
   QbReport.setEspressManagerUsed(true);
   // Open up specified Report
   QbReport report = new QbReport((Applet) null,
     "help/quickstart/templates/QuickStart48.rpt");
   // Export the report to PDF file
   report.export(QbReport.PDF, "generatedReport");
   // Begin Code : Send message to client saying file has been exported
   toClient.println("<html>");
   toClient.println("<head>");
   toClient.println("</head>");
   toClient.println("<body>");
   toClient.println("The report has been exported to generatedReport.pdf in
 your EspressReport installation root directory.");
   toClient.println("</body>");
   toClient.println("</html>");
   // End Code : Send message to client saying file has been exported
  } catch (Exception e) {
   e.printStackTrace();
  }
  // Flush the outputStream
  toClient.flush();
  // Close the writer; the response is done.
  toClient.close();
 }

 public String getServletInfo() {
  return "QuickStart481 servlet for EspressReport";
 }
}
```

To deploy the servlet:

- Move `QuickStart481.class` from `<EspressReport installation directory>/help/quickstart/classes` directory to `<Tomcat installation>/webapps/ROOT/WEB-INF/classes` directory.

When the servlet is run using the URL **`http://localhost:8080/servlet/QuickStart481`**, the report is generated to your `<EspressReport installation directory>/generatedReport.pdf` and the following message will be displayed:

The report has been exported to generatedReport.pdf in your EspressReport installation root directory.

Message Shown After Report is Generated

The main part of the code is in the `QuickStart481`. There, a `QbReport` object called `report` is created using the `QuickStart48.rpt` template. The following constructor is used. The `QbReport` object report was then exported as PDF using the `export` method:

```
<QbReport object>.export(int exportFormat, String exportFilename);
```

Q.4.8.2. Stream Chart (DHTML)

The following code shows how to display an existing report template (in this case `QuickStart48.rpt`) in a servlet. The servlet then streams the report (along with the chart) as a DTHML document to the client.

```
import quadbase.reportdesigner.ReportAPI.QbReport;
import java.applet.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.awt.Font;

public class QuickStart482 extends HttpServlet {

 public void doGet(HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException {
  System.out.println("QuickStart482 Servlet: Do Get....");
  res.setContentType("text/html");
  OutputStream toClient = res.getOutputStream();
  try {
   // Connect to EspressManager
   QbReport.setEspressManagerUsed(true);
   String reportLocation = "help/quickstart/templates/QuickStart48.rpt";
   // Create the ObReport object
   QbReport report = new QbReport((Applet) null, reportLocation);

   // Specify the parameters for connecting to RPTImageGenerator
   report.setDynamicExport(true, "127.0.0.1", 8080);

   // Stream report to client
   report.export(QbReport.DHTML, toClient);
  } catch (Exception e) {
   e.printStackTrace();
  }
  toClient.flush();
  toClient.close();

 }

 public String getServletInfo() {
  return "QuickStart482 Servlet for EspressReport";
 }
}
```

To deploy the servlet:

- Move `QuickStart482.class` to `<Tomcat installation>/webapps/ROOT/WEB-INF/classes` directory;
- Compile `RPTImageGenerator` `<EspressReport installation directory>/ImageGenerator`;

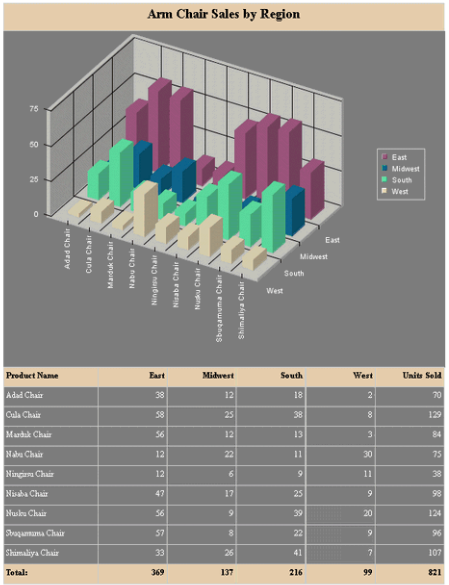
- Move `RPTImageGenerator.class` to `<Tomcat installation>/webapps/ROOT/WEB-INF/classes` directory.
- For Tomcat 7.x or higher version, the servlet, i.e. `QuickStart482` in this example, needs to be explicitly mapped in web.xml:

```
<servlet>
  <servlet-name>QuickStart482</servlet-name>
  <servlet-class>QuickStart482</servlet-class>
</servlet>
```

and

```
<servlet-mapping>
  <servlet-name>QuickStart482</servlet-name>
  <url-pattern>/servlet/QuickStart482</url-pattern>
</servlet-mapping>
```

When the servlet is run using the URL **`http://localhost:8080/servlet/QuickStart482`**, the following HTML page will be displayed:



Arm Chair Sales by Region

Product Name	East	Midwest	South	West	Units Sold
Adad Chair	38	12	18	2	70
Cula Chair	58	25	38	8	129
Marduk Chair	56	12	13	3	84
Nabu Chair	12	22	11	30	75
Ningirsu Chair	12	6	9	11	38
Nisaba Chair	47	17	25	9	98
Nusku Chair	56	9	39	20	124
Sbuqamuma Chair	57	8	22	9	96
Shimaliya Chair	33	26	41	7	107
Total:	**369**	**137**	**216**	**99**	**821**

Report generated

The main part of the code is in the `QuickStart482`. There, a `QbReport` object called `report` is created using the `QuickStart48.rpt` template. Dynamic export is then called to use the `RPTImageGenerator` (provided with EspressReport) using the following lines:

```
<QbReport object>.setDynamicExport(boolean isDynamicExport, String
 servername, int servletRunnerPort);
```

The DHTML file for the top level report is then generated and passed to the `OutputStream`.

```
<QbReport object>.export(int exportFormat, OutputStream out);
```

Q.4.9. Launch Report Designer

The following code shows how to launch the Report Designer (in default mode) via the Report API:

```
import java.awt.*;
import javax.swing.*;
import java.io.*;
import quadbase.reportdesigner.designer.*;

public class QuickStart49 {

 public static void main(java.lang.String[] args) {

  try {
   QuickStart49 doReport = new QuickStart49();
  } catch (Exception ex) {
   ex.printStackTrace();
  }
 }

 public QuickStart49() {

  // Begin Code : Start Designer in default mode and show the Designer
  QbReportDesigner.setUseSysResourceImages(true);
  QbReportDesigner designer = new QbReportDesigner(null);
  designer.setVisible(true);
  // End Code : Start Designer in default mode and show the Designer
 }
}
```

The class file for the above source is located in `<EspressReport installation directory>/help/quickstart/classes` directory.

To run the class file, move `QuickStart49.class` to `<EspressReport Installation directory>` and use the command **`java QuickStart49`**.

The main part of the code is in the `QuickStart49`. There, a `QbReportDesigner` object called `designer` is created and shown:

```
QbReportDesigner(Object parent);
<QbReportDesigner object>.setVisible(boolean b);
```

Chapter 1. Designer Guide

1.1. Overview / Introduction

Welcome to EspressReport. Written in 100% pure Java, EspressReport is a set of tools that allows you to easily create professional, information-rich reports from many data sources and include them in your applications, web based or otherwise. Among other components, EspressReport comes with Report Designer and a robust object oriented API. The Report Designer provides a powerful, yet easy to use graphical user interface where you can quickly edit and format content, turning raw data into polished reports in a matter of minutes. The finished report design can be saved in a template for subsequent distribution with the latest data. To publish a report on the web, you can write JSP/servlet code using the API to refer to the template and generate the report in a desired format. Supported export formats include PDF, DHTML, text, Excel (XLS) and Excel 2007 (XLSX) spreadsheets. Reports can also be viewed in a browser with Report Viewer or EspressReport's API running as applet, and later printed in a WYSIWYG hard copy.

Included in the report designer is a Query Builder and Data Source Manager that allows you to quickly extract data from JDBC, XML, and ASCII sources. The sophisticated Data View features allow administrators to create local data catalogs for end users and present data from databases in simplified view. Using this feature, end users with no knowledge of databases can easily perform ad-hoc reporting.

To facilitate easy report design and creation, EspressReport supports five report types, namely: simple columnar, summary break, crosstab, master & details and mailing labels. Advanced features allow users to include sub-reports, drill-downs, parameters, insert hyperlinks and images, manipulate data with over eighty-five built-in formulas and incorporate sophisticated charts and graphs into reports.

1.1.1. EspressReport Documentation

There are four sections of the EspressReport Documentation.

Quick Start Guide: This is a good starting point for EspressReport first time users. It covers most commonly used features and provides Report Designer and API tutorials.

Designer Guide: This explains all of the functionality of the Report Designer and teaches you how to develop queries in the Query Builder and format and manipulate data to create polished reports.

Programming Guide: This covers the Report API and teaches you how to create reports programmatically, as well as incorporate them into servlets, JSPs, and applications.

1.2. Architecture & Installation

There are seven main components of EspressReport, namely: Report Designer, Chart Designer, Report API, Report Viewer, Page Viewer, EspressManager, and Scheduler.

Report Designer: Report Designer is a GUI tool that allows users to build reports in a point-and-click environment. Included with the Report Designer are interfaces to access data, query databases, and design charts. The Report Designer can run as a client application or in a client-server configuration with the Designer loaded as an applet through a web browser.

Chart Designer: Chart Designer is a GUI tool that allows users to build and design charts in a point-and-click environment. Launched from within Report Designer, it allows users to create charts that can be embedded within reports or stand-alone charts that can be run using the Report API.

Report API: Report API is an easy-to-use application programming interface that allows users to imbed reporting functionality into their applications, servlets, or JSPs, either on the server-side or client-side. It can run on most platforms with few or no changes because it is pure Java. Any and every part of the report is customizable using the API, giving users full control over report formatting at run-time. Reports can be created and deployed with just a few lines of code.

Report Viewer: Report Viewer is an integrated applet that allows users to view and interact with reports. The applet shows reports in a paginated format, giving users an opportunity to navigate around reports using a pop-up menu. Also, report templates can be directly hyperlinked together by using the applet. EspressReport can generate HTML pages with the applet imbedded, without requiring any code.

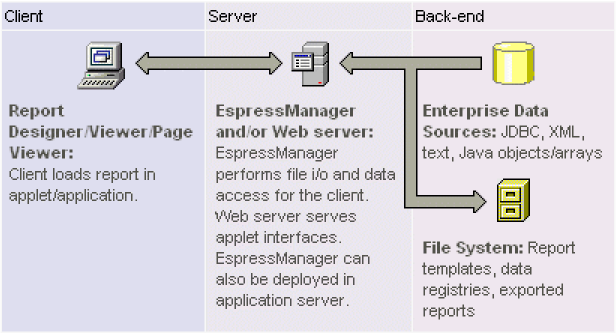
Page Viewer: Page Viewer is an integrated applet like Report Viewer that uses page serving technology. With Page Viewer, pages in the report are only sent to the client when requested. This allows users to view or preview large reports.

EspressManager: EspressManager serves as the “back end” to the Report Designer and Scheduler interfaces. It supports Report Designer running as an applet. EspressManager handles the data access and file I/O activities on the server-side. Additionally, EspressManager provides database connection and data buffering. EspressManager also runs the scheduling process on the server-side, executing reports according to the user-defined jobs. It can run as an application process on the server, or it can be deployed as a servlet within an application server.

Scheduler: The Scheduler is a small interface that runs in conjunction with EspressManager. It allows users to schedule reports and other events, as well as manage scheduled tasks.

1.2.1. EspressReport Architecture

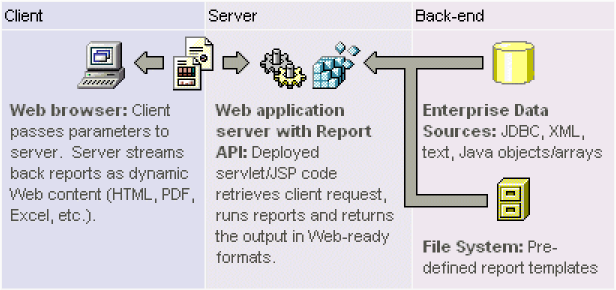
EspressReport has a number of different configurations in which it can run both at design-time and at run-time. At design-time, the Report Designer GUI interface, which includes data access tools and charting tools, can be loaded as an application on a client machine or as a JNLP(More about Applets in JNLP: Section 1.2.7 - Run Applets in WebStart with JNLP file in a client server architecture. The following diagram illustrates EspressReport running with the Report Designer at design-time:


Client
Server
Back-end
Report Designer/Viewer/Page Viewer: Client loads report in applet/application.
EspressManager and/or Web server: EspressManager performs file i/o and data access for the client. Web server serves applet interfaces. EspressManager can also be deployed in application server.
Enterprise Data Sources: JDBC, XML, text, Java objects/arrays
File System: Report templates, data registries, exported reports

EspressReport Designer Architecture

When the Report Designer is running, the EspressManager component runs on the server-side. It can run either as an application process or as a servlet deployed within an application server/servlet runner. The EspressManager performs data access and file I/O which is prevented by the client applet due to security restrictions. The EspressManager also provides connection and data buffering for database connections, as well as concurrency control for a multi-user development environment. The EspressManager must be running in conjunction with the Report Designer.

At run-time, the EspressManager does not need to be running. EspressReport is designed to run embedded within other application environments. It can have a minimal deployment of the API classes and report template files. The following diagram illustrates EspressReport running in a servlet/JSP environment:


Client
Server
Back-end
Web browser: Client passes parameters to server. Server streams back reports as dynamic Web content (HTML, PDF, Excel, etc.).
Web application server with Report API: Deployed servlet/JSP code retrieves client request, runs reports and returns the output in Web-ready formats.
Enterprise Data Sources: JDBC, XML, text, Java objects/arrays
File System: Pre-defined report templates

EspressReport in a Servlet Environment

When running in an application server, the Report API can be used to generate reports within servlets and JSPs, and stream the generated reports to the client browser. Clients can also view reports by loading the Report Viewer applet or the Page Viewer applet.

Report generation can be handled on demand, triggered by application processes, or scheduled. A Scheduler interface is also provided with EspressReport. EspressManager must be running in order to run the Scheduler.

In addition to running in a client-server environment, EspressReport can be run in a thick-client application. The Report API can be used in conjunction with the Viewer and/or Page Viewer to show or generate reports in an application interface. In this configuration, EspressManager does not need to be running, as the whole process can be contained on the client side. Note that this is not the case for applets.

1.2.2. Installation

There are four versions of EspressReport installer available: one for Windows, one for Unix, one for Mac OS X, and a pure Java version.

Windows Version

To start the Windows installer, run the `installER.exe` file and the installer will launch.

Unix Version

To start the Unix installer, run the `installER.bin` file and the installer will launch.

Mac OS X Version

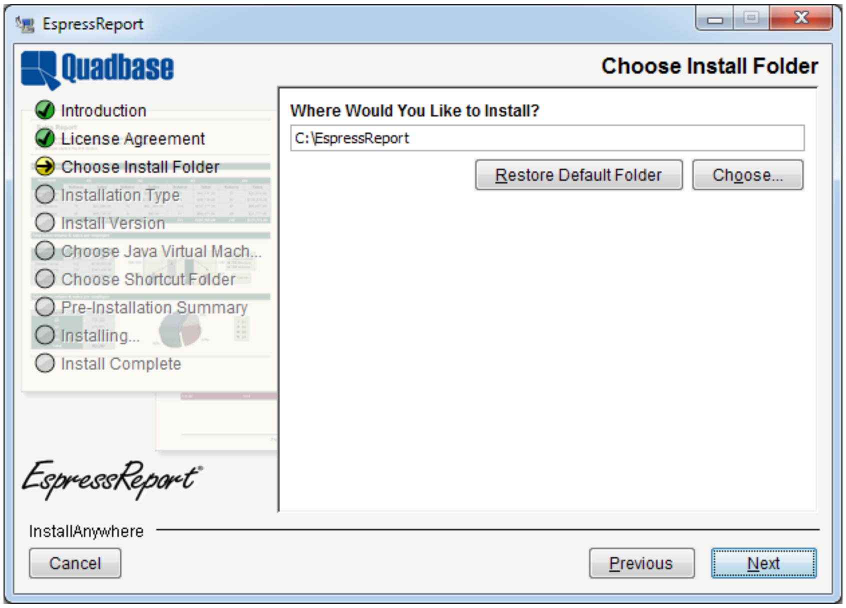
To start the Mac installer, double click the `InstallER.zip` file to extract the `InstallER.app` file. Double-click on `InstallER.app`, and the installer will launch.

Pure Java Version

To start the pure Java installer, a Java Virtual Machine, the equivalent of Java 1.8 or higher, must already be installed on the machine where EspressReport is to be installed. Make sure that the JVM is included in your path (or move the `installER.jar` file to the same directory as the JVM). From a command prompt navigate to the directory where you have placed the `installER.jar` file, and type the following command: **`java -jar installER.jar`**. The installer will then launch.

Once the installation program has launched and you agree to the license agreement, the first option asks you to specify the directory into which you would like to install EspressReport.

The default location is \EspressReport\. You can specify a new directory or browse to one by clicking the *Choose…* button.

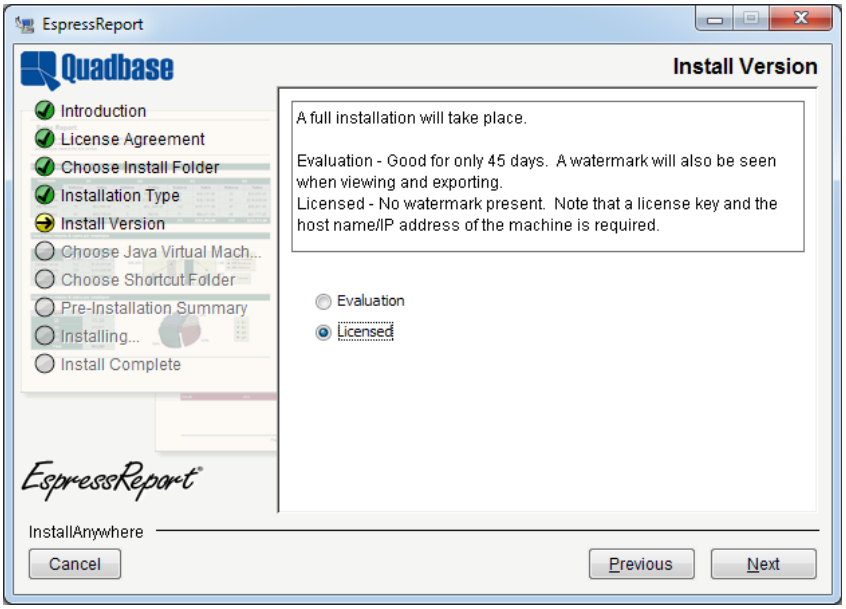


Where Would You Like to Install?

C:\EspressReport

Choose Location Dialog

Once you choose the installation directory, the next option asks you whether you want to install the evaluation or licensed version.



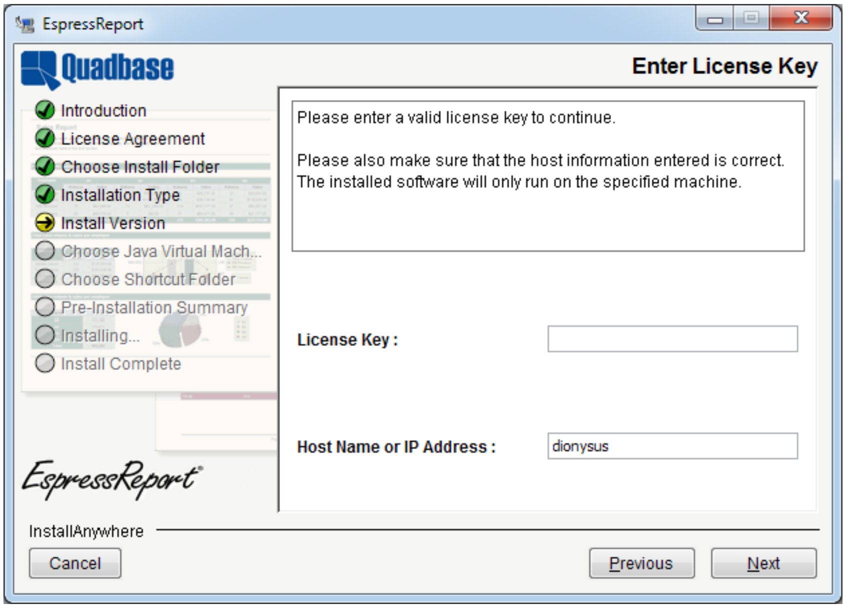
A full installation will take place.

Evaluation - Good for only 45 days. A watermark will also be seen when viewing and exporting.
Licensed - No watermark present. Note that a license key and the host name/IP address of the machine is required.

○ Evaluation
● Licensed

Install Version Dialog

If you select to install a licensed version, the next dialog will prompt you to enter your license key and verify the host name of the machine on which you're installing EspressReport.



Please enter a valid license key to continue.

Please also make sure that the host information entered is correct. The installed software will only run on the specified machine.

License Key :

Host Name or IP Address : dionysus

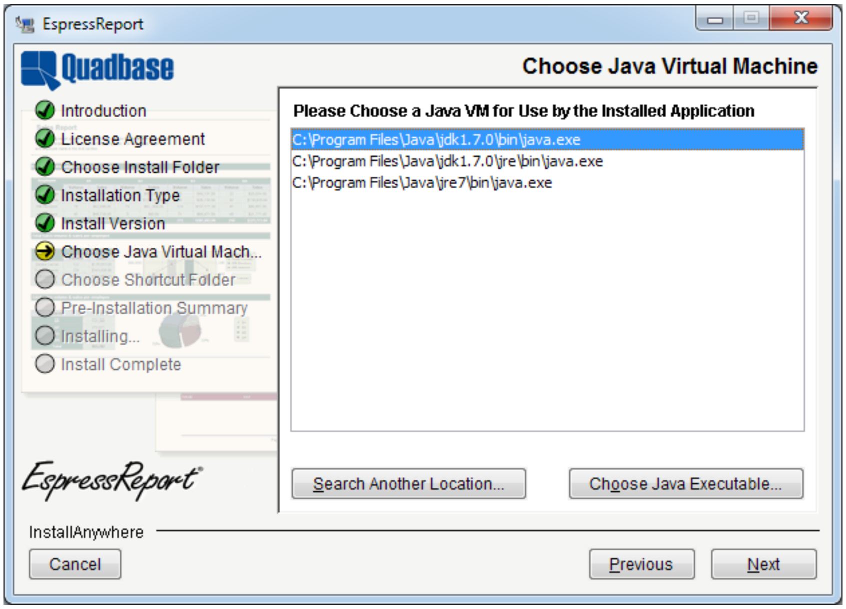
Enter License Key Dialog

After you enter all informations, the installer will attempt to register your license key with Quadbase. If the registration fails, you will be unable to install the release version of EspressReport. However, you will still be able to install the evaluation version. After the installation is complete, you can register your key online at https://www.quadbase.com/register/, or contact Quadbase Sales for additional help.

Note

After the installation is complete, the release version will only run for the hostname specified in this dialog, so double-check it to make sure it's correct. You can also use the IP address of the machine if you prefer.

The next window allows you to specify which Java Virtual Machine version should be used to run EspressReport. (Since it is a Java program, you will need JVM to run it.)



Please Choose a Java VM for Use by the Installed Application

C:\Program Files\Java\jdk1.7.0\bin\java.exe
C:\Program Files\Java\jdk1.7.0\jre\bin\java.exe
C:\Program Files\Java\jre7\bin\java.exe

Choose JVM Dialog

The dialog prompts you to select which Java Virtual Machine you want to use. A list of compatible JVMs on the system will be displayed, allowing you to pick one. If the JVM you want to use is not listed, you can click the *Select Another Location...* button to browse to it. When using the windows installer, you can also choose to install the JVM bundled with the installer, if a compatible JVM has not been found.

Note

The JVM that is bundled with the installer only provides Java Runtime Environment. In order to develop and execute Java code using the EspressReport API, you will need to have Java Development Kit and/or compiler/IDE. Also, if you're using pure Java or Mac OS X version of the installer, this option will not appear. Make sure that you're running the installer with the JVM you wish to run the program.

The last option in the installer is for Windows or Mac OS X version of the installer. It allows you to specify the location of program shortcuts. By default the shortcuts are added to a program group called EspressReport in the Start Menu.

For Mac OS X, you can create aliases on desktop, in dock, or in a folder of your choice.

After you complete the last option, you will be shown a summary of all the options you've selected. Click the *Next* button and the program will install.

1.2.3. Configuration

The configuration for EspressReport is already set and it can be run without changing anything. However, if you want to:

- Change the port number that EspressManager use
- Change the webroot
- Add/delete/modify user

You will need to modify the configuration settings. These settings are stored in a text file called *config.txt*. It is in the following location: `<EspressReportInstallDir>/userdb/config.txt`. A sample configuration file is shown below:

```
[port]
22071

[webroot]
C:\Intepub\wwwroot

[users]
{user-name encoded-password}
guest
Name1 Password1
Name2 Password2

[smtp host]
quadbase.com

[smtp secured]
true

[smtp ssl]
true
```

```
[smtp port]
587

[smtp username]
guest

[smtp password]
password
```

Under the `[port]` section, you can set the port number for EspressManager to use. This number is set to **`22071`** by default. You can also give an explicit IP address here for EspressManager to use instead of making EspressReport query the machine for the IP address. An explicit IP address can be given by adding the address after the port number. For example, changing:

```
[port]
22071
```

to

```
[port]
22071, 204.147.182.31
```

will result in EspressManager always using that IP address when it starts. You can specify multiple domains for EspressManager to establish a connection in the `[port]` section of the `config.txt` file. Domains are added after the IP address in the `[port]` section.

```
[port]
port number, ip address, domain1, domain2, ..., domain_n
```

The search sequence is IP address, domain_n, then domain1.

Under the `[user]` section, each line consists of username and password. Default username is **`guest`** with no password. You can remove this line and add as many additional users as you like. Each username and password are separated by one or more spaces (usernames and passwords cannot contain spaces). Also, usernames and passwords are case sensitive.

Each development license allows only one concurrent user to be logged on. If you purchased more than one development license, you can set up the development kits on separate machines or set up concurrent users on a single machine.

Note

There is no limit to the number of users that have login privileges to EspressManager. This license restriction applies only to the number of simultaneous users.

The `[smtp host]` section allows you to specify the smtp server name used for sending email notification or reports with the scheduler. For more information about using the scheduler, please see Section 1.13 - Scheduling.

If your SMTP server requires authentication, fill in the following sections as well. If your SMTP server uses SSL/TLS, set `[smtp ssl]` to `true`.

```
[smtp secured], [smtp ssl], [smtp port], [smtp username], [smtp password]
```

Otherwise, you can leave these sections blank.

When EspressManager starts for the first time, it will generate a configuration file called `EspressManager.cfg` in the directory in which it is started. This file contains the IP address and port number on which EspressManager is listening, as defined in the `config.txt` file. When Report Designer is started, it will look for this file for connection information to EspressManager. If it is not able to find this file, it will use the default IP of `127.0.0.1` (localhost) and port `22071`.

Other EspressReport components and applications/servlets/JSPs using the Report API will also (by default) look for the `EspressManager.cfg` file to make a connection to EspressManager. However, with the Report Viewer, Page Viewer, and Report API, you can explicitly set the IP and port that the component should use in order to connect to EspressManager. For more information about deploying EspressReport components, please see Section 2.7 - Deployment.

1.2.3.1. Increasing maximum memory heap size for applets

All applets have a maximum memory heap size of 16 megabytes by default. In Windows, this can be increased by going to `Control Panel` and selecting Java (or Java Plugin). Click on the *Java* tab and then *View* button. Enter **`-Xmx256M`** (without the quotes) under *Runtime Parameters* and click the *OK* button.

1.2.4. Starting EspressManager

Before ReportDesigner can be started, EspressManager must be running. In most cases, EspressManager should be started on the web server, but it is possible to use EspressReport on a stand-alone machine by running both EspressManager and ReportDesigner locally. The assigned IP number will be 127.0.0.1. This will create a log file `<EspressReportInstallDir>/EspressManager.log`, which can be used to monitor resource usage as well as diagnose problems. It is not necessary to install or start EspressManager on each ReportDesigner user machine.

To start EspressManager, run `EspressManager.bat` or `EspressManager.sh` file in the EspressReport root directory (this file is generated automatically during installation). For Windows installations, you can use program shortcuts that were added to Start menu or desktop, depending on the options that were specified during installation. EspressManager will then start automatically with default properties.

You can there also Add/delete/modify users.

"Manage Users" button can be used to add and remove user name and encrypted password in the config.txt file.

You can also configure EspressManager with your own settings using several arguments that have been provided. The argument starts with dash `-` followed by a command. No space is needed in each argument, but at least one space is required to separate different arguments.

You can enter these arguments in the command line when starting EspressManager and you can also edit the `espressmanager.bat`/`espressmanager.sh` file to add arguments.

Arguments in the text file are also picked up when EspressManager is started as a servlet process. For more information about this, please see Section 1.2.4.2 - Starting EspressManager as a Servlet.

-help

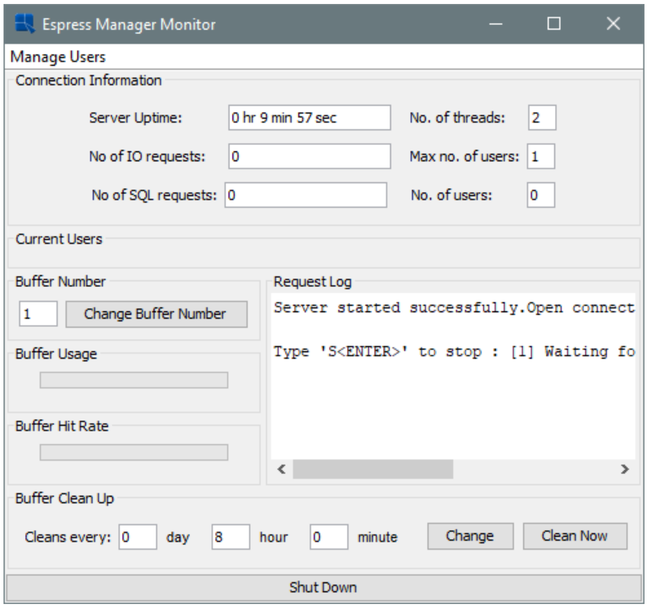
When you type **`-help`**, the EspressManager provides information on available arguments and their meaning. **`-h`** or **`-?`** can also be used to get the same online help information.

-log

When you type **`-log`** argument, a log file is created and all the client/server network information will be stored. An administrator can open the log file to evaluate each user's request. The log is saved as `EspressManager.log`.

-monitor:ON/OFF

When you specify **`-monitor`** argument, the EspressManager monitor appears as a graphic user interface that can be used to view the status as well as modify the EspressManager settings.



Server Uptime: 0 hr 9 min 57 sec

No. of threads: 2

No of IO requests: 0

Max no. of users: 1

No of SQL requests: 0

No. of users: 0

Buffer Number: 1

```
Server started successfully.Open connect

Type 'S<ENTER>' to stop : [1] Waiting fo
```

Cleans every: 0 day 8 hour 0 minute

EspressManager Monitor

-runInBackground:ON/OFF

This argument allows you to specify whether to run EspressManager as a background process or not. Running as a background process eliminates user input for shutdown and allows EspressManager to be run from a script. In order to make this argument work properly, it should be used in conjunction with the **`-monitor:OFF`** argument. When running EspressManager this way, the only way to shut it down is to end the process.

-recordLimit:nn

This argument allows you to set a maximum limit for the number of records EspressManager will retrieve from a database when executing a query. Once the maximum is reached, EspressManager will stop running the query. This feature prevents users from retrieving more records than can be stored in memory, and prevents crashes due to out of memory errors.

-queryTimeout:sss

This argument allows you to specify a timeout interval in seconds for chart queries. Once the query execution time passes the timeout argument, EspressReport will abort the query. This feature prevents users from accidentally creating run-away queries.

-DBBuffer:nnn

If a database is being used as data source for a chart, you can choose to buffer both the database connection and the data used for the chart using this argument. The number of connections and queries that will be stored depends on the number specified in the *DBBuffer* argument from 1 to 999.

-DBCleanAll:ddhhmm

Data in the data source may be updated periodically. Therefore, when the data buffer options is used (i.e. *DBBuffer* has a non-zero value), you may want to refresh the buffer to get the latest data. The **`-DBCleanAll`** argument can be used to indicate a time period after which data is cleared from the buffer and fetched from the data source. The value of ddhhmm means **`dd`** days, **`hh`** hours and **`mm`** minutes. EspressManager also supports omitted format, meaning that it will translate the value with the assumption that the omitted digits are the higher digits. For example:

> **`-DBCleanAll:101010`** means clean the buffer every 10 days, 10 hours, and 10 minutes.
>
> **`-DBCleanAll:1010`** means clean the buffer every 10 hours and 10 minutes.

`-DBCleanAll:10` means clean the buffer every 10 minutes.

This argument is invalid if the **`-DBCleanAll`** argument is set to 0 (i.e. always clean memory).

The refresh interval is printed when EspressManager is started.

-RequireLogin

This argument is used to apply security to ReportDesigner when it is launched via the API. Normally, when ReportDesigner is called via the API, there is no user authentication. This allows users to apply their own authentication to the programs, but it can also allow unauthorized users access to the server. To prevent this, users can turn on this argument (values are true/false) to force authentication for ReportDesigner when it is called from the API. With this argument turned on, every time a `QbReportDesigner` object is displayed, a correct username and password (defined in the `config.txt` file) must also be supplied via API method calls. For more information about calling ReportDesigner from the API, please see Section 2.3.5.7.11 - Calling Report Designer from Report API.

-QbDesignerPassword

This argument allows you to set password when the **`-RequireLogin`** argument is turned on. Instead of using a login from the `config.txt` file, you can set a specific password that must be supplied to the server via a method call when displaying a `QbReportDesigner` object. For more information about calling ReportDesigner from the API, please see Section 2.3.5.7.11 - Calling Report Designer from Report API.

-ScheduleCallBackClass:classfile:

This argument allows you to specify a class for the scheduler listener mechanism. The listener returns a handle to a scheduled report before the schedule executes. For more information about this feature, please see Section 2.4.8 - ICallBackScheduler Interface.

-ListenerManagerClass:classfile:

This argument allows you to specify a class for the scheduler listener mechanism. The listener returns a handle to a scheduled report before the schedule executes. For more information about this feature, please see Section 2.4.9 - Scheduler Listener.

-PageCleanUpTime:ddhhmm:

This argument allows you to specify a cleanup interval for the `/pages/` directory in EspressReport. Files are generated in this directory when users invoke the page viewer component using a report template (`.pak/.rpt`) or a `QbReport` object. For more information about the page viewer, please see Section 2.2 - Introduction to Page Viewer. The time arguments for this argument are the same as for the **`-DBCleanAll`** argument.

-MaxRecordInMemory:nnn

This argument allows you to set the maximum number of records that should be allowed in memory. Unlike the record limit which stops the query, this feature will begin paging the data to temporary files in the system when the threshold is reached. The viewing/exporting of the report will continue, but the data will be read from the paging files. This feature prevents the system from running out of memory when running reports.

-MaxCharForRecordFile:nnn

This argument allows you to set the maximum characters allowed per field in the paging file that is generated when record file storage is invoked. Any characters in a record in the data that are longer than this argument will be truncated in the finished report.

-FileRecordBufferSize:nnn

This argument allows you to set the number of records that should be paged at a time when the record file storage is invoked. The size of the buffer affects performance, so the larger the buffer size, the faster the report is generated.

-globalFormat:xmlfile:

This argument allows you to supply a global format XML file to set the default look and feel of report elements. Everytime a user selects to create a blank report, the format properties in this file will be applied. For this argument, file path to the XML file relative to EspressManager must be specified, i.e. (**`-globalFormat:Templates/FormatFile.xml`**). For more information about the global formatting features, please see Section 1.5.9.1 - Global Formatting.

-fontMapping:xmlfile: This argument allows you to supply a font mapping XML file to set the default font mapping for PDF export. Everytime a user creates a report, the system font mapping will be applied. For this argument, file path to the XML file relative to EspressManager must be specified, i.e. (**`-fontMapping:Templates/FontFile.xml`**). For more information about the font mapping features, please see Section 1.7.2.1 - PDF Font Mapping.

-htmlDpi:nn: This argument allows you to hard-code the screen resolution that should be used when exporting reports to DHTML or HTML format. By default, the system resolution is used. However, this can cause some discrepencies when reports are generated on a Linux or a Unix server and viewed on a Windows client. Generally, setting DPI to 96 (**`-htmlDpi:96`**) will produce a consistent export.

-singleTableForDistinct-ParamValue When you use **`-singleTableForDistinctParamValue`** argument, the parameter dialog for parameterized charts will be drawn differently. When you map a parameters to a database column, the distinct list will be drawn by running select distinct on the mapped field only. The default behavior (without this argument) will use the joins and conditions in the original query to constrain the distinct parameter list. For more information about parameterized queries, please see Section 1.3.2.2.2 - Parameterized Queries.

-schedulerBuffer:nnn: This argument allows you to create a pool of Reports for the scheduler to use. This feature can be helpful when you have multiple schedule jobs that run the same report. Instead of reloading the same report for each job, the jobs can use the report in the buffer. This improves schedule execution time. For large numbers of simultaneous schedule jobs, the best performance is achieved when the schedule buffer is set to the same size as the schedule thread limit.

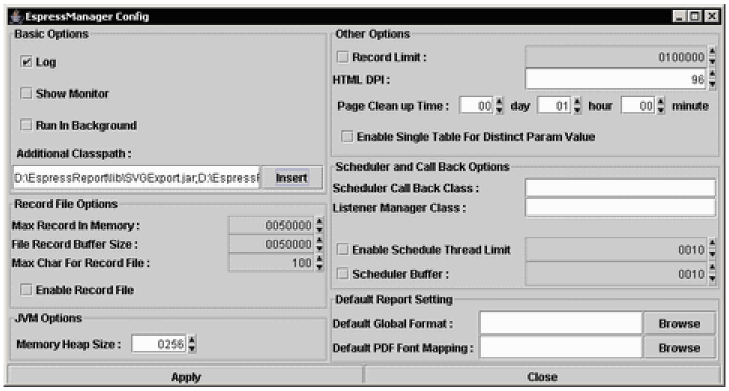
-xmlEncoding:encoding: This argument allows you to specify the encoding that EspressReport should use when writing XML files. This includes data registry files, XML report templates, XML exports, XML global formatting, and font mapping files. This parameter needs to be set in both EspressManager and Report Designer. For more information about XML encoding, please see Section 1.7.1.2 - XML Encoding.

-paperSize:LETTER/A4 Set this argument to define the default paper size. Note that you will need to set this parameter in the `reportdesigner.bat/.sh` file as well.

If you're running on Mac OS X and you selected to create aliases when installing, you will need to modify the `espressmanager.app` package to change the EspressManager settings. To do this, right-click (**Ctrl**+**Click**) on `espressmanager.app` and select *Show Package Contents* from the pop-up menu. Next, navigate to the Contents folder where you will see a file called `Info.plist`. Opening this file in a text editor adds the arguments to the **`lax.command.line.args`** under Java properties.

1.2.4.1. EspressManager Configuration Interface

Many of the EspressManager configuration features highlighted in the previous section can also be set using a stand-along EspressManager configuration interface provided with EspressReport. To launch the interface, make sure the EspressManager is not running, then execute `ManagerConfig.bat/.sh`. A configuration window will open.



☑ Log

□ Show Monitor

□ Run In Background

Additional Classpath : D:\EspressReport\lib\SVGExport.jar;D:\EspressR

Max Record In Memory : 0050000

File Record Buffer Size : 0050000

Max Char For Record File : 100

□ Enable Record File

Memory Heap Size : 0256

□ Record Limit : 0100000

HTML DPI : 96

Page Clean up Time : 00 day 01 hour 00 minute

□ Enable Single Table For Distinct Param Value

Scheduler Call Back Class :

Listener Manager Class :

□ Enable Schedule Thread Limit 0010

□ Scheduler Buffer : 0010

Default Global Format :

Default PDF Font Mapping :

EspressManager Configuration Window

The following configuration options can be set in this interface:

- Log
- EspressManager Monitor on/off
- Run in Background on/off
- Record Limit
- HTML DPI
- Page Clean-up Time
- Single Table for Distinct Param Value
- Scheduler Callback Class
- Listener Manager Class
- Scheduler Thread Limit
- Schedule Buffer
- Record File Settings
- Default Global Format
- Default PDF Font Mapping

For a description of all the options listed above, please see the previous section. In addition to the configuration options, users can also append EspressManager's CLASSPATH using this interface. Database drivers and other external classes can be added to EspressManager by clicking the *Insert* button and browsing to the desired file.

When you click the *Apply* button, all changes will be saved to `espressmanager.bat/.sh`.

1.2.4.2. Starting EspressManager as a Servlet

In addition to running as an application process, EspressManager can be run as a servlet within an application server/servlet runner. In this environment, EspressManager uses http to communicate with the client instead of sockets. The advantage to this configuration is that EspressManager can share the same port as the application server which makes it easier to deploy from behind firewalls. In addition, users can perform remote administration when running EspressManager this way.

To deploy EspressManager as a servlet, complete the following steps. As an example, Tomcat is located at C:\Tomcat and EspressReport installed to C:\ER70

1.2.4.2.1. Tomcat settings

To link Tomcat installation with our product installation, access the file "C:/Tomcat/conf/server.xml", and add your code defining one or more product installations between the "Valve" and "Host" sections.

```
<context path="/ER70" docBase="/home/user/Quadbase/TomcatER/ER"><Manager pathname=""/></context>
```

1.2.4.2.2. Startup settings

For Windows operating systems

Access the folder "C:/Tomcat/bin," copy the file "startup.bat" and paste it in your chosen installation of EC. Then, open the file and edit code to direct the Tomcat startup file to the Tomcat installation folder.

```
set "CATALINA_HOME=C:\Tomcat"
```

For Linux operating systems

Create a startup.sh file in <ER install directory> and make it executable.

Type the following two lines into the newly created <ER install directory>/startup.sh file

```
EXECUTABLE="<path to tomcat>/bin/catalina.sh"
```

```
exec "$EXECUTABLE" start "$@"
```

For example:

```
EXECUTABLE="/home/user/tomcat/bin/catalina.sh"
```

```
exec "$EXECUTABLE" start "$@"
```

> **Note**
>
> The rest of this guide is the same for Windows and Linux.

1.2.4.2.3. File launch settings

To make changes in the launching files, first, select the "EspressReport.jsp" find the end of the file and find the commented lines, and remove the comments.

```
<!-- Use these params if running EspressManager as servlet <param name="comm_protocol" value="servlet"/> <param name="comm_url" value="<%=codebase%>"/> <param name="servlet_context" value="servlet"/> -->
```

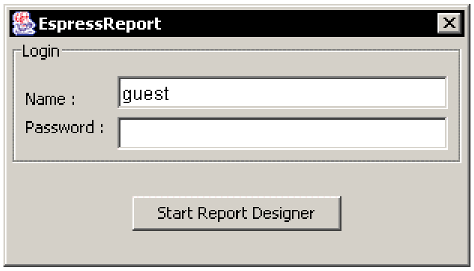
These are the parameters for connecting to EspressManager servlet, "servlet_context" may need edit, for example, in Tomcat/conf/servlet.xml, you configure your application as "/ER70", the value will be "ER70/servlet".

In the next step, you will need to edit "reportdesigner.bat" or "reportdesigner.sh" and append the following parameter at the end of the file: "-servlet:http://your-server-ip:your-port/ER70/servlet". Please note that this parameter must be placed at the end of the file, as placing it anywhere else will result in the designer being treated as a Java parameter and not run as intended.

When running EspressManager as a servlet, all components that connect to EspressManager need to be modified to take this into account. This includes Report Designer, Report Viewer, and any application/servlet/JSP using the Report API. For more information about EspressReport deployment, please see Section 2.7 - Deployment.

1.2.5. Starting Report Designer

Stand-Alone If you're running Report Designer locally, first start the EspressManager by executing `espressmanager.bat` or `.sh` file then execute `reportdesigner.bat` or `.sh` file (You can also execute `shortcuts/aliases` for Windows or Mac). A dialog box will appear prompting you to log on to the EspressManager. If you have set up users in the `config.txt` file enter your user name and password. If you have not set up users, enter the user name **`guest`** and password **`guest`**. Click the *Start Report Designer* button. This will launch the application.



guest

Report Designer Login Window

Browser If you're running Report Designer remotely, make sure that EspressManager is running on the remote machine, then type the URL for EspressReport in your browser, i.e. **`http://machinename/espressreport/index.html`**. The page will contain a dialog prompting you to log on to EspressManager. If you haven't modified the users in `config.txt` file, enter **`guest`** as the username with no password and then click the *Start Report Designer* button. This will launch the application in a new window.

> **Note**
>
> Because EspressReport requires a JVM the equivalent of Java 1.8 or higher, you will need to download the Java plugin in order to run Report Designer via browser. Also, the applet in the index.html page is configured to run on a Windows client. If you're running the applet on another platform, you will need to modify the HTML source of the page. In the HTML source, there are two applets - one of which is commented out. To run on a non-Windows client, comment out the first applet and un-comment the second one.

1.2.5.1. Connecting to EspressManager Running as a Servlet

If EspressManager is running as a servlet, as described in Section 1.2.4.2 - Starting EspressManager as a Servlet, you will need to make some modifications before ReportDesigner can be started successfully.

Stand-Alone If you're running ReportDesigner using `.bat` or `.sh` file, you will need to modify the file to indicate that EspressReport is running as a servlet and to point to the location of the EspressManager servlet. To do this, add the following argument to the `.bat` or `.sh` **`file -servlet:http://IP Address:Port/Context`**.

For example:

`-servlet:http://127.0.0.1:8080/servlet` indicates that the EspressManager is running on localhost with port 8080 in `/servlet/` context. Below is an example of the modified reportdesigner.bat

set JAVA_EXECUTABLE=C:\ER70\jre\bin\java set PATH=%PATH%;.\lib "%JAVA_EXECUTABLE%" -Xmx256M -classpath ".\lib\axis.jar;.\lib\axis-commons-logging.jar;.\lib\axis-commons-discovery.jar;.\lib\axis-jaxrpc.jar;.\lib\barbecue.jar;.\lib\ExportLib.jar;.\lib\FlashExport.jar;.\lib\hsqldb.jar;.\lib\javaws.jar;.\lib\jsqlparser.jar;.\lib\jsyntaxpane.jar;.\lib\qblicense.jar;.\lib\ReportDesigner.jar;.\lib\SVGExport.jar;.\lib\wsdl4j.jar;.\lib\xercesImpl.jar;.\lib\xml-apis.jar;.\lib\zxing-core.jar;.\lib\zxing-javase.jar;.;.\lib\commons-codec.jar;.\lib\commons-collections.jar;.\lib\commons-compress.jar;.\lib\commons-io.jar;.\lib\commons-math3.jar;.\lib\log4j-api.jar;.\lib\poi.jar;.\lib\poi-ooxml.jar;.\lib\poi-ooxml-schemas.jar;.\lib\SparseBitSet.jar;.\lib\xmlbeans.jar;" quadbase.reportdesigner.designer.ReportClient -templatesDirectory:Templates -servlet:http://192.168.0.51:8080/ER70/servlet

Browser If you're running ReportDesigner through a Java Web Start jnlp application that is connecting to EspressManager running as a servlet, you will need to add the following applet-desc parameter tags near the end of ER70\EspressReport.jsp file, which in our example, is shown below

The following parameter tags need to be added within the element of `applet-desc` of `jnlp` contents:

```
<applet-desc
      name="Espress Report Designer"
      main-class="quadbase.chart.designer.InitApplet"
      width="300"
      height="150">
      <param name="comm_protocol" value="servlet"/>
      <param name="comm_url" value="http://127.0.0.1:8080/
ER70"/>
      <param name="servlet_context" value="ER70/servlet"/>
  </applet-desc>
```

The first parameter indicates that the EspressManager is running as a servlet. The second one is the URL (saved in a JSP expression, `<%codebase%>`) that the application server is using, and the third one is the servlet context under which the EspressManager is deployed.

Restart your Tomcat server.

To access the ReportDesigner, open your web browser and go to the following address: **`http://127.0.0.1:8080/ER70/index.html`**. You may either need to open `EspressReport.jnlp` (Firefox/Chrome) directly or save it to your download directory and run the jnlp file from there to open the ReportDesigner (Microsoft Edge).

Tip

Sometimes `ReportDesgner.jnlp` cannot be launched after it is downloaded. In Windows 10, go to Java Control Panel to:

Enable 2 Options:

- From Security tab: Enable Java Content in Browser (Required for Applets and Web Start Applications)
- From Web Settings tab: Keep temporary files on my computer.

The jnlp file can be launched.

Caution

Jnlp file is cached since the "Keep Temporary files" enabled. You need go to Java Control Panel, from Web Settings, click *Delete Files* to delete "Cached Applications and Applets". Then you can get updated Jnlp file launched.

1.2.6. Backward compatibility patches

If you upgraded from an older version, you may notice some changes in the default behavior. Although the backward compatibility is kept as much as possible, sometimes a new behavior is preferred. The new behavior should be better for most users, but if you already have some charts or reports from an older version, you may want to keep the old behavior, so your charts and reports look exactly the same as before. That is why we provide backward compatibility patches.

Caution

Patches are for advanced users only. Please apply them only if you need them and if you know what you are doing. If you are not sure, please contact support.

The patches can be found in `<EspressReport_Installation_Directory>/lib/Patches` directory. They are stored in JAR archives. To apply a patch, you only need to add the appropriate JAR file to the classpath of your application as described below.

If you want to apply a patch to all designers (for reports and charts) and viewers, you have to edit `espressmanager.bat` and `reportdesigner.bat` files (if you use Windows), or `espressmanager.sh` and `reportdesigner.sh` files (if you use other OS) in your EspressReport installation directory and add relative path to the patch JAR file (e.g. `./lib/Patches/patch1.jar`) to the classpath parameter. If you are running Report Designer from HTML page, you also have to edit `index.html` file in your EspressReport installation directory and add relative path to the patch JAR file to the `archive` attribute of the `applet` tag.

If you are using API, you have to include the patch JAR file in the classpath of your application.

Below is a list of all available patches in the current version.

patch1.jar - turn off chart axis padding by default

- default behavior (without the patch) - axis padding is on by default
- behavior with the patch - axis padding is off by default
- new behavior has been introduced in version 4.0
- this feature can also be set using the `IAxis.setAxisPaddingAdded` method in API or in the Axis Scale dialog in the Chart Designer

patch2.jar - add left margin for annotation text in charts

- default behavior (without the patch) - annotation text does not have left margin
- behavior with the patch - annotation text has left margin
- new behavior has been introduced in version 5.0
- this feature cannot be set by public API nor UI
- annotation text is legend text, chart titles and any text inserted using Insert → Text in the Chart Designer

patch3.jar - use 0 to 1 as min/max value when chart axis autoscale for axis pt less than 1

- default behavior (without the patch) - maximum and minimum are always set according to data if autoscale is used
- behavior with the patch - if max-min is < 1 and autoscale is used, min is set to 0 and max is set to 1
- new behavior has been introduced in version 5.4
- this feature cannot be set by public API nor UI
- **It is not recommended to use this patch. The original behavior is a bug.**

patch4.jar - turn off new pie chart label placement algorithm (calculate label placement based on pie sector position)

- default behavior (without the patch) - new pie label placement algorithm is used
- behavior with the patch - old pie label placement algorithm is used
- new behavior has been introduced in version 6.0
- this feature cannot be set by public API nor UI

patch5.jar - always use integer value for chart axis auto scale

- default behavior (without the patch) - integer is always used for axis auto scale
- behavior with the patch - axis value data type is used for axis auto scale
- new behavior has been introduced in version 6.0
- this feature cannot be set by public API nor UI

patch6.jar - disable minimum and maximum error check for chart axis scale

- default behavior (without the patch) - the error check is enabled
- behavior with the patch - the error check is disabled (so you can set maximum value lower than the one set in your dataset - same for minimum)
- new behavior has been introduced in version 6.2
- patch is for API only
- this feature cannot be set by public API nor UI

patch7.jar - turn off Single color for categories feature for columnar and bar charts by default

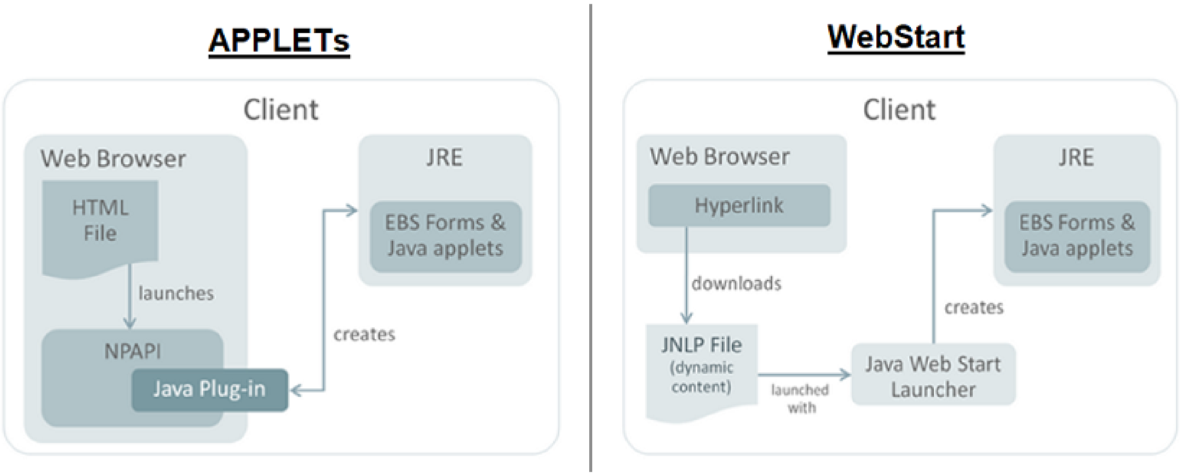
- default behavior (without the patch) - Single color for categories feature is turned on by default
- behavior with the patch - Single color for categories feature is turned off by default
- new behavior has been introduced in version 6.3
- this feature can also be set using the `IDataPointSet.setSingleColorForCategories` method in API or in the Chart Options dialog in the Chart Designer

patch8.jar - line chart end to end revert single point data to display on left axis

patch9.jar - display stack label despite not having enough space in the stack to render it

1.2.7. Run Applets in WebStart with JNLP file

Trying to run Java Applets in newer browsers and/or with newer Java versions can lead to problems with Applets being deprecated or not supported any more. Officially were Applets blocked to run in common browsers from about year 2016. However, it is still possible to run Applets in a similar use case as before (open a Java application from a remote server via your browser) thanks to Java WebStart technology that's supported by Java 8. Both Applets and Java Web Start are marked as deprecated technologies from Java 9.



Applets vs WebStart

In this case, EspressReport is Java 8 compiled application and is best to use it with Java 8. For using Applets with WebStart in newer versions, you must use some different implementation of WebStart. Popular choices are OpenWebStart or IcedTea-Web and it is already included in some newer Java distributions.

To run a Java Applet via a Java Web Start JNLP file, you have to specify the `applet-desc` parameter in the JNLP file configuration file.

```
   <applet-desc
      name="Chart Viewer"
      main-class="quadbase.chartviewer.Viewer"
      width="800"
      height="600">
 <param name="filename" value="help/examples/ChartAPI/data/test.tpl"/>
 <param name="preventSelfDestruct" value="true"/>
  </applet-desc>
```

The `param` tags specify parameters for the applet. The parameters are different for each applet. The specific parameters are mentioned in the documentation chapter for each applet.

JNLP files can be either run locally when the required libraries (containing the compiled source code) are loaded as a local file or remotely when the libraries are loaded via HTTP/HTTPS.

To run an Applet in a JNLP file remotely via HTTP/HTTPS, the EspressManager has to be run as a servlet in an application server. See the following chapter to learn how to do that: Section 1.2.4.2 - Starting EspressManager as a Servlet

When running JNLP files remotely via HTTP/HTTPS, the parameters `comm_protocol`, `comm_url` and `servlet_context` need to be added to the `applet-desc` element. You can either fill in the values manually (as the following code example shows) or you can have the values filled automatically in a JSP file.

```
<applet-desc
  name="Espress Report Designer"
  main-class="quadbase.reportdesigner.designer.ReportClient"
  width="380"
  height="160">
 <param name="comm_protocol" value="servlet"/>
 <param name="comm_url" value="http://127.0.0.1:8080"/>
 <param name="servlet_context" value="Espress70/servlet"/>
</applet-desc>
```

Note

If you want to pre-fil the values automatically in a JSP file, the actual values highly depend on your server configuration. The previous example is just illustrative.

1.3. Working with Data Sources

The first step in designing a report is to retrieve data you want to use. Within Report Designer, you can draw data from JDBC compliant databases, text files, XML files, EJBs, and even bring in object/array data through class files. All data source information is stored within the Data Source Manager.

1.3.1. The Data Source Manager

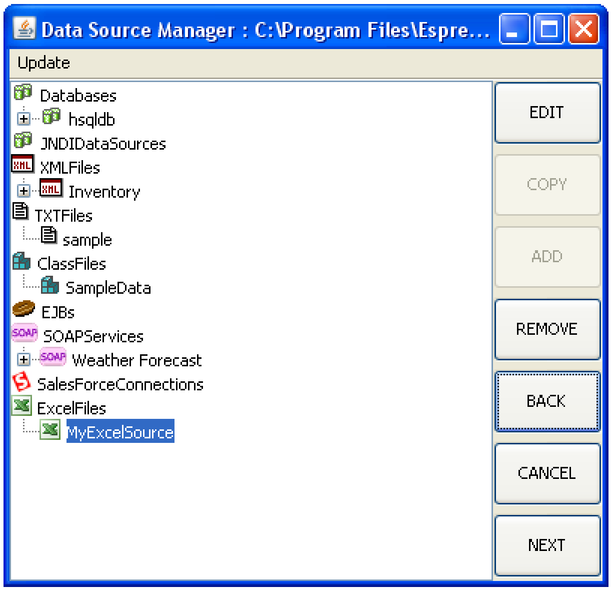
The Data Source Manager is an integrated utility that stores location and connection information for the data sources employed by a particular user. The information is stored in XML registry file. You can set up as many different data registry files as you like and there is no limit to the number of data sources that can be stored within a single registry. The registry is used as an aid at design time and is not required to deploy reports since the data or the data source information is stored within the report file.

Note

The XML data source repository only stores location and connection information, not the actual data. It is not an XML file that contains data to be used in a report.

1.3.1.1. Using Data Source Manager

When you start a new report (by selecting *New* from the File menu, or by clicking the *New* button on the toolbar), or when you first launch Report Designer, the report wizard will launch. The Wizard is a step-by-step process that will guide you through the basic construction of your report. The first dialog in the report wizard will prompt you to specify the data registry file that you would like to use, or to create a new registry. By default, the data source registry files are saved in DataRegistry directory. Once you either open an existing registry or start a new one, the Data Source Manager window will open.



Data Source Manager

Left side of the window contains a tree listing all data sources in the registry file. Grouped under *Databases* are individual databases and their associated queries and data views. Grouped under *JNDIDataSources* are database sources that use JNDI (Java Naming and Directory Interface) name to connect instead of JDBC. Grouped under *XMLFiles* are all XML files and their associated queries. Grouped under *TXTFiles* are all specified Text files. Grouped under *ClassFiles* are all specified class files. Grouped under *EJBs* are all specified EJB connections, and grouped under *SOAP* are all specified SOAP data sources.

Right side of the window contains a series of buttons controlling all of the functions of the Data Source manager. Each button performs the following function:

Edit: This option allows you to modify attributes of a data source. For a database, it allows you to change the connection information and modify queries/data views. For XML files, it allows you to change the file and its location, and modify XML queries. For text files, it allows you to change the display name and file location. For class files, it allows you to change the display name and modify the location. And for EJBs, it allows you to change the display name as well as parameter values.

Copy: This option is not only available for database nodes and bigdata nodes but also for the specified query or data view.

Add: This option allows you to add a data source. It will create a new source depending on which node is selected on the left side of the window. Hence, if you select *TXTFiles* and click the *Add* button, you will be prompted to add a new text file data source.

Remove: This option will remove the selected data source.

Back: This option allows you to go back one step in the wizard and change the registry file that you are using.

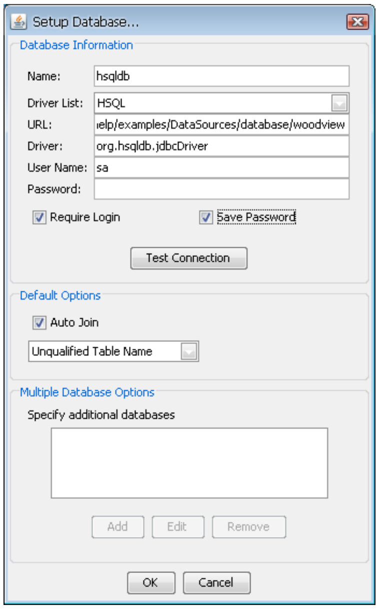
Cancel: This will cancel the wizard process.

Next: This will use the currently selected data source for the chart and proceed to the next step in the wizard.

1.3.2. Data from a Database

EspressReport can draw data from any JDBC compliant database. In order to connect to a database via a 3rd party driver (other than the JDBC bridge), you will need to modify the EspressManager so that it will pick up the classes for the driver. You can add the classes using the configuration interface (see Section 1.2.4.1 - EspressManager Configuration Interface), or by manually editing `EspressManager.bat/.sh` file. In this file, add the appropriate classes or archives to the "-classpath" argument. If you're running on Mac OS X and you selected to create aliases during installation, you will need to modify `espressmanager.app` package to add the JDBC driver to the classpath. To do this, right-click (**CTRL**+**Click**) on `espressmanager.app` and select *Show Package Contents* from the pop-up menu. Then navigate to the Contents folder where you will see a file called `Info.plist`. Open this file and add the appropriate classes or archives to the classpath argument. Note that JDBC drivers for MS SQL Server, MySQL, Oracle, Informix and PostgreSQL databases are included as a convenience. Other database JDBC jar files are not included due to licensing, multiple drivers, and/or other concerns, although support for these databases exist and the jar files can be explicitly added.

The first step in using a database as the data source is to set up a database in the registry and specify the connection information. To add a database, click on the *Databases* node and click the *Add* button. This will bring up a window prompting you to specify the connection information for that database. You can choose a database to connect to from the Driver List or specify the information manually. Fields to enter are database name, URL, and driver. You can also select whether the database requires a login and whether you want to save username and password information. If you select to save login and password information, you can then enter these informations in *User Name* and *Password* textboxes. Then click the *Ok* button and the new database will be added to the Data Source Manager window.



Name: hsqldb
Driver List: HSQL
URL: ıelp/examples/DataSources/database/woodview
Driver: org.hsqldb.jdbcDriver
User Name: sa
Password:

☑ Require Login ☑ Save Password

☑ Auto Join

Unqualified Table Name

Specify additional databases

Add Database Dialog

In order for EspressReport to create a connection to the database, the following information must be provided:

URL: This JDBC URL specifies the location of the database to be used. A standard JDBC URL has three parts, which are separated by colons:

jdbc:<subprotocol>:<subname>

The three parts of a JDBC URL are broken down as follows:

1. `jdbc` - the protocol. The protocol in a JDBC URL is always `jdbc`.

2. `<subprotocol>` - the name of the driver or the name of a database connectivity mechanism, which may be supported by one or more drivers. A prominent example of a subprotocol name is `odbc`, which has been reserved for URLs that specify ODBC data source names. For example, to access a database through a JDBC-ODBC bridge, you have to use a following URL: **jdbc:odbc:Northwind**

   In this example, the subprotocol is `odbc` and the subname `Northwind` is a local ODBC data source, i.e. `Northwind` is specified as a system DSN under ODBC.

3. `<subname>` - a way to identify the database. The subname can vary, depending on the subprotocol, and it can have a subsubname with any internal syntax the driver writer chooses. The function of a subname is to give enough information to locate the database. In the previous example, `Northwind` is enough because ODBC provides the remainder of the information.

Databases on a remote machine require additional information to be connected to. For example, if a database is to be accessed over your company Intranet, the network address should be included in the JDBC URL as part of the subname and should follow the standard URL naming convention of

//hostname:port/subsubname

Assuming you use a protocol called `VPN` for connecting to a machine on your company Intranet, the JDBC URL you need to use can look like this:

jdbc:vpn://dbserver:791/sales (similar to `jdbc:dbvendorname://machineName/SchemaName`)

It is important to remember that JDBC connects to a database driver, not the database itself. The JDBC URL that identifies the particular driver is determined by the database driver vendor. Usually, your database vendor also provides you with the appropriate drivers. It is highly recommended that users contact their database driver vendor for the correct JDBC URL that is needed to connect to the database driver.

Driver: This is the appropriate JDBC driver that is required to connect to the database. If you are using the JVM included within the installation (or Oracle's J2SE), use the following driver specification to connect to an ODBC data source: **sun.jdbc.odbc.JdbcOdbcDriver**

You can also specify a JDBC driver name specific to your database if you are NOT using the JDBC-ODBC bridge. For example, the Oracle database engine will require `oracle.jdbc.driver.OracleDriver`.

User Name: This is the login used for the database.

Password: Password for the above user.

Once you specify the connection information, you can test the database connection by clicking the *Test Connection* button. This will test the connection using the information you've provided and report any problems.

The *Default Options* portion of the dialog allows you to specify some properties for queries generated through the Query Builder interface or data views. You can specify whether to auto-join selected tables. Auto-join will attempt to join primary and foreign keys defined in the database. You can specify the table name format that should be used for queries either unqualified (only table name), or 2-part or 3-part qualified. Properties specified here will become the default setting for new queries and data views. They can also be modified for individual queries.

The *Multiple Database Options* portion of the dialog allows you to specify additional databases (i.e. additional database URLs) to obtain data from within the query. This option is only available when the database (original and any additional database) is MS SQL Server and 3-Part Qualified Table Name option is chosen. Note that the same login details as well as the same driver (as defined in the original connection) are used to connect to the specified additional databases as well. The query can obtain data by referencing a column in the additional database using a 3-Part table nomenclature.

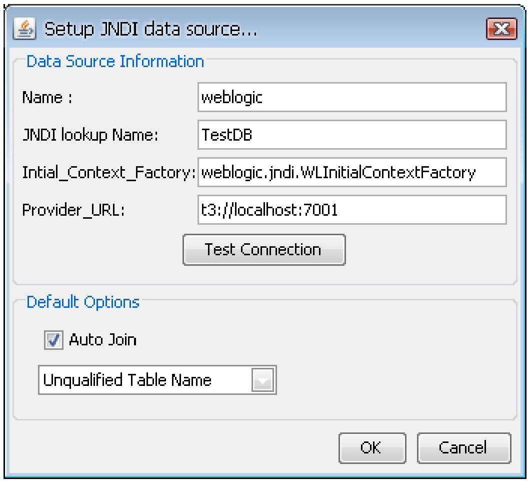
There are two sample databases included within the EspressReport installation. One is a HSQL (a pure Java application database) database and the other one is a MS Access database. Both contain the same data and are located in `help/examples/DataSources/database directory`. For details about how to set up connections to these sample databases, please see Section Q.3.2.1 - Setup A Database Connection.

1.3.2.1. JNDI Data Sources

In addition to connecting to databases via JDBC, EspressReport lets you use the JNDI (Java Naming and Directory Interface) to connect to data sources. In EspressReport, JNDI data sources are treated just like database data sources and support the same functionality (queries, parameters, data views, etc.). The advantage of using a JNDI data source is that it potentially makes it easier to migrate reports between environments. If data sources in both environments are set up with the same lookup name, reports can be migrated without any changes.

To connect to a JNDI data source in the EspressReport, you must have a data source deployed in your Web application environment and you must have EspressManager running as a servlet in the same environment. For more information about running EspressManager as a servlet, see Section 1.2.4.2 - Starting EspressManager as a Servlet.

To setup a JNDI data source, select the *JNDIDataSources* node in the Data Source Manager and click the *Add* button. This will bring up a dialog allowing you to specify the connection information.



Setup JNDI data source...

Data Source Information

Name : weblogic

JNDI lookup Name: TestDB

Intial_Context_Factory: weblogic.jndi.WLInitialContextFactory

Provider_URL: t3://localhost:7001

Default Options

☑ Auto Join

Unqualified Table Name

JNDI Setup Dialog

The first option allows you to specify a display name for the data source. The second option allows you to specify the JNDI lookup name for the data source. The third option allows you to specify the initial context factory for the data source, and the last option allows you to specify the provider URL. This information will vary depending on the application server you're using as different vendors implement JNDI data sources differently. You can test the connection by clicking the *Test Connection* button.

1.3.2.2. Queries

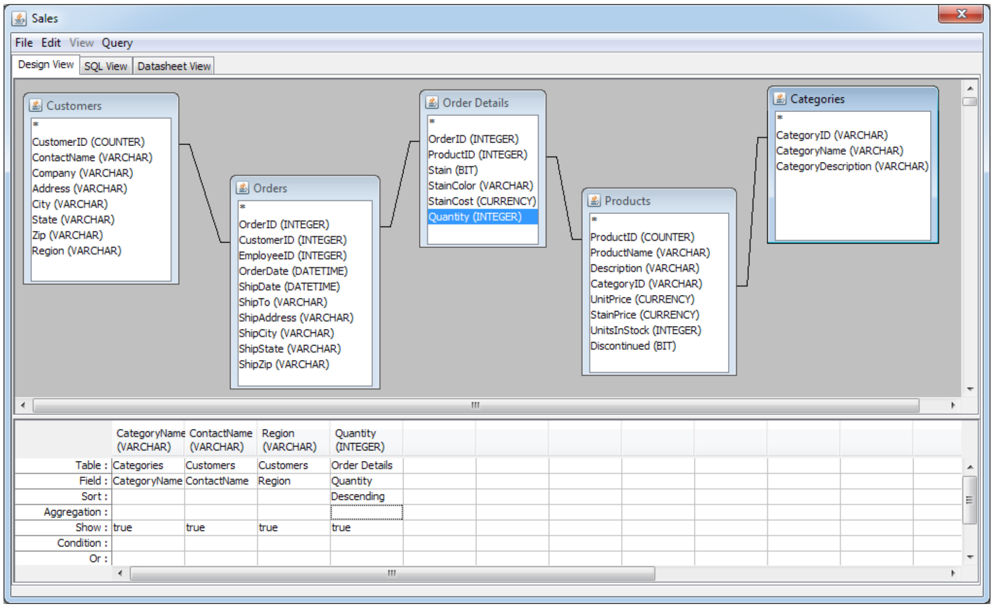
Once you add a database, a new node for your database will appear in the Data Source Manager window. When you expand the node, you will see two more nodes, one called *Queries* and one called *Data Views*. These are the two ways to retrieve data from your database. To create a new query, select the *Queries* node and click the *Add* button. A dialog will come up prompting you to specify a query name and select whether you would like to enter SQL statement or launch the Query Builder.

If you select to enter an SQL statement, a dialog box will come up allowing you to type in your SQL statement. From this dialog, you can also load a QRY or text file containing SQL text, or execute a stored procedure. If you select to launch the Query Builder, the Query Builder will open in a new window, allowing you to construct the query visually. After you finish building or entering the query, you will return to the Data Source Manager window and the query will appear as a new entry under the *Queries* node for your database.

1.3.2.2.1. Using Query Builder

The Query Builder is an integrated utility that allows you to construct queries against relational databases in a visual environment. To launch the query builder, add a new query within the Data Source Manager and select the *Open Query Builder* option. The Query Builder will then open in a new window. You can also launch the query builder to modify an existing query by double clicking the query name in the Data Source Manager.

The main Query Builder window consists of two parts. The top half of the window contains all the database tables selected for the queries and their associated columns. The top window also shows what joins have been set up between column fields. The lower half of the main window or QBE (query by example) window contains columns that have been selected or built for the query and their associated conditions.



	CategoryName (VARCHAR)	ContactName (VARCHAR)	Region (VARCHAR)	Quantity (INTEGER)
Table :	Categories	Customers	Customers	Order Details
Field :	CategoryName	ContactName	Region	Quantity
Sort :				Descending
Aggregation :				
Show :	true	true	true	true
Condition :				
Or :				

Query Builder Window

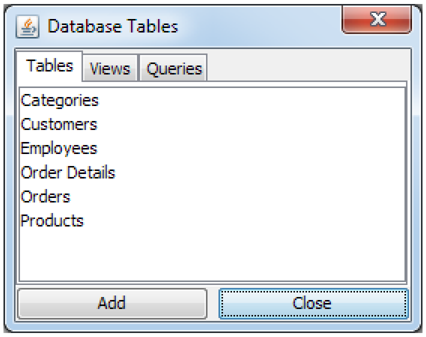
There are three tabs at the top of the Query Builder window. These allow you to toggle between different views. The *Design View* tab is the main designer window described above, the *SQL View* tab shows the SQL statement that is generated by the current query and the *Datasheet View* tab shows the query result.

When you finish constructing the query, select *Done* from the File menu to return to the Data Source Manager.

1.3.2.2.1.1. Tables

When the Query Builder launches for the first time, a tabbed window will appear containing a list of all the tables within the database. A second tab contains a list of all the views in the database, and the third tab contains a list of other queries you have designed for the database under a heading called *Queries*. From this window, you can select the tables/views/queries from which you would like to build the query. You can also load a previously designed query as a table. To add a table, select it and click the *Add* button or double click on the table name. When a table is added, it will appear in the main Query Builder window and will display all columns within that table. To remove a table, right click within the table and select *Delete* from the pop-up menu. You can also specify a table alias and sort the fields alphabetically from this menu. You can close the tables window by clicking the *Close* button. To re-open it, select *Show Tables* from the *Query* menu.

> **Note**
>
> By default, the tables will appear using the name format you specified when setting up the database connection. You can change the naming by selecting *Table Name Format* from the Query menu.



Query Builder Tables Window

1.3.2.2.1.2. Joins

When you select database tables for the query, the Query Builder can auto-detect joins between column fields based on primary key-foreign key relationships in the database. Auto-joins will be added depending on which option you selected when setting up the database connection. Auto-joins will create a standard join between tables. A join is represented by a line drawn between two fields in the top half of the design window. To remove a join or edit its properties, right click on the line and select your choice from the pop-up menu. To add a join, click and drag one column field to another in a different table. A join will then appear. You can change the auto-join settings by selecting *Auto Join* from the Query menu.

Join Properties: Selecting *Join Properties* from the pop-up menu will bring up three options allowing you to select the type of join used between the column fields. Query Builder only supports equi-joins. Inequality joins can be easily accomplished using the *conditions* field. You can specify inner joins, left outer joins, and right outer joins. See the examples below for an explanation of the different join types.

Suppose you have the following two tables: Customers and Orders

CustomerID	CustomerName
1	Bob
2	Ivan
3	Sarah
4	Randy
5	Jennifer

OrderID	CustomerID	Sales
1	4	$2,224
2	3	$1,224
3	4	$3,115
4	2	$1,221

An inner join on **CustomerID** on the two tables will result in combining rows from the **Customers** and **Orders** tables in such a way that each row from the **Customers** table will be “joined” with all the rows in the **Orders** table with the matching **CustomerID** value. Rows from the **Customers** table with no matching **CustomerID** fields from the **Orders** table will not be included in the query result set.

Now suppose you create a query by selecting the **OrderID**, **CustomerName**, and **Sales** fields with an inner join on the **CustomerID** field. The select statement generated by the Query Builder would look like this:

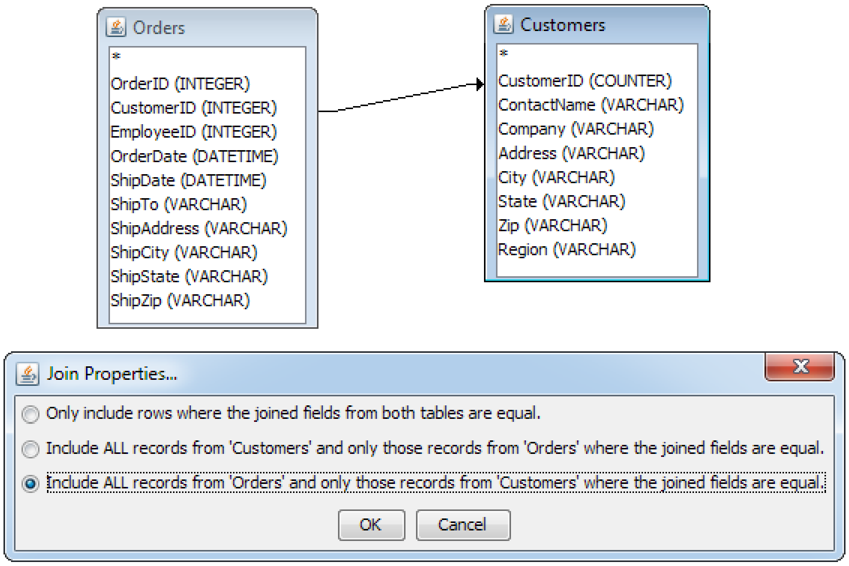
```
Select Orders.OrderID, Customers.CustomerName, Orders.Sales
From Customers, Orders
Where Customers.CustomerID = Orders.CustomerID
Order by Orders.OrderID;
```

The result of the query is shown below:

OrderID	CustomerName	Sales
1	Randy	$2,224
2	Sarah	$1,224
3	Randy	$3,115
4	Ivan	$1,221

As you can see, the **CustomerName** entries “Bob” and “Jennifer” do not appear in the result set. This is because neither customer has placed an order. There are situations where you may want to include all the records (in this example customer names) regardless whether matching records exist in the related tables(s) (in this case the **Orders** table). You can achieve this result using outer joins.

The Query Builder gives you the option of either right or left outer joins. The keywords “right” and “left” are not significant. It is determined by the order that the tables are selected in the Query Builder. If the outer table (the one that will have all records included regardless of matching join condition) is selected first, then Query Builder will use a right outer join. If the outer table is selected after the other join table, a left outer join is used. In our example, the **Customers** table has been selected before the **Orders** table, hence to select all of the records from the **CustomerName** field, the Query Builder will use a right outer join on the **CustomerID** fields.



○ Only include rows where the joined fields from both tables are equal.

○ Include ALL records from 'Customers' and only those records from 'Orders' where the joined fields are equal.

● Include ALL records from 'Orders' and only those records from 'Customers' where the joined fields are equal.

Join Properties Dialog

Now, using the previous example, suppose you create the same query as before, except this time you specify to include all records from the **Customers** table. The select statement generated by the Query Builder would look like this:

```
Select Orders.OrderID, Customers.CustomerName, Orders.Sales
From Orders right outer join Customers on Orders.CustomerID =
 Customers.CustomerID
Order by Orders.OrderID;
```

The result of the new query is shown below:

OrderID	CustomerName	Sales
	Jennifer	
	Bob	
1	Randy	$2,224
2	Sarah	$1,224
3	Randy	$3,115
4	Ivan	$1,221

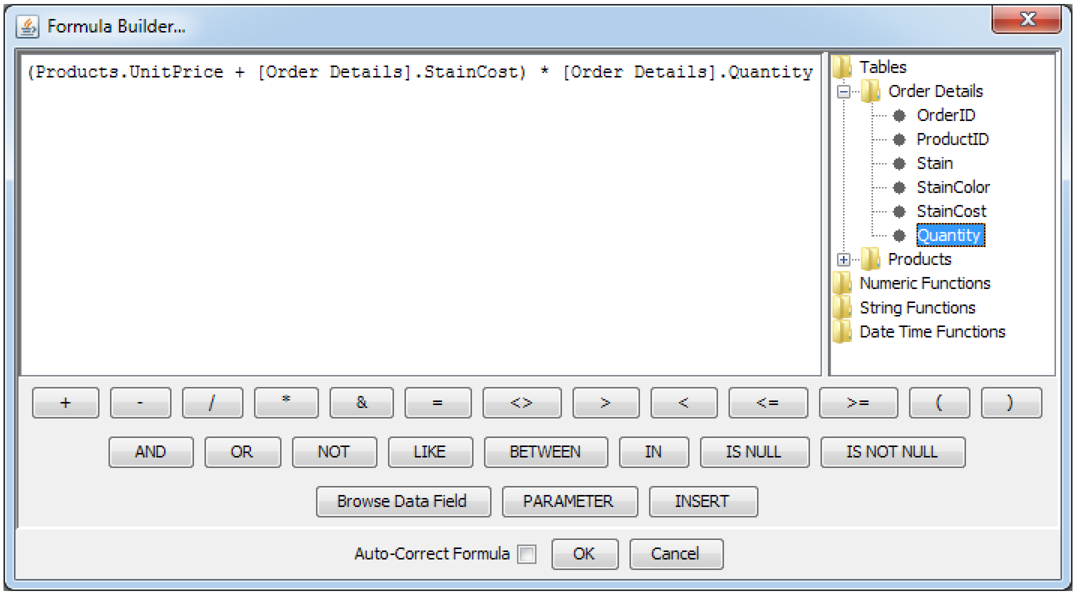
As you can see, all of the customer names have now been selected and null values have been inserted into the result set where there are no corresponding records. If you specify an outer join, the join line connecting the two tables in the Query Builder will become an arrow in the direction of the join.

1.3.2.2.1.3. Columns

The QBE window contains information on column fields selected for the query, as well as any conditions for the selection.

Selecting Column Fields: You can add column fields to the query from any table that has been selected in one of two ways. You can double-click on a field name within a table to add it to the query, or you can double-click on the *Table* or *Field* fields to bring up a drop-down menu with field choices. You can remove a column from the query by right clicking in the lower window and selecting *Delete Column* from the pop-up menu, or by selecting Edit → Delete Column. Once you select a column field, you can specify how you want to sort the column, in ascending or descending order, by double clicking on the *Sort* field. You can also specify group by or column aggregation by double clicking on the *Aggregation* field. Aggregation options include: *Group By*, *Sum*, *Average*, *Min*, *Max*, *Count*, *Standard Deviation*, *Variance*, *First*, *Last*, and *Where*. If you select group by for one column, then you are required to specify group by (or aggregation) for all of the other columns. To specify a column alias, right click on the column and select *Alias* from the pop-up menu.

Building Columns: To build your own column, right click on a blank column in the QBE window and select *Build* from the pop-up menu. This will launch the Formula Builder. The Formula Builder allows you to construct columns in a visual environment using the tables that you have selected and the formula library for the database that you are using.



```
(Products.UnitPrice + [Order Details].StainCost) * [Order Details].Quantity
```

Formula Builder Window

Conditions: You can place conditions on the query selection by entering them in the *Condition* or *Or* fields. A condition placed in the *Condition* field creates an `AND` clause within the generated SQL. A condition placed in the *Or* field creates an `OR` clause within the SQL. Right clicking in either field and selecting *Build* from the pop-up menu will bring up the Formula Builder. In the Formula Builder, you can specify standard conditions, `=, <, >, BETWEEN, LIKE, NOT`, etc., as well as construct formulas to filter the query. You can also specify a query parameter here.

Note

EspressReport can auto-correct items entered as query conditions by appropriately appending the field name and encasing string arguments in quotes. For example, if you enter `= ARC`, EspressReport will change the query condition to `Categories.CategoryName='ARC'`. If you're using complex functions (i.e. database functions that take multiple string arguments), EspressReport may not be able to properly parse the function. You can turn off the auto-correct feature by un-checking the box at the bottom of the formula builder window.

1.3.2.2.1.4. Using Database Functions

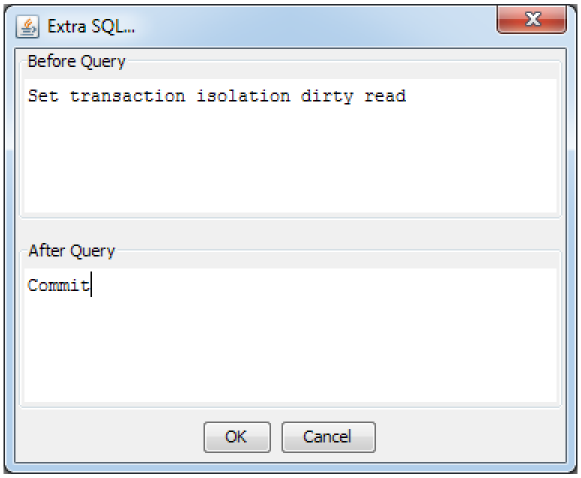
The formula builder component in the query builder allows you to use database specific functions when building a column or condition for the query. You can use the functions that are supplied or add your own to the interface.

EspressReport comes with the function libraries for Oracle, Access, MS SQL, and DB2 pre-loaded. They are stored in XML format in `DatabaseFunctions.xml` file in `userdb` directory. For databases with functions not stored in XML, EspressReport will use default ones. You can specify different database functions by editing the XML file or creating a new one based on `DatabaseFunctions.dtd` file in `userdb` directory. A sample database functions file might look like this:

```
<DatabaseFunctions>
      <Database ProductName="ACCESS">
            <FunctionSet Name="Numeric Functions">
                  <Function>Abs(number)</Function>
                  <Function>Atn(number)</Function>
            </FunctionSet>
      </Database>
</DatabaseFunctions>
```

1.3.2.2.1.5. Adding Extra SQL

Sometimes it is necessary to add extra SQL statements to run before or after a query. For example, you may need to set a transaction level or call a stored procedure before executing a query and/or commit a transaction or drop a temporary table after executing a query. The query builder allows you to specify these extra SQL statements by selecting *Extra SQL* from the Query menu. This will bring up a window allowing you to write statements to execute before and/or after a query.



```
Set transaction isolation dirty read
```

```
Commit
```

Extra SQL Dialog

You can enter any SQL statements you would like to run before and/or after the query in the appropriate boxes. Once you finish, click the *Ok* button and the statements will be added to the query.

1.3.2.2.1.6. Query Output

The *SQL View* and *Datasheet View* tabs allow you to see two different views of the query.

SQL View: The *SQL View* tab shows you the SQL statement generated by the query in the design view. It allows you to see how the Query Builder is translating different operations into SQL. You can edit the generated SQL if you want, however, if you change the SQL and then return to the *Design View*, all changes you made will be lost. If you save a query after changing the SQL, the query will re-open in the *SQL View* tab if you select to edit it.

Datasheet View: The *Datasheet View* tab shows you the query result in data table form (this tab is also available in the Enter SQL dialog). The datasheet view will show you all the data that is drawn as a result of executing the query. Going to the datasheet view will also test the query to check for design errors. You can navigate the query result by using the toolbar at the bottom of the window.

⏮ Go to the first page of the data table

⏪ Go to the previous page of the data table

● Go to a specific row of data

⏩ Go to the next page of the data table

⏭ Go to the last page of the data table

☰ Set number of rows to display per page (default is 30)

Exporting Queries: You can export queries in one of two ways. You can output the SQL statement as a text or you can output the query result as CSV file. To export a query, select *Export* from the File menu. A second menu will appear giving you the option to *Generate SQL* or *Generate CSV*. Select the desired option and a dialog box will appear prompting you to specify the filename and location.

> **Note**
>
> To save the query and exit the Query Builder, select *Done* from the File menu.

1.3.2.2.2. Parameterized Queries

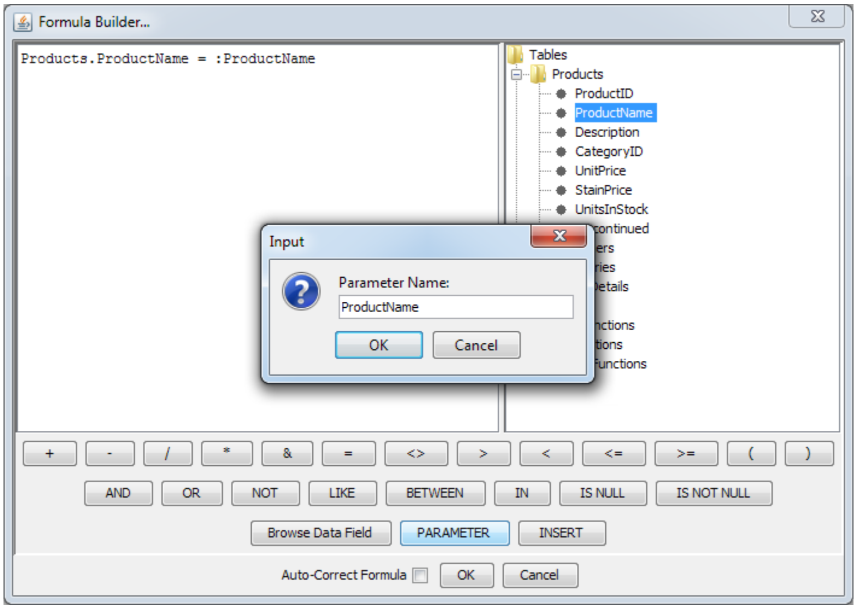
You can also use the Query Builder to design parameterized queries. This feature allows you to filter the data at run-time. Parameterized queries are also used for parameter drill-down in charts and reports. For more information about drill-down reports, please see Section 1.10 - Drill-Down.

Query parameters can be defined when typing an SQL statement or using the Query Builder. They can also be defined when running data views (this is covered in the next section). A parameter is specified within an SQL statement by the `":"` character. Generally, the parameter is placed in the `WHERE` clause of an SQL Select statement. For example, the following SQL statement

```
Select * From Products Where ProductName = :Name
```

specifies a parameter called `Name`. You can then enter a product name at run-time and only retrieve data for that product.

Within the Query Builder, you can specify a query parameter by right clicking on the *Condition* field and selecting *Build* from the pop-up menu. The Formula Builder will open, allowing you to place a condition on the column.



```
Products.ProductName = :ProductName
```

Parameter Name: ProductName

Specifying a Parameter in the Formula Builder

You can insert a parameter by clicking the *PARAMETER* button. A second dialog will appear prompting you to specify a name for the parameter. Type the parameter name, click *OK* and then click *OK* again to close the formula builder. You can specify as many different parameters for query as you like.

1.3.2.2.2.1. Multi-Value Parameters

EspressReport supports a special kind of parameter that takes an array of values as input rather than a single value. Multi-value parameters are useful when you want to filter the result set based on an unknown number of values. For example, say a report is run to return a list of customers for a specific state/province. Users could select as many different states/provinces as they wanted and return the relevant information.

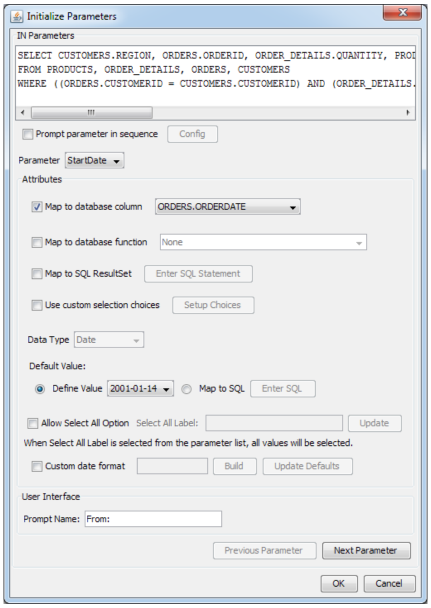
To create a multi-value parameter, place a parameter within an `IN` clause in an SQL statement. For example, the following query

```
Select Customers.Company, Customers.Address, Customers.City,
 Customers.State, Customers.Zip
From Customers
Where Customers.State IN (:State);
```

will create a multi-value parameter named `State`. Multi-value parameters will only be created if there is only one parameter in the `IN` clause. If you place more than one parameter in the `IN` clause, i.e. `Customers.State IN (:State1, :State2, :State3)`, it will create three single value parameters instead.

1.3.2.2.2.2. Initializing Query Parameters

When you attempt to save (by selecting *Done* from the File menu) or preview (by clicking the *Datasheet View* tab) a parameterized query, you will first be prompted to initialize the parameter. You can also initialize it by selecting *Initialize Parameters...* from the Query menu or by clicking the *Initialize Parameters* button in the Enter SQL Dialog.



```
SELECT CUSTOMERS.REGION, ORDERS.ORDERID, ORDER_DETAILS.QUANTITY, PROI
FROM PRODUCTS, ORDER_DETAILS, ORDERS, CUSTOMERS
WHERE ((ORDERS.CUSTOMERID = CUSTOMERS.CUSTOMERID) AND (ORDER_DETAILS.
```

□ Prompt parameter in sequence

Parameter: StartDate

☑ Map to database column: ORDERS.ORDERDATE

□ Map to database function: None

□ Map to SQL ResultSet

□ Use custom selection choices

Data Type: Date

● Define Value: 2001-01-14 ○ Map to SQL

□ Allow Select All Option Select All Label:

When Select All Label is selected from the parameter list, all values will be selected.

□ Custom date format

Prompt Name: From:

Initialize Parameter Dialog

From this dialog you can specify the following options:

Map to database column:

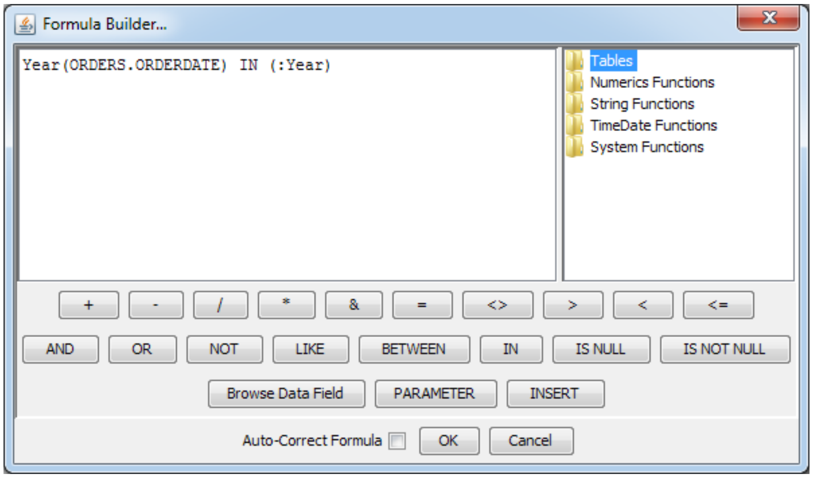
This option allows you to specify a column from the database whose values will be used for the parameter input. Selecting this option modifies the parameter prompt that the end user will get when previewing or running the report in the Report Viewer. If you map the parameter to a database column, then the user will be prompted with a drop-down list of distinct values from which to select a parameter value. If you do not map, the user will have to type in a specific parameter value.

Tip

Normally, this drop-down list is populated by running a select distinct on the column while applying the joins and conditions from the query. If you prefer to get all data from the column without constraints (sometimes this can improve performance of the parameter prompts), you can set the `-singleTableForDistinctParamValue` argument when starting EspressManager. For more information about EspressManager configuration options, see Section 1.2.4 - Starting EspressManager

Map to database function:

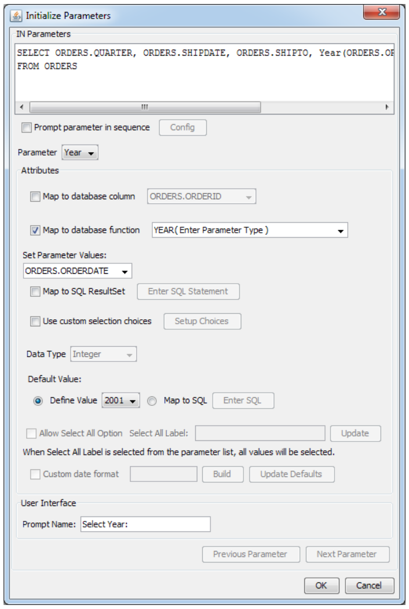
The *map to database column* feature is very handy if you want to enter a valid value for a parameter from a list box, but sometimes you rather want a computed value or a derived value from a database column. For example, you want to find all orders from year 2007. However, OrderDate is a date. What you want is to apply the *Year* function to the OrderDate column. This is the impetus behind this feature. Mapping a parameter to a database function is very similar to mapping to a column. In the formula builder, enter a condition comparing a function result to a parameter as shown below:



```
Year(ORDERS.ORDERDATE) IN (:Year)
```

Condition for Mapping to Database Function

In the initialize parameter dialog, check the *Map to database* function checkbox and the values will be automatically filled in.



```
SELECT ORDERS.QUARTER, ORDERS.SHIPDATE, ORDERS.SHIPTO, Year(ORDERS.OR
FROM ORDERS
```

□ Prompt parameter in sequence

Parameter: Year

□ Map to database column: ORDERS.ORDERID

☑ Map to database function: YEAR(Enter Parameter Type)

Set Parameter Values: ORDERS.ORDERDATE

□ Map to SQL ResultSet

□ Use custom selection choices

Data Type: Integer

Default Value: ● Define Value 2001 ○ Map to SQL

□ Allow Select All Option Select All Label:

When Select All Label is selected from the parameter list, all values will be selected.

□ Custom date format

Prompt Name: Select Year:

Map Parameter to Database Function

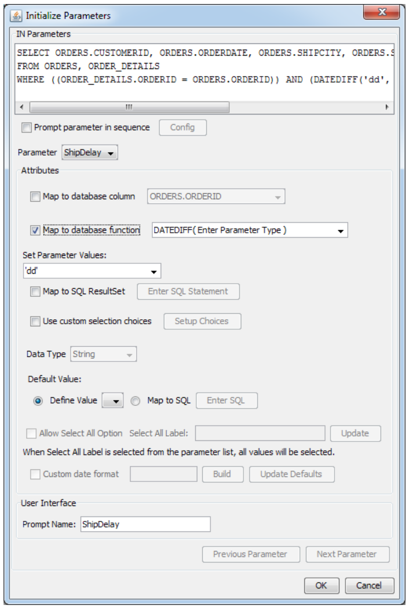
The list of custom functions is extracted from `DatabaseFunctions.xml` file located in `/userdb/` directory. Modify the .xml file if you wish to add a new database or custom functions. The new functions will appear in this list after you restart the program.

If your database is not listed in the `.xml` file, the function list will be populated with functions listed in the JDBC driver. However, the function parameters are not provided. For example, the HSQL database will not be listed in the `.xml` file.

An interesting example using the HSQL database is as follows. Suppose you would like to create a report for orders that were delayed. You can utilize the HSQL DateDiff function to find the number of days for the order to ship.

```
DATEDIFF('dd', ORDERS.ORDERDATE, ORDERS.SHIPDATE)
 >= :ShipDelay
```

This function finds the difference between the order date and the ship date and displays the result in terms of days. If you initialize the parameter and check the map to database function, the following window will be displayed:



Initialize Parameters

IN Parameters

```
SELECT ORDERS.CUSTOMERID, ORDERS.ORDERDATE, ORDERS.SHIPCITY, ORDERS.S
FROM ORDERS, ORDER_DETAILS
WHERE ((ORDER_DETAILS.ORDERID = ORDERS.ORDERID)) AND (DATEDIFF('dd',
```

☐ Prompt parameter in sequence

Parameter: ShipDelay

Attributes

☐ Map to database column: ORDERS.ORDERID

☑ Map to database function: DATEDIFF(Enter Parameter Type)

Set Parameter Values: 'dd'

☐ Map to SQL ResultSet

☐ Use custom selection choices

Data Type: String

Default Value: ● Define Value ○ Map to SQL

☐ Allow Select All Option Select All Label:

When Select All Label is selected from the parameter list, all values will be selected.

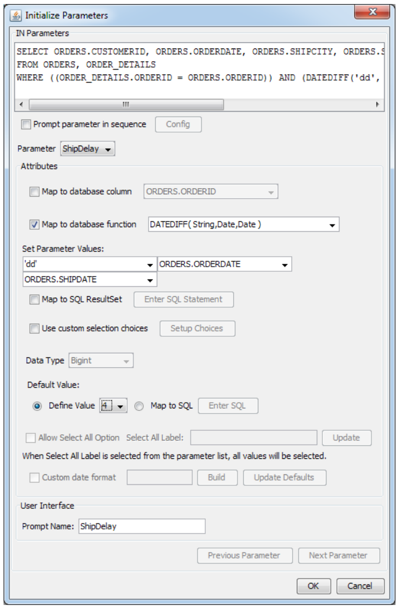
☐ Custom date format

User Interface

Prompt Name: ShipDelay

No Parameter Types for HSQL Function

The `DateDiff` function requires a string and two date values for the parameters. Enter these parameter types in the parentheses. This will bring up three set parameter value lists. Enter **dd** (day) for the first parameter, select `Orders.OrderDate` from the list for the second parameter, and select `Orders.ShipDate` from the list for the third parameter. The default values will be updated with the function results.



Initialize Parameters

IN Parameters

```
SELECT ORDERS.CUSTOMERID, ORDERS.ORDERDATE, ORDERS.SHIPCITY, ORDERS.S
FROM ORDERS, ORDER_DETAILS
WHERE ((ORDER_DETAILS.ORDERID = ORDERS.ORDERID)) AND (DATEDIFF('dd',
```

☐ Prompt parameter in sequence

Parameter: ShipDelay

Attributes

☐ Map to database column: ORDERS.ORDERID

☑ Map to database function: DATEDIFF(String,Date,Date)

Set Parameter Values: 'dd' ORDERS.ORDERDATE ORDERS.SHIPDATE

☐ Map to SQL ResultSet

☐ Use custom selection choices

Data Type: Bigint

Default Value: ● Define Value 4 ○ Map to SQL

☐ Allow Select All Option Select All Label:

When Select All Label is selected from the parameter list, all values will be selected.

☐ Custom date format

User Interface

Prompt Name: ShipDelay

Map Parameter to HSQL Function

Map to SQL ResultSet:

A parameter mapped to a database column will give you a list of distinct values in a drop-down list box for the user to choose from when running the report. However, to produce the list of values, a select distinct on the column with the joins and conditions from the query will be run. In some cases, this can be a time-consuming process. To obviate this problem, and in fact gain complete control as to what and how to populate the drop-down list box, you can write your own select statement to populate the drop-down list. An added bonus is that parameters that are in the query can be included in this query. With proper joins and parameters included, you can use this feature to facilitate cascading parameters (See Section 1.3.2.2.2.3 - Cascading Parameters). Below is an example:

Suppose you have two parameters in the query. So, your query is as follows:

```
SELECT CATEGORIES.CATEGORYID,
 PRODUCTS.PRODUCTNAME, PRODUCTS.UNITPRICE,
 PRODUCTS.UNITSINSTOCK
FROM PRODUCTS, CATEGORIES
WHERE ((PRODUCTS.CATEGORYID =
 CATEGORIES.CATEGORYID))
AND (((CATEGORIES.CATEGORYID =:category) AND
 (PRODUCTS.PRODUCTNAME =:product)))
```

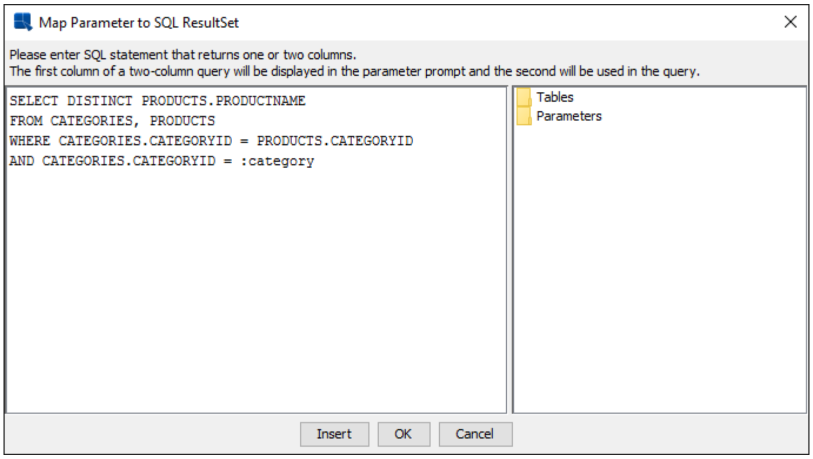
In the `config` prompt in `initialize parameter`, set the order for parameter prompting to `category` first, then `product`.

The select statement for parameter `category` can be the following:

```
SELECT DISTINCT CATEGORIES.CATEGORYID
FROM CATEGORIES
```

The select statement for parameter `product` will be as shown below:

```
SELECT DISTINCT PRODUCTS.PRODUCTNAME
FROM CATEGORIES, PRODUCTS
WHERE CATEGORIES.CATEGORYID = PRODUCTS.CATEGORYID
AND CATEGORIES.CATEGORYID = :category
```



Map Parameter to SQL ResultSet

Please enter SQL statement that returns one or two columns.
The first column of a two-column query will be displayed in the parameter prompt and the second will be used in the query.

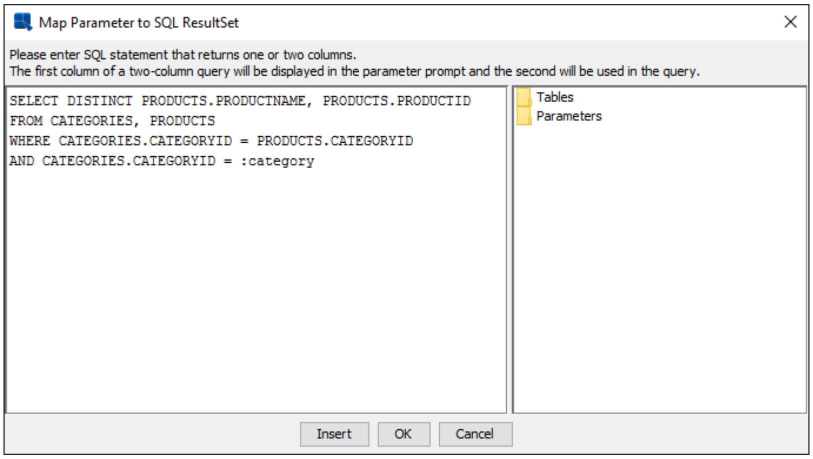
```
SELECT DISTINCT PRODUCTS.PRODUCTNAME
FROM CATEGORIES, PRODUCTS
WHERE CATEGORIES.CATEGORYID = PRODUCTS.CATEGORYID
AND CATEGORIES.CATEGORYID = :category
```

Select Statement for Product

When the user runs the template, `category` will be prompted first. Then the value of `category` chosen will be used to filter for `product`.

The select statement mapped to a parameter can have either one or two columns in the select list. It is clear that if one column is in the select list, it must be the column that supplies list of distinct values for the parameter. Another useful feature provided here is that you can actually select two columns in the select list such that the first one of the columns will supply values for the drop-down list while the second column will be the actual parameter value for the filter condition. Consider the following example.

Suppose your database has a table with product ID as the primary key. When your end user wants to search for products from the database, they want to use the *product name* as parameter (therefore it is listed in the query as first) since a *product ID* could be just a cryptic code (therefore it is listed in the query as second). Using this feature, you can choose product name for the values in the drop-down list and product ID as the actual value filter condition.



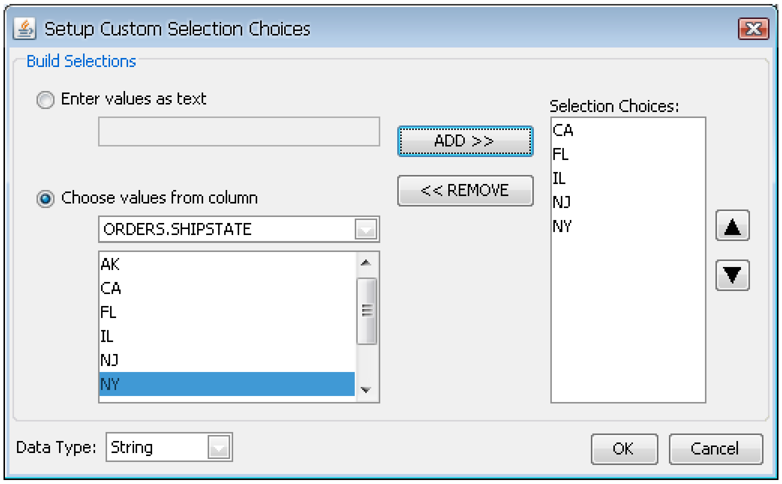
Please enter SQL statement that returns one or two columns.
The first column of a two-column query will be displayed in the parameter prompt and the second will be used in the query.

```
SELECT DISTINCT PRODUCTS.PRODUCTNAME, PRODUCTS.PRODUCTID
FROM CATEGORIES, PRODUCTS
WHERE CATEGORIES.CATEGORYID = PRODUCTS.CATEGORYID
AND CATEGORIES.CATEGORYID = :category
```

Select Statement with Two Columns

Use custom selection choices:

Rather than having a drop-down menu with all the distinct column values, you can also create a custom list of parameter values. To create this list, select this option and click the *Setup Choices* button. This will launch a new dialog allowing you to create a list of choices.



○ Enter values as text

● Choose values from column

ORDERS.SHIPSTATE

AK
CA
FL
IL
NJ
NY

Selection Choices:
CA
FL
IL
NJ
NY

Data Type: String

Custom Parameter List Dialog

In this dialog, you can either enter custom values or select values from the distinct values of a column in the database. Once you finish specifying the values for the list, click the *OK* button and the choices will be saved.

Default Value:

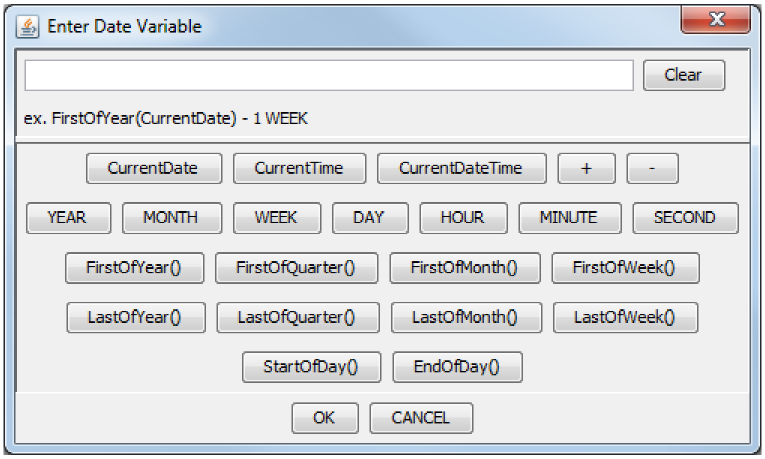
This allows you to specify a default value for the parameter. Although you don't have to specify a default value, it is recommended that you do so. If you do not supply a default value, you cannot open or manipulate the report template without the data source present.

You can either select a single value manually (either choose it from a list or type it manually, it depends on the mapping method you chose) or map the default value to a SQL query.

For multi-value parameters (see Section 1.3.2.2.2.1 - Multi-Value Parameters), the SQL query can return more than one value. In such case, several values will be chosen as default parameter values.

Date Variable:

This option is only available when the parameter is not mapped to a database column or function, or it's mapped to a SQL resultset and not set to a custom selection choice. This option is only intended for parameters with variable type date/time. When you click this button, the following panel will pop up, listing all the supported keywords.



ex. FirstOfYear(CurrentDate) - 1 WEEK

Enter Date Variable Dialog

This dialog allows you to select one of the three keywords: `CurrentDate`, `CurrentTime`, and `CurrentDateTime`. You can add or subtract units of time from the current date/time, allowing you to have a dynamic date range. For example, a report can have the following default values:

```
StartDate: CurrentDate - 1 WEEK
EndDate: CurrentDate
```

This would indicate that every time the report is run, the default prompt should be one week ago to the current date. Other supported time units are `YEAR, MONTH, DAY, HOUR, MINUTE`, and `SECOND`. This feature only supports a single addition or subtraction, it does not support multi-value parameters.

You can also use functions to define the parameter value:

FirstOfYear() Argument format: `CurrentDate`, `CurrentDateTime`, e.g. `FirstOfYear(CurrentDate)`

This function returns a date of the first day of the year from the argument. For example, when the argument evaluates to `2012-08-14`, the function returns `2012-01-01`.

LastOfYear()

Argument format: `CurrentDate`, `CurrentDateTime`, e.g. `LastOfYear(CurrentDate)`

This function returns a date of the last day of the year from the argument. For example, when the argument evaluates to `2012-08-14`, the function returns `2012-12-31`.

FirstOfQuarter()

Argument format: `CurrentDate`, `CurrentDateTime`, e.g. `FirstOfQuarter(CurrentDate)`

This function returns a date of the first day of the quarter which includes the date from the argument. For example, when the argument evaluates to `2012-08-14`, the function returns `2012-07-01`.

LastOfQuarter()

Argument format: `CurrentDate`, `CurrentDateTime`, e.g. `LastOfQuarter(CurrentDate)`

This function returns a date of the last day of the quarter which includes the date from the argument. For example, when the argument evaluates to `2012-08-14`, the function returns `2012-09-30`.

FirstOfMonth()

Argument format: `CurrentDate`, `CurrentDateTime`, e.g. `FirstOfMonth(CurrentDate)`

This function returns a date of the first day of the month from the argument. For example, when the argument evaluates to `2012-08-14`, the function returns `2012-08-01`.

LastOfMonth()

Argument format: `CurrentDate`, `CurrentDateTime`, e.g. `LastOfMonth(CurrentDate)`

This function returns a date of the last day of the month from the argument. For example, when the argument evaluates to `2012-08-14`, the function returns `2012-08-31`.

FirstOfWeek()

Argument format: `CurrentDate`, `CurrentDateTime`, e.g. `FirstOfWeek(CurrentDate)`

This function returns a date of the first day of the week which includes the date from the argument. For example, when the argument evaluates to `2012-08-14`, the function returns `2012-08-12` (Sunday is considered as the beginning of the week).

LastOfWeek()

Argument format: `CurrentDate`, `CurrentDateTime`, e.g. `LastOfWeek(CurrentDate)`

This function returns a date of the last day of the week which includes the date from the argument. For example, when the argument evaluates to `2012-08-14`, the function returns `2012-08-18` (Saturday is considered as the end of the week).

StartOfDay()

Argument format: `CurrentTime`, `CurrentDateTime`, e.g. `StartOfDay(CurrentDateTime)`

This function returns a time of the start of the day from the argument. For example, when the argument evaluates to `2012-08-14 12:15:03`, the function returns `2012-08-14 00:00:00.0`.

EndOfDay()

Argument format: `CurrentTime`, `CurrentDateTime`, e.g. `EndOfDay(CurrentDateTime)`

This function returns a time of the end of the day from the argument. For example, when the argument evaluates to `2012-08-14 12:15:03`, the function returns `2012-08-14 23:59:59:999`.

Data Type:

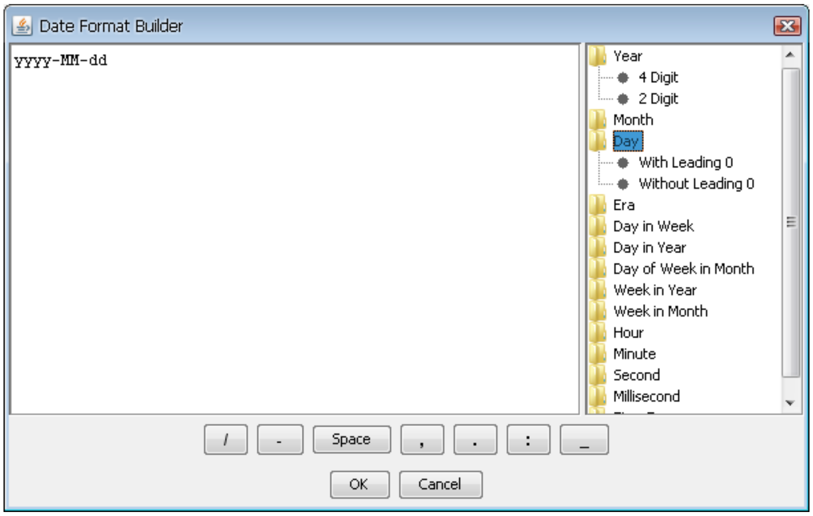
This allows you to specify data type for the parameter value(s). If you mapped the parameter to a column, the data type is set automatically.

Allow Select All Option:

Use this to add an option to the parameter prompt dialog that will allow users to select all parameter values even for single-value parameters. See the Section 1.3.2.2.2.4 - All Parameters for more details.

Custom Date Format:

This allows you to set the format in which the date parameter should be entered. This option is only available if you have not mapped the parameter to a column or entered custom selection choices (i.e. the end user will be typing in the date value). When you check this option, you can enter the date format in a combination of characters that represent time elements. You can build the format easily using the date format builder by clicking the *Build* button.



```
yyyy-MM-dd
```

Date/Time Format Builder

The builder contains a list of elements available on the right. You can mouse over the elements to see an example of each presentation. The bottom section contains a set of separators available for use.

For more information about characters and their formatting, please refer to the documentation for the `printDate()` function in Section 1.8.2.8.3 - Date/Time Functions. Formatting for this option is the same as for the format argument of the function.

Prompt Name: This allows you to specify the prompt that is given to the user in the parameter dialog.

If you map the parameter, the user will see either a drop down box (single value parameter) or a list box (multi-value parameter) containing various options. If you choose not to map the parameter, the user will see a textbox to enter their own value. In case of a multi-value parameter, it is recommended to let the user inform in the parameter prompt that this parameter accepts multiple values. Users can separate multiple values using a comma (e.g. `ARC, DOD, TRD`). If the text requires the use of comma, the user can use quotes to include the comma within the filter string (e.g. `"Doe, John", "Smith, Mike"`).

Clicking the *Previous Parameter* and the *Next Parameter* buttons allows you to initialize each of the parameters that have been defined in the query.

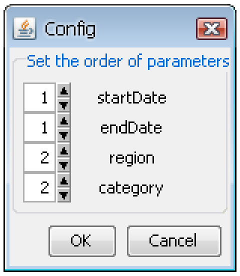
When you select to use a parameterized query to design a report, or open a report that uses a parameterized query, the report will load/start with the default values. You will be prompted to provide parameter values when you preview the report.

1.3.2.2.2.3. Cascading Parameters

By default, the user is prompted to enter all report parameters at once in the prompt dialog. This configuration, however, may not be the best approach if some parameters are mapped to database columns with a significant number of distinct values. It can be difficult to select from a very large list and depending on the parameter combination, users may be able to select parameters that do not return any data.

To assist with these problems, EspressReport provides a feature that allows the user to configure the order in which the parameters should be entered. With this feature enabled, the user enters parameters in the dialog in a predefined order. As such, each selection will be applied as a filter to the next parameter prompt(s). Using cascading parameters can limit the number of distinct values presented to the user and can prevent the user from selecting invalid parameter combinations.

To enable cascading parameters, check the option marked *Prompt parameter in sequence* in the parameter initialization dialog. Then click the *Config* button to set the order of the parameter prompts. A dialog will open showing all the parameters defined in the query.



Config

Set the order of parameters

1 startDate
1 endDate
2 region
2 category

Parameter Sequence Dialog

Using the spin boxes, you can set the sequence for the query parameters. User will be prompted to start with the lowest numbered parameter and work his/her way up to the highest one. If two or more parameters share the same number, the user will be prompted to enter those parameters at the same time (in the same dialog).

By default, the parameter values for the next level are generated by rerunning the entire query with the previously prompted parameters filled in. If the original query is slow to execute, you can improve performance by mapping higher order parameters to SQL Queries. You can even include previously selected parameter values in the mapped query. For more information, please see Section 1.3.2.2.2.2 - Initializing Query Parameters.

1.3.2.2.2.4. All Parameters

Sometimes you want to select all parameter values at once. The *All Parameters* feature allows you to do so.

Single-value parameters

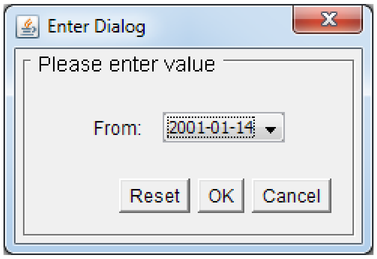
It is possible to select all parameter values at once, even for parameters that do not allow multi-value selection.

> **Note**
>
> There is a difference between multi-value selection and all-value selection. See the Inner Workings chapter to learn more.

For example: Let's assume you have a condition like this:

```
WHERE column = :Parameter
```

In such case, the parameter prompt dialog will not allow you to select more than one value.



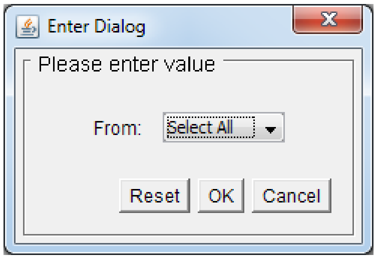
Enter Dialog

Please enter value

From: 2001-01-14

Typical single-value parameter

But you can use the *Select All Values* feature to add an option to the parameter value list that will allow viewers to select all parameter values at once (even if the parameter does not allow multi-value selection).



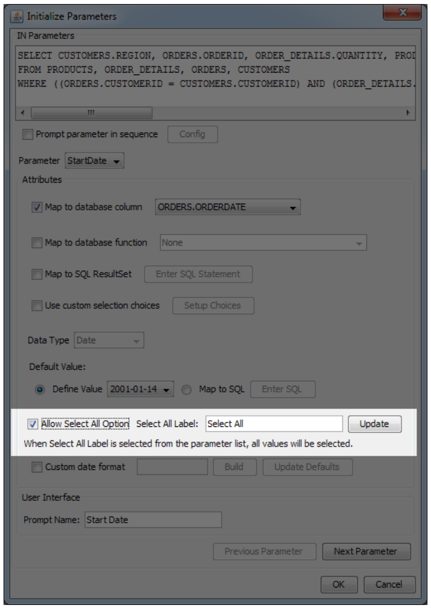
Enter Dialog

Please enter value

From: Select All

Single-value parameter with the Select All functions enabled

The *Select all* feature can be enabled in the *Initialize Parameters* dialog (see Section 1.3.2.2.2.2 - Initializing Query Parameters for more details) by selecting the *Allow Select All Option* checkbox.



Initialize Parameters

IN Parameters

```
SELECT CUSTOMERS.REGION, ORDERS.ORDERID, ORDER_DETAILS.QUANTITY, PROI
FROM PRODUCTS, ORDER_DETAILS, ORDERS, CUSTOMERS
WHERE ((ORDERS.CUSTOMERID = CUSTOMERS.CUSTOMERID) AND (ORDER_DETAILS.
```

□ Prompt parameter in sequence

Parameter: StartDate

Attributes

☑ Map to database column: ORDERS.ORDERDATE

□ Map to database function: None

□ Map to SQL ResultSet

□ Use custom selection choices

Data Type: Date

Default Value:

● Define Value: 2001-01-14 ○ Map to SQL

☑ Allow Select All Option Select All Label: Select All

When Select All Label is selected from the parameter list, all values will be selected.

□ Custom date format

User Interface

Prompt Name: Start Date

This option is only available for parameters that meet the following requirements:

1. The parameter uses one of the following operators. If there are multiple occurrences of this parameter in the query, all parameter comparison operators have to be one of these:

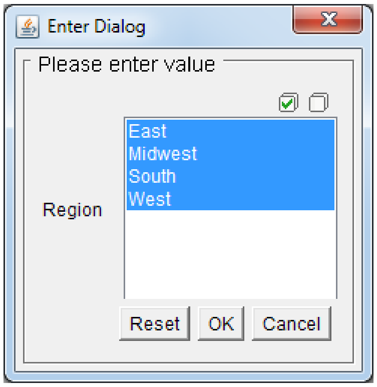
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
=	equal to

2. The parameter is mapped to the same column as the column from the parameter condition **or** the parameter isn't mapped to anything.

After you select the *Allow Select All* option, the *Select All Label* field activates, allowing you to enter a text that will be used for selecting all data from the query. For parameters that are mapped to a column, this text will be displayed in the parameter value list in 1st place. For parameters that aren't mapped to anything, entering this text as the parameter value will result in selecting all data.

Multi-value parameters

Unlike single-value parameters, the *Select All* feature is enabled for all multi-value parameters by default (in fact, it can't be disabled, because disabling it for multi-value parameters would make no sense). All you have to do to use this feature is to click on the ☑ *Select All* icon in the parameter prompt.



However, multi-value parameters can work in two modes:

1. If the parameter meets the conditions from the previous paragraph and the parameter is on the first cascading level (i.e. parameter cascading is disabled or the parameter is on the first cascading level), it is parsed by the SQL parser and the parameter condition is nullified. Nullifying the parameter optimizes the query and prevents it from causing performace issues or even errors. See the Inner Workings to learn more about how it works.

2. If the parameter doesn't meet the conditions from the previous paragraph or if it's not on the first cascading level, selected values will be injected to the query as a list of values separated by comma. If there is a large amount of values injected to the query as a list, the query can become quite long. Long queries can cause performance issues or even errors, so it is **not recommended** to use this option for parameters with many values.

Inner Workings

If a report viewer chooses to select all parameter values for a single-value parameter or for a multi-value parameter that meets the conditions for parameter disabling, the query is then automatically parsed and a special condition is added to the parameter which basically disables the parameter.

For example: The following query

```
select *
from table
where column > parameter_value
```

Would be parsed and passed to the database as:

```
select *
from table
where ((column > parameter_value) OR (1 = 1))
```

This example also demonstrates another important thing: selecting all values for the < (less than) or > (greater than) operators returns all values from the table (if there are no other conditions) rather than returning no data at all (because condition like `WHERE <all data from the Date column> > Date` would return no data...).

Because EspressReport allows you to use many database systems, parsing may fail for certain complex queries in certain databases. In such case a warning dialog will be displayed.

In such situations, you have the following three options:

1. Try to modify the query so it can be parsed by our parser.

2. Add your own *Select all parameters* condition to the query.
   For example:

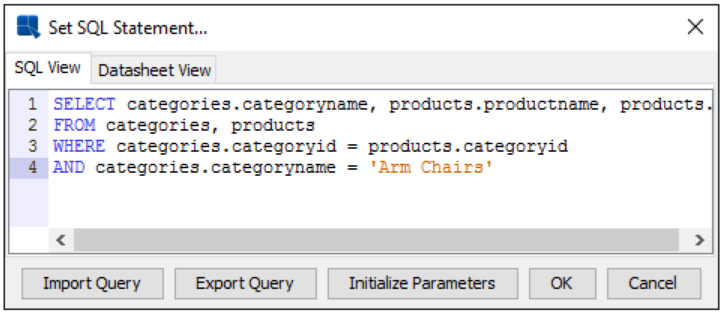
   ```
   WHERE ((column = :Parameter) OR (:Parameter
    LIKE 'selectall'))
   ```

> **Note**
>
> If you embed all parameters directly to the query, leave the *Allow Select All Option* option **disabled**.

3. Contact Quadbase support [https://www.quadbase.com/support-overview/].

1.3.2.2.3. Entering SQL Statements

Typically, the Query Builder is recommended for creating queries. However, there are cases when it is necessary to enter SQL statements directly: for example if the query is already created in a QRY file, if the query is built into a stored procedure/function, or if the query requires commands not supported by the Query Builder. In these situations, select *Enter SQL statement* to open the Set SQL Statement window. Here you can enter SQL statements directly into the text area as shown below or you can load an existing QRY File.



```
SELECT categories.categoryname, products.productname, products.
FROM categories, products
WHERE categories.categoryid = products.categoryid
AND categories.categoryname = 'Arm Chairs'
```

Enter SQL Statement Dialog

To preview the result set, click on the *Datasheet View* tab.

1.3.2.2.3.1. Calling Oracle Stored Procedures

Compared to other database systems, Oracle uses a different approach when it comes to stored procedures and functions. For example, on MS SQL Server, using the `EXEC` command will return a result set. However, Oracle requires the use of an `OUT` parameter with a `REF CURSOR` type to return the result set. In addition, Oracle will not accept multiple statements from a single query. Therefore, it is necessary to store the query within a stored function and use special syntax to access the existing Oracle stored procedures.

To access your Oracle stored procedures, the first step is to define a weakly typed `REF CURSOR` using the following PL/SQL statement:

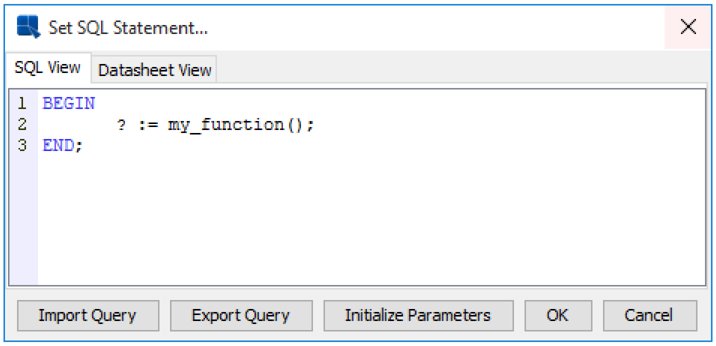
```
CREATE OR REPLACE PACKAGE types
```

```
AS

      TYPE ref_cursor IS REF CURSOR;

END;
```

This `ref_cursor` type will be used to store the query result set and return as an `OUT` parameter. The next step is to create a function which calls your stored procedure and executes your query. The following skeleton code will return a simple query using the `ref_cursor` type.

```
CREATE OR REPLACE FUNCTION my_function()

      RETURN types.ref_cursor

AS

      result_cursor types.ref_cursor;

BEGIN

      do_stored_procedure();
      OPEN result_cursor FOR
            SELECT * FROM Categories

      RETURN result_cursor;

END;
```

Now that the Oracle stored function is set up, it can be easily called from ReportDesigner using a special PL/SQL like syntax. In the Set SQL Statement window, enter the following syntax to call the Oracle stored function:



```
BEGIN
        ? := my_function();
END;
```

Calling simple Oracle stored function

The `BEGIN ... END;` syntax alerts the system that the user is trying to access an Oracle stored function. And the “?” notifies the ReportDesigner that a variable is reserved for the `OUT` parameter. The JDBC syntax for calling Oracle stored procedures is as follows:

```
( call ? := my_function() )
```

However, EspressReport does not support this format. Preview the results by clicking the *Datasheet View* tab.

Here is a more practical example to illustrate how stored procedures can be used with EspressReport to develop useful solutions. Suppose you have a table called *employee_table* that stores an organization's location hierarchy such as the one shown below:

ID	NAME	PARENT	EMPLOYEE
1	All	NULL	0
2	America	1	0
3	Europe	1	0
4	New York	2	20
5	Santa Clara	2	30
6	Dallas	2	12
7	London	3	14
8	Paris	3	11

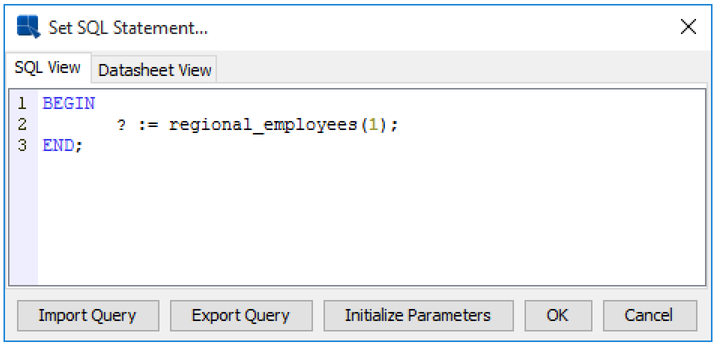
The table lists the various corporate locations in a tree structure. The numbers of employees are stored in the leaf nodes (e.g. New York, London, etc.) and each node contains information about its immediate parent. Suppose you want to create a report that displays the number of employees in a certain region and information about the separate branches within that region. For example, if the user inputs `ID = 2` (America), you want the report to display the total number of employees in America along with the branch locations. Using Oracle's `CONNECT BY` and `START WITH` clauses, the problem can be solved with two simple Oracle Stored Functions:

```
CREATE OR REPLACE FUNCTION sum_employees(locID IN NUMBER)

      RETURN NUMBER

AS

      sum_emp NUMBER;

BEGIN

      SELECT sum(employee) INTO sum_emp
      FROM employee_table
      CONNECT BY PRIOR id = parent
      START WITH id = locID;

      RETURN sum_emp;

END;

CREATE OR REPLACE FUNCTION regional_employees (locID IN NUMBER)

      RETURN types.ref_cursor

AS

      result_cursor types.ref_cursor;

BEGIN

      OPEN result_cursor FOR
            SELECT id, name, sumEmployees(id) AS Employees
            FROM employee_table
            CONNECT BY prior id = parent
            START WITH id = locID;

      RETURN result_cursor;
```

```
END;
```

The function `sum_employees` takes the starting node as an argument and finds the sum of all leaf nodes that are descendents of that node. For example, `sum_employees(3)` returns 25 because there are 25 employees in Europe (14 in London, 11 in Paris). The second function, `regional_employees`, traverses through the tree structure starting with the `locID` and builds a result set from the `ID`, `Name` and the result from the `sum_employees` function. The result set is then returned through a `REF CURSOR`.

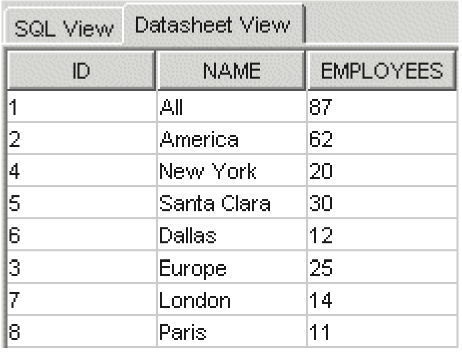
To call a stored function that requires an argument, enter the following statements in the Set SQL Statement window:



```
BEGIN
        ? := regional_employees(1);
END;
```

Calling regional_employees function

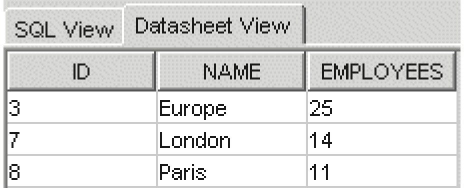
Preview the results by clicking the *Datasheet View tab*.



ID	NAME	EMPLOYEES
1	All	87
2	America	62
4	New York	20
5	Santa Clara	30
6	Dallas	12
3	Europe	25
7	London	14
8	Paris	11

Result set from regional_employees

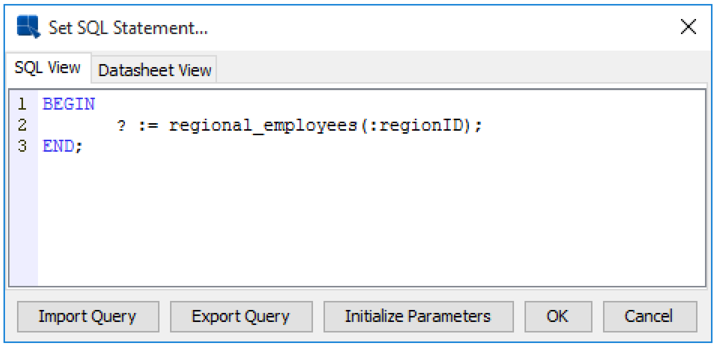
As seen in the results, the `CONNECT BY` clause traverses the tree recursively listing the American nodes together before listing the European nodes. If the user is only interested in European locations, they can enter **`3`** for the parameter and the following result set would return this:



ID	NAME	EMPLOYEES
3	Europe	25
7	London	14
8	Paris	11

Result set from regional_employees in Europe

To create a parameterized report, use the `:param_name` syntax. The SQL parser in EspressReport will be able to differentiate between the colon used for parameters and the one used for the assignment operator (`:=`). Here is an example of using the parameters:



```
BEGIN
        ? := regional_employees(:regionID);
END;
```

Calling Oracle Stored Function using Parameter

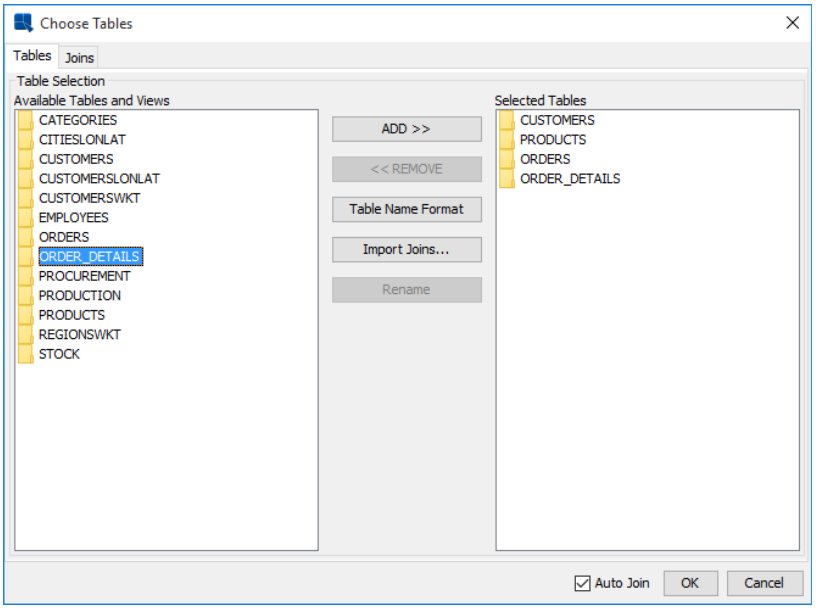
When using `IN` parameters, it is necessary to initialize the parameters prior to executing the query. It is especially important to set the correct default data type for executing stored procedures, because the parameters cannot be mapped to existing columns. You can find more information about initializing parameters in Section 1.3.2.2.2 - Parameterized Queries. To try this example, `<EspressReportInstall>\help\examples\data\locationHierarchyExample.sql` contains SQL commands to create `employee_table` as well as two stored functions.

1.3.2.3. Data Views

In addition to the query interfaces, EspressReport provides another way of retrieving database data - data views. Data views provide a simplified view of the database in which users can design queries by simply selecting fields without using the Query Builder or having any knowledge of the underlying database structure. Using data views, administrators can predefine tables, joins, and fields, creating a local schema for the user to select from.

For example, an administrator could set up a data view for the sales department. The appropriate database tables and fields are pre-selected and grouped in a manner congruent with business users' logic. For example a group called “invoices” would have the appropriate customer and order fields. End users would then select this data view, pick the pertinent fields, specify a date range, and then begin designing a report.

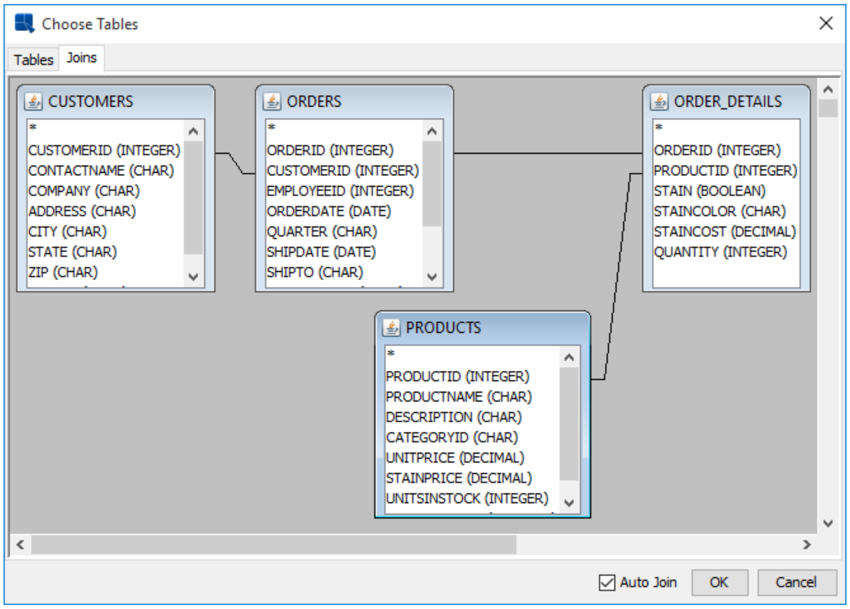
To create a data view, select the *Data Views* node in the Data Source Manager window and click the *Add* button. A new window will open allowing you to select the database tables you want to use for the data view.



Data View Choose Tables Dialog

Left window contains all available database tables and views. You can add a table by selecting it in the left window and clicking the *ADD>>* button. By default, the data view will use the name format you specified when setting up the database connection. You can change the naming by clicking the *Table Name Format* button, or specify a table alias by clicking the *Rename* button. You can also import selected tables and joins from another data view by clicking the *Import Joins…* button.

The *Joins* tab of this window allows you to specify the joins between the selected tables.

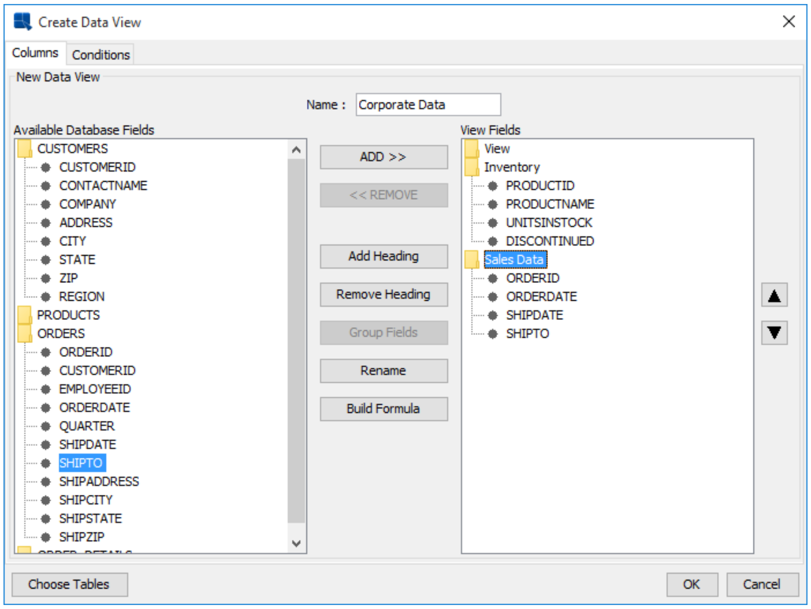


☑ Auto Join

Data View Joins Dialog

The *Joins* tab shows all selected tables and their associated fields. The tables will be auto-joined depending on which option you selected when setting up the database connection. These auto-joins create a standard join between tables represented by a line. To remove a join or edit join properties, right click on the line and select your choice from the pop-up menu. To add a join, click and drag one column field to another in a different table. A join will then appear. Data views use the same join properties as the Query Builder. For more information about join properties, please see Section 1.3.2.2.1.2 - Joins.

After you finish selecting and joining tables, click the *OK* button and a new window will open allowing you to construct the data view.



Name : Corporate Data

Create Data View Dialog

The left window contains a list of tables you have selected and their associated fields. Each folder represents a table and can be opened and closed by double clicking. The right window contains fields that have been selected for the data view. To add a field to a data view, select it in the left window and click the *ADD>>* button. Fields can be removed from the data view in the same way by selecting a field in the right window and clicking the *<<REMOVE* button. You can create a calculated column by clicking the *Build Formula* button. This will open the formula builder, allowing you to build the column. You can also define an alias by selecting any of the view fields in the right window and clicking the *Rename* button.

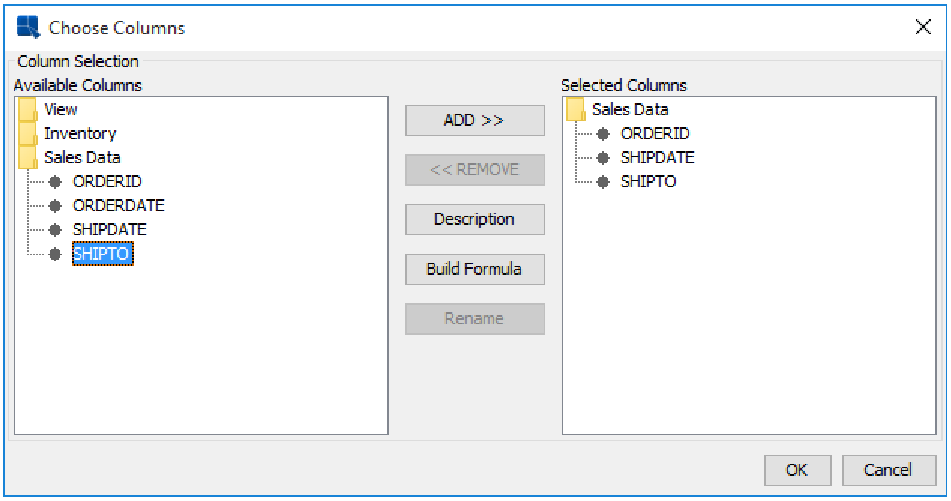
You can also group fields within the data view by adding headings. This allows you to create your own organizational structure of *virtual tables* that group data from different database tables under one heading. To create a heading, click the *Add Heading* button. You will then be prompted to specify a name for the heading. The new heading will then appear as a folder in the right window. To add fields under a heading, first select the fields you want to add from the right window and click the *Group Fields* button. You will then be presented with a drop-down menu, allowing you to select the heading under which you would like to add the fields.

The *Conditions* tab contains a formula builder window that allows you to specify certain filtering criteria for end users. Anything added in this window will be added to the *Where* clause of the generated SQL. For more information about using the formula builder, please see Section 1.3.2.2.1.3 - Columns.

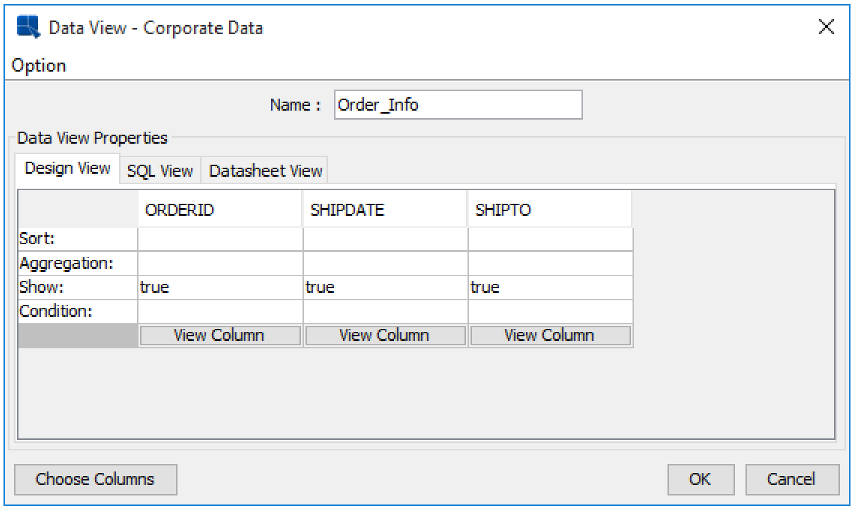
Once you finish creating the data view, click the *OK* button and the data view will be added to the Data Source Manager. Users can now use this view to construct ad-hoc queries.

When you design a report using a data view as the data source (by selecting the data view and clicking the *Next* button), a window will open allowing you to select which fields in the view you want to use for the report. From this dialog, you can also build computed fields based on the available view columns.

After you select the fields, click the *OK* button and a new window will open allowing you to specify sorting, aggregation, and filtering conditions for the data view.



Data View Choose Fields Dialog



Name : Order_Info

	ORDERID	SHIPDATE	SHIPTO
Sort:			
Aggregation:			
Show:	true	true	true
Condition:			
	View Column	View Column	View Column

Data View Conditions Window

You can specify sorting, aggregation, and conditions for every field in the data view by double clicking on the respective field. Sorting and aggregation can be selected from drop-down menus. Double clicking on the *Conditions* field brings up a new window that allows you to specify simple selection criteria like `>, <, =,` and `between`. Users can build more advanced filtering criteria by right clicking on the *Conditions* field and selecting *Build* from the pop-up menu. This will open the Formula Builder window allowing you to build a condition. You can also display all of the unique values in the column by double clicking on the *View Column* button.

The Option menu in the upper left corner of the conditions window allows you to select a vertical/horizontal view for the conditions window, initialize any parameters in the data view, or save the query.

The selection set and conditions that you specify will be saved as a data view query with the name that you specify in the name field. Data view queries are saved under the node for the data view. A report created from the data view will refer to the data view query for updating/modification.

1.3.2.3.1. Data View Parameters

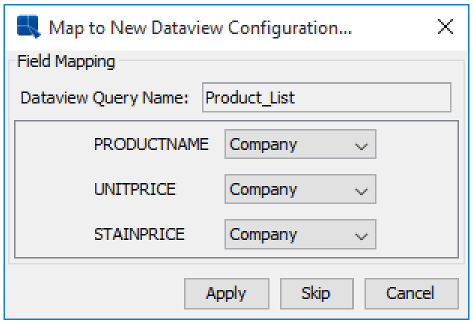
As with Query Builder, users can specify query parameters in Data Views. To add a parameter to a data view, select a data view in the Data Source Manager and click *View* to run the data view. After you select fields for the data view and you are in the conditions window, right-click in the *Condition* field for a column and select *Build* from the pop-up menu. This will bring up the formula builder, allowing you to specify a parameter in the same way as in Query Builder. For more information about this, please see Section 1.3.2.2.2 - Parameterized Queries.

Once you enter the parameter, you will be prompted to initialize it if you go to the *Datasheet View* tab and then click *Ok* to continue with the report wizard, or if you save the selections as a query. You can also initialize the parameter by selecting *Initialize Parameters* from the Option menu.

1.3.2.3.2. Updating Data View Queries

Sometimes you may need to make changes to the structure/make-up of the data view as your data model or requirements change. Changes could include adding/removing fields or re-naming them. You can propagate changes from the data view to its associated queries by selecting it in the data source manager and selecting *Data View Queries* from the Update menu.

All of the queries associated with the view will be scanned and any inconsistencies in fields or field names will be presented for you to update.



Map to New Dataview Configuration...

Field Mapping

Dataview Query Name: Product_List

PRODUCTNAME	Company
UNITPRICE	Company
STAINPRICE	Company

Update Query Fields Dialog

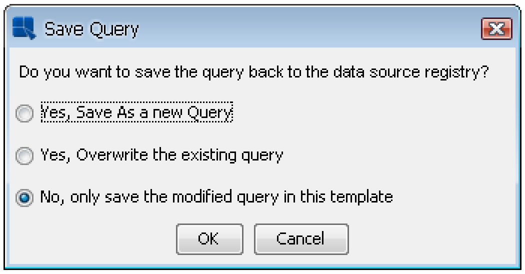
For each query, you will be prompted to change any fields that no longer match the data view structure. For each field, you can select a field from the data view to map it to, or remove the field from the query. If you do not want to change anything in the query, you can click the *Skip* button. The query will continue to run, but it will refer to the old data view structure. Click the *Apply* button to save the changes to the data view query.

1.3.2.4. Editing Queries

If you have selected to build a report using database data either by designing a query in the Query Builder, writing an SQL statement, or running a data view, you can modify the query directly from the ReportDesigner without having to go back to the Data Source Manager.

To modify a report's query, select *Modify Query* from the Data menu in Report Designer or click the *Modify Query* button on the toolbar. If you have designed a query in the Query Builder, then the Query Builder interface will re-open allowing you to modify the query. If you have entered an SQL statement, a textbox will open allowing you to modify the SQL. If you have used a data view, the data view conditions window will re-open allowing you to change the filters or pick additional fields.

Once you specify all the changes, you will be given an option to modify the query in the data registry, save a new query in the data registry, or modify only the query in the template.



Save Query

Do you want to save the query back to the data source registry?

○ Yes, Save As a new Query

○ Yes, Overwrite the existing query

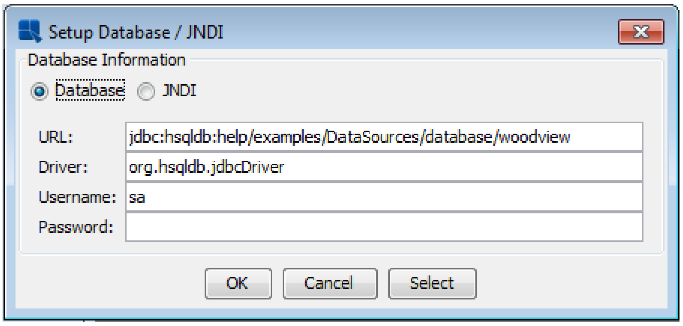
● No, only save the modified query in this template

Saving Query Options

Once you specify the save options, the modified query will be applied to the report. Note that if you made significant changes to the query, you may need to perform the data mapping for the report again. For more information about data mapping, please see Section 1.4 - Report Types and Data Mapping.

1.3.2.5. Editing Database Connections

If you have selected to build a report using database data, you can also directly edit a template's database connection information from within the report Designer. To modify the connection information in the report, select Data → Modify Database. This will bring up a dialog allowing you to specify a different URL, Driver, Username, and/or password for the report to use when connecting to the database.

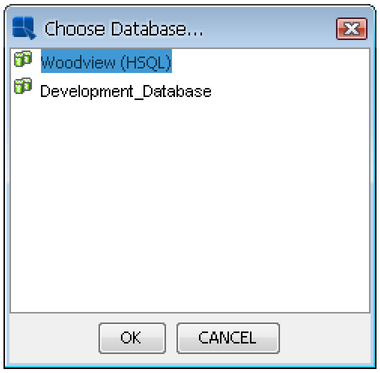


● Database ○ JNDI

URL: jdbc:hsqldb:help/examples/DataSources/database/woodview
Driver: org.hsqldb.jdbcDriver
Username: sa
Password:

Change Database Connection Dialog

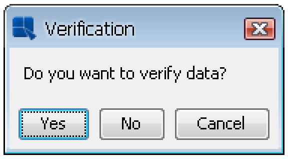
In addition to manually entering the database information, you can retrieve the database connection information from a data registry. To do this, click the *Select* button on the Database Connection dialog. This will allow you to browse to XML registry file from which you want to pull the database connection. When you select a registry file, you will be presented with a list of databases defined in the registry.



Woodview (HSQL)
Development_Database

Select Database from Registry

Select the database that you want to use and click the *Ok* button. The connection information for that database will be automatically applied to the connection dialog. After you set the connection information for the template, click the *Ok* button to apply the changes. A dialog will open asking you if you would like to verify the new connection information.



Do you want to verify data?

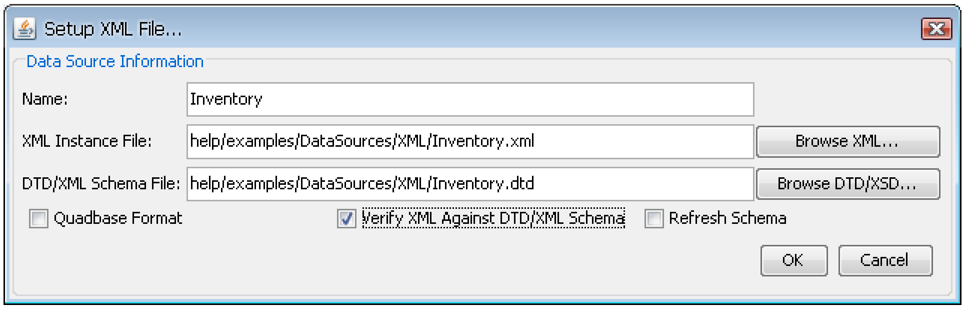
Verify Connection Dialog

Unless you know that data source isn't present, it's generally a good idea to check that the supplied connection information is correct. If you're changing the database connection information in a report, all the sub-reports, charts, and drill-down layers in the report that use the same connection information will be automatically updated as well.

Unlike the modify query feature, changes to a template's database connection will only saved be to the template. It will not be saved to the data registry.

1.3.3. Data from XML and XBRL Files

In addition to relational databases, EspressReport allows you to retrieve data and query XML files. XML data can be in virtually any format, but you need to specify a DTD file or an XML Schema (XSD) along with the XML data. To set up an XML data source, select *XMLFiles* node in the Data Source Manager and click the *Add* button. A dialog will open prompting you to specify options for the new XML source.



Name: Inventory

XML Instance File: help/examples/DataSources/XML/Inventory.xml

DTD/XML Schema File: help/examples/DataSources/XML/Inventory.dtd

□ Quadbase Format ☑ Verify XML Against DTD/XML Schema □ Refresh Schema

Setup XML Data Source Dialog

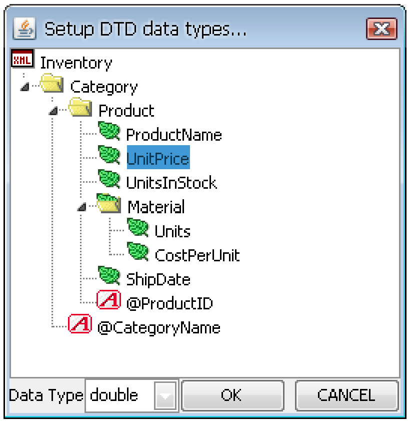
The first option allows you to specify a display name for the XML data source. The second option allows you to specify the location of the XML file from which you want to retrieve data. Note that you can also specify an XBRL file in this field. You can set up a data source that retrieves XML data from an HTTP server here as well, by adding the appropriate URL as the file location. The third option allows you to specify the location of a valid DTD or XML schema file for the XML file.

The *Quadbase Format* checkbox allows you to indicate whether the XML file is in the form of an XML export from EspressReport. For example, if you select to export a report's data in XML format, you can read it back in using this format. For more on exporting to XML, see Section 1.7.2 - Exporting Reports. When you use a file in this format, you do not have to specify a DTD or XML Schema.

The *Verify XML against DTD/XML Schema* checkbox will make sure that the supplied XML file/source complies with the layout specified in the DTD or XML schema file. Because queries are designed based on the structure of the DTD/XML Schema file, a non-conforming XML source could produce unexpected results. If the XML does not conform to the DTD or XML schema, you will be given a warning. You can, however, continue setting up the data source.

The *Refresh Schema* checkbox will reload the schema or DTD definition to incorporate any changes to the structure in the XML data source in the registry. This option is only necessary if the DTD or schema definition has changed since the data source was first created.

Once you finish setting up the XML file and the DTD/XML Schema, a new dialog will open allowing you to specify the data type for all the selectable elements in the XML data source. The dialog will be different depending on whether you are using a DTD file or XML Schema.



Data Type: double

DTD Data Type Selection Dialog

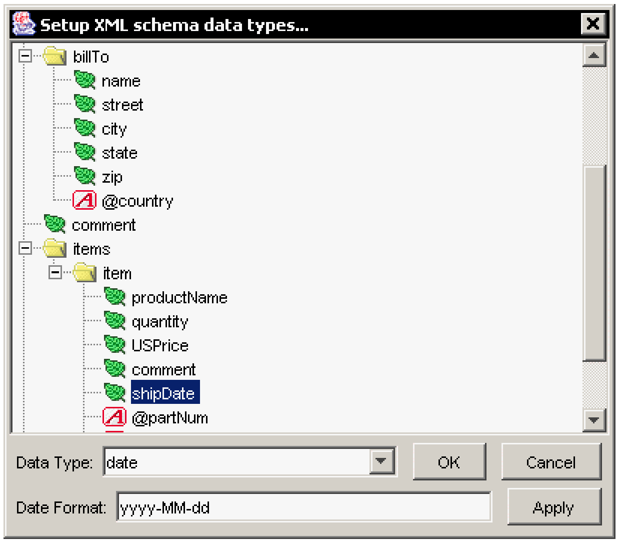
Because DTD files do not specify a correct data type for elements, all elements are considered to be Strings by default. To change the data type of an element, you have to select the element and pick a data type from the drop-down window at the bottom of the dialog. To ensure proper results when you query the XML file, you should set the data type for all selectable elements. This includes leaf nodes, parent nodes that contain data, and attribute elements. The following data types are supported:

- String

- Integer
- Double
- Date (If you specify date as the data type, you will also be required to specify the date format.)
- Boolean

Once you finish specifying the data types, click the *OK* button and the XML source will be added to the Data Source Manager.

If you're using an XML schema, a different data types dialog will open.



Data Type: date

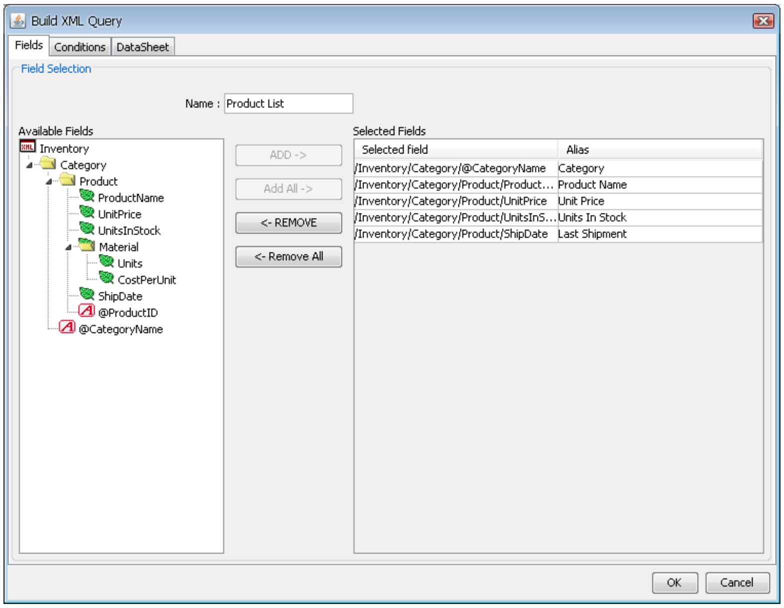
Date Format: yyyy-MM-dd

XML Schema Data Type Selection Dialog

Generally, the data types should already be defined in the XML schema file, but you can make any changes in this dialog. Once you finish specifying the data types, click the *OK* button and the XML source will be added to the Data Source Manager.

1.3.3.1. XMLQueries

Once you set up an XML data source, you can then create queries to select nodes, specify filtering conditions, and transform the tree structure into the tabular form used by EspressReport. To add a query, select the node for your XML source and click the *Add* button. This will launch the XML query builder interface that allows you to construct a query.



Name : Product List

Selected Fields

Selected field	Alias
/Inventory/Category/@CategoryName	Category
/Inventory/Category/Product/Product...	Product Name
/Inventory/Category/Product/UnitPrice	Unit Price
/Inventory/Category/Product/UnitsInS...	Units In Stock
/Inventory/Category/Product/ShipDate	Last Shipment

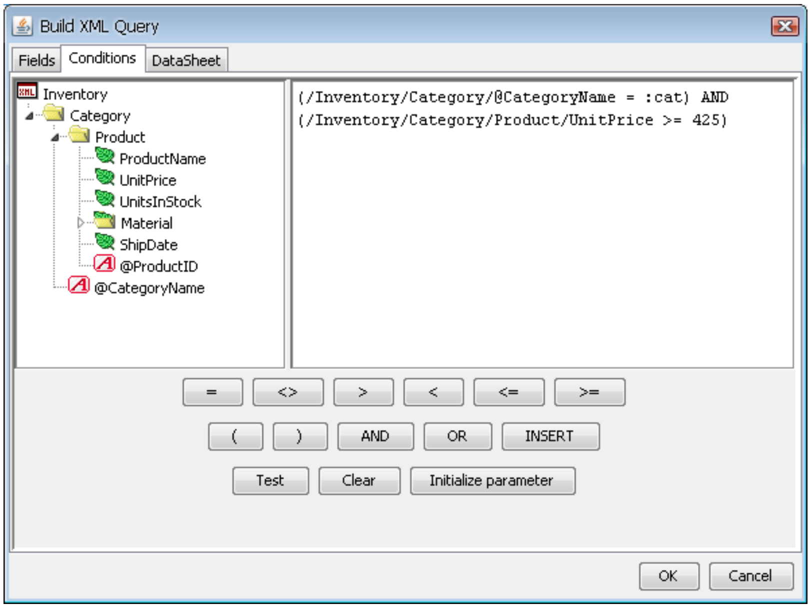
XML Query Field Selection Tab

The first tab in the XML query builder allows you to select the fields/nodes from the XML file that you would like to use in the report. The left side of the window contains the tree structure from the DTD or XML schema file. You can pick any selectable elements and add them to the query by clicking the *Add* button. Selected fields will appear in the right side of the window. You can specify an alias for any field by double clicking the *Alias* field for a column and typing the new column alias.

Note

Each selected element will become a column in the report. For results where a one-to-one relationship cannot be determined, the tabular structure is built using all available permutations in the data (similar to a cross-join in SQL). For best results, it is recommended that you select fields for a query where a clear hierarchical relationship is present.

The *Conditions* tab of the XML query builder allows you to specify some basic filtering criteria for your selection.



```
(/Inventory/Category/@CategoryName = :cat) AND
(/Inventory/Category/Product/UnitPrice >= 425)
```

XML Query Conditions Tab

You can specify an equal, not equal, greater than, less than, less or equal to, or greater or equal to condition for any selectable element in the XML file. You can also use the AND and OR operators to build compound conditions. Fields are specified using a direct path down the XML tree. Currently, only direct path is supported. You cannot use more complex XPath expressions. To add a field, you can double click on it in the left side, or you can select it and click the *Insert* button.

After you finish writing the conditions, click the *Test* button to verify that the syntax is correct.

The *DataSheet* tab allows you to preview the query result and see how the XML data is converted to tabular form. You can navigate through the result set the same way as in Query Builder (Section 1.3.2.2.1.6 - Query Output).

Once you select the fields and specify the appropriate conditions, click the *OK* button. The query will then be added as a new node under your XML source in the Data Source Manager and can now be used to create a report.

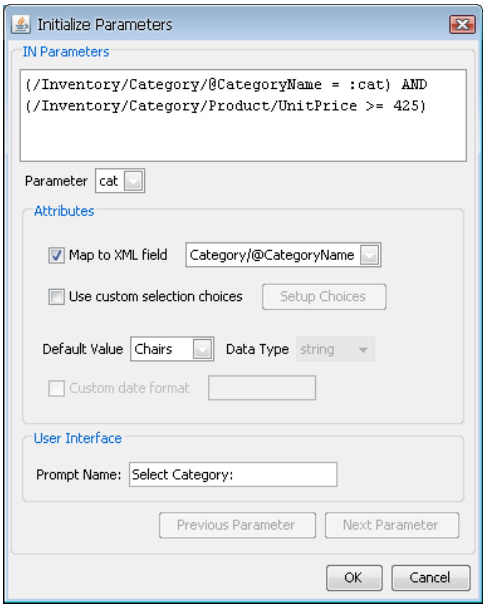
There is a sample XML file and a sample DTD/XSD included in the EspressReport installation. The files are located in `help/examples/DataSources/XML` directory of your installation and are called `Inventory.xml` and `Inventory.dtd`. There is also a sample servlet that allows you to stream XML data to the ReportDesigner. The servlet code and instructions are located in `help/examples/DataSources/XML/servlet` directory.

1.3.3.1.1. XML Parameters

As with database queries, you can also specify parameters for XML queries. The same syntax “:” is used to denote a parameter in the XML condition as it is in a query condition. So the following XML condition:

```
/Inventory/Category/@CategoryName = :category
```

would place a dynamic filter on the query for the *CategoryName* attribute. XML parameters are initialized in the same way as query parameters. The initialization dialog will appear if you try to preview or close the query, or you can trigger it by clicking the *Initialize Parameters* button. The only difference is that instead of mapping to a database field, the parameter prompt can be mapped to a node in the XML file.



```
(/Inventory/Category/@CategoryName = :cat) AND
(/Inventory/Category/Product/UnitPrice >= 425)
```

Parameter: cat

☑ Map to XML field: Category/@CategoryName

☐ Use custom selection choices

Default Value: Chairs

Data Type: string

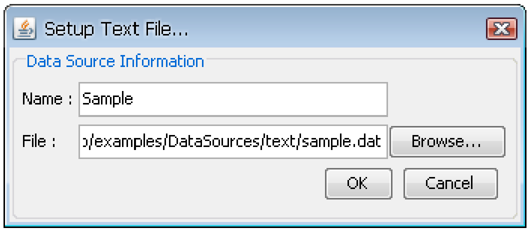
☐ Custom date format

Prompt Name: Select Category:

XML Parameter Initialization Dialog

1.3.4. Data from Text Files

EspressReport also allows you to retrieve data from flat text files. To add a text file as a data source, select the *TXTFiles* node in the Data Source Manager and click the *Add* button. A dialog will open prompting you to specify a display name and the location of the text file you want to use.



Name : Sample

File : ;/examples/DataSources/text/sample.dat

Add Text Data Source Dialog

Tip

The text file can be also retrieved from URL. To do so, enter the URL to the *File* text field. You have to enter a full URL with protocol etc. (e.g. `http://www.quadbase.com/textfile.txt`).

After you specify the information, click the *OK* button and the text source will appear under the *TXTFiles* node of the Data Source Manager window.

1.3.4.1. Formatting Requirements for Text Files

There are certain formatting requirements for the data within a text file in order make it readable by EspressReport. Generally, data is expected to be in a form similar to the following:

```
String,date,decimal
Name,day,Volume
"John","1997-10-3",32.3
"John","1997-4-3",20.2
"Mary","1997-9-3",10.2
"Mary","1997-10-04",18.6
```

The above data file is a plain text file. The first row specifies the data types and the second row specifies the field names. The third row and so on are the records. Every text file must consist of these three parts. There are four records, with three fields each in the example data file. The delimiter between the fields may be one of the following characters: `","`, `";"`, or `" "` (that is comma, semi-colon, or space). Each field can be put in quotes (single or double).

1.3.4.2. Data Types and Format for Text Files

In text data files, the data type is specified using a keyword. The following is a list of recognized keywords and their corresponding JDBC type and Java type.

Data File Keywords (Not Case Sensitive)	**JDBC Type**	**Java Type in EspressReport**
Boolean, logical, bit	BIT	Boolean
tinyint	TINYINT	byte
smallint, short	SMALLINT	short
int, integer	INTEGER	int
long, bigint	BIGINT	long
float	FLOAT	double
real	REAL	float
double	DOUBLE	double
numeric	NUMERIC	java.math.BigDecimal
decimal	DECIMAL	java.math.BigDecimal
date	DATE	java.sql.Date
time	TIME	java.sql.Time
timestamp	TIMESTAMP	java.sql.Timestamp
string	CHAR	String
varchar	VARCHAR	String
longvarchar	LONGVARCHAR	String

For certain data types, the data in a text file must be presented in a specific format. The following is a list of the data types that require specific formatting:

Data Type	**Format**	**Example**
Date	yyyy-mm-dd or yyyy-mm	2001-06-12 or 2000-06
Time	hh:mm:ss	12:17:34
Timestamp	yyyy-mm-dd hh:mm:ss	2001-06-12 12:17:34
Boolean	true/false, t/f, 1/0 (case insensitive)	true

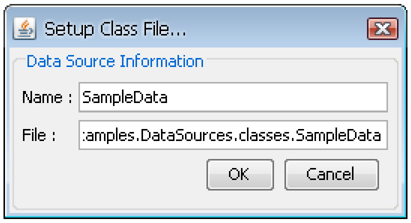
There is a sample text file included in the EspressReport installation. The file is located in `help/examples/DataSources/text` directory of your installation and is called `Sample.dat`.

1.3.5. Data from Class Files

For maximum flexibility, EspressReport allows you to design reports using object or array data by providing an interface to pass data to the Report Designer as an argument. Using the API, you can implement `IDataSource` to return an `IResultSet` object similar to the `java.sql.ResultSet` interface used for JDBC result sets. Users can provide their own implementation of `IResultSet` or use one provided by EspressReport.

For more information about this feature, please see Appendix 2.B.5 - Data passed in a Custom Implementation.

To add a class file as a data source, select the *ClassFiles* node in the Data Source Manager and click the *Add* button. A dialog will open prompting you to specify a display name and the location of the class file you want to use.



Setup Class File...

Data Source Information

Name : SampleData

File : :amples.DataSources.classes.SampleData

Add Class File Dialog

There is sample code available in `help/examples/DataSources/classes` directory of the installation. The complied code will generate a class file that passes the data array into ReportDesigner. Note that in order to use a class file as a data source, you must have the file or directory containing the package in your classpath (or the `-classpath` argument of `EspressManager.bat` or `EspressManager.sh` file when using ReportDesigner).

1.3.5.1. Parameterized Class Files

EspressReport provides an additional API interface, `IParameterizedDataSource`, that allows you to define report parameters within the context of a class file data source.

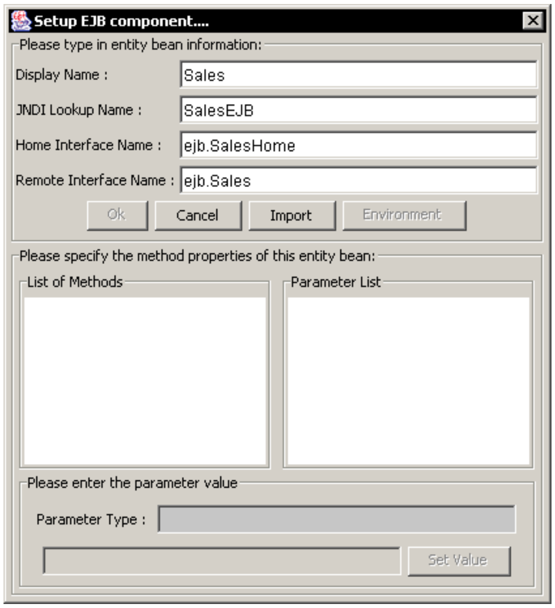
Parameters that are defined within the context of a class file work in the same way as query parameters. For more information about setting up a parameterized class file data source, please see Appendix 2.B.5 - Data passed in a Custom Implementation.

1.3.6. Data from EJBs

Using Enterprise JavaBeans™ technology, developers can simplify the development of large enterprise applications. With EJB technologies, developers can rely on building business logic and allow the application server (EJB container) to manage all system level services.

When working in the Java EE™ environment, persistent data interfaces are provided by entity beans. Although the underlying storage mechanism might be a relational database, the application data model is the EJB and it may not be desirable to have a reporting tool making redundant database connections. For this situation, EspressReport allows users to query data directly from an entity bean.

To add an EJB as a data source, the EJB must first be deployed in the application server and the client JAR file containing the appropriate stub classes must be added to your classpath (or the `-classpath` argument of the `EspressManager.bat` or `EspressManager.sh` file when using Report Designer). Select the *EJBs* node in the Data Source Manager and click the *Add* button. This will bring up a dialog allowing you to specify a display name and the connection information for the bean.

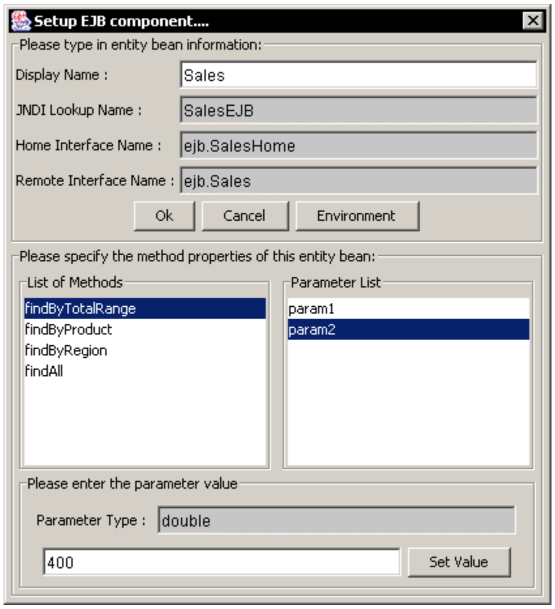

Setup EJB component....
Please type in entity bean information:
Ok
Cancel
Import
Environment
Please specify the method properties of this entity bean:
List of Methods
Parameter List
Please enter the parameter value
Set Value

Display Name : Sales
JNDI Lookup Name : SalesEJB
Home Interface Name : ejb.SalesHome
Remote Interface Name : ejb.Sales
Parameter Type :

Add EJB Dialog

To connect to an EJB data source, you must provide the JNDI lookup name for the bean (this is specified when you deploy the bean). For EJB 1.1 users, specify the name for the home interface and the remote interface. For EJB 2.0 users, specify the local home interface and the local interface. Once you specify all the information, click the *Import* button, which will analyze the home interface and retrieve all of the finder methods. Any methods found will be populated in the *List of Methods* section in the dialog.

The same dialog can be used to filter the data being retrieved based on parameters that are present in the finder methods. When you select a method in the left dialog, any parameters present will appear in the *Parameter List* section. You can then click on a parameter to set its value.


Setup EJB component....
Please type in entity bean information:
Ok
Cancel
Environment
Please specify the method properties of this entity bean:
List of Methods
Parameter List
Please enter the parameter value
Set Value

Display Name : Sales
JNDI Lookup Name : SalesEJB
Home Interface Name : ejb.SalesHome
Remote Interface Name : ejb.Sales

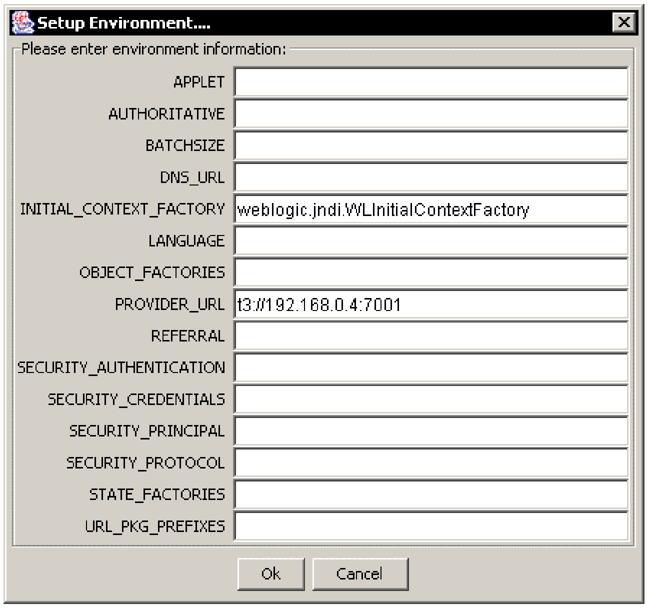
List of Methods:
- findByTotalRange
- findByProduct
- findByRegion
- findAll

Parameter List:
- param1
- param2

Parameter Type : double
400

Specifying EJB Parameter Values Dialog

When you select a parameter, the data type of the parameter will appear in the lower portion of the window. Below that you will be able to specify a value for the parameter. Be sure to enter a correct value for the data type and then click the *Set Value* button. This will fix the parameter values. Once you finish setting up all parameter values, click the *Environment* button. This will bring up a new dialog allowing you to specify environment properties for your application server. This information is necessary for EspressManager to connect to the EJB.



Setup Environment....

Please enter environment information:

APPLET

AUTHORITATIVE

BATCHSIZE

DNS_URL

INITIAL_CONTEXT_FACTORY weblogic.jndi.WLInitialContextFactory

LANGUAGE

OBJECT_FACTORIES

PROVIDER_URL t3://192.168.0.4:7001

REFERRAL

SECURITY_AUTHENTICATION

SECURITY_CREDENTIALS

SECURITY_PRINCIPAL

SECURITY_PROTOCOL

STATE_FACTORIES

URL_PKG_PREFIXES

EJB Environment Setup

The fields here are the available environment properties for the JNDI context interface. You don't have to specify all values, only the information necessary for your environment (application server). Once you finish specifying the environment variables, click the *OK* button. You will be returned to the EJB setup window. Click *OK* again to finish setting up the EJB data source. A new node will appear under *EJBs* with your EJB.

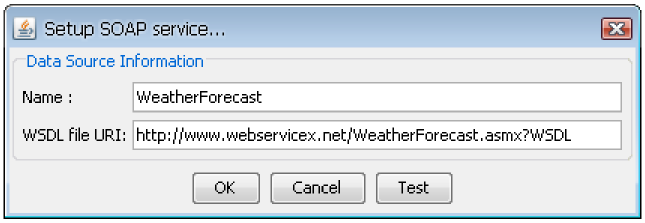
You can modify the parameter values by selecting your EJB source, and clicking the *Edit* button.

There is a sample EJB data source included in the installation in `help/examples/DataSources/EJB` directory. The directory contains `sales.ear` and `salesClient.jar` files, as well as the source code for the Sales entity bean. The `sales.ear` file is designed to be deployed in Java 2 Reference Implementation and uses the Cloudscape database as the underlying storage mechanism. You can use `SalesClient.java` program to populate the Cloudscape database. The `salesClient.jar` file contains the stub classes to connect to the deployed Sales EJB and needs to be in the EspressManager classpath.

1.3.7. Data from SOAP with WSDL support

EspressReport also allows you to retrieve data using SOAP (Service Oriented Architecture Protocol). To connect to a SOAP data source using WSDL, you don't need to know any URLs for the services, SOAP actions, operation names or parameters. All you need to know is a location of WSDL file, which contains all the necessary information.

To set up a SOAP data source, select the *SOAPServices* node in the Data Source Manager and click the *Add* button. A dialog will open prompting you to specify options for the new SOAP data source.



Setup SOAP service...

Data Source Information

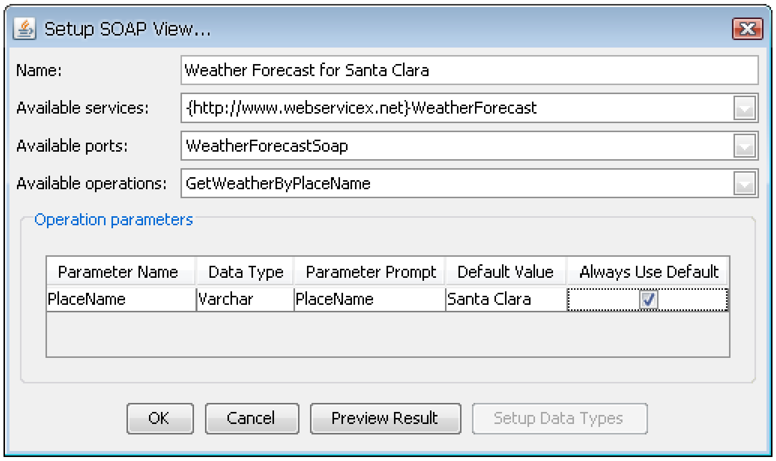
Name : WeatherForecast

WSDL file URI: http://www.webservicex.net/WeatherForecast.asmx?WSDL

SOAP data source setup

The first option allows you to specify a display name for the data source. The second option allows you to specify a location of the WSDL file. The location can be either absolute path on the server or path relative to your EspressReport installation directory or URL. Once you specify the connection information, you can test the connection to the WSDL file by clicking the *Test* button. This will test retrieving the WSDL file from the URI you've provided, check the file for any supported SOAP operations and report any problems. After clicking *OK*, a new SOAP data source node will be added to the data registry.

To add a new SOAP View, select an existing SOAP data source and click the *Add* button. A dialog will open prompting you for all the parameters necessary to make a SOAP query.



Name: Weather Forecast for Santa Clara

Available services: {http://www.webservicex.net}WeatherForecast

Available ports: WeatherForecastSoap

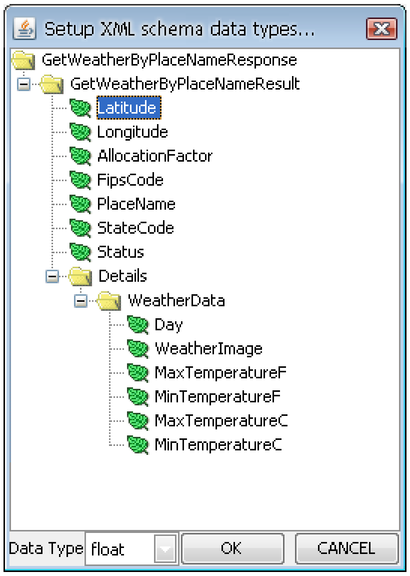
Available operations: GetWeatherByPlaceName

Parameter Name	Data Type	Parameter Prompt	Default Value	Always Use Default
PlaceName	Varchar	PlaceName	Santa Clara	☑

Setup SOAP View dialog

The first option in the dialog allows you to specify a display name in the Data Source Manager. Next, there are three drop-down menus in the dialog. The first drop-down menu contains all the SOAP services described in the WSDL file. Once you specify a service, the second drop-down list will be populated with all the ports of this service. Selecting a port will populate the last drop-down list with all the operations of this port. If you move the mouse over any drop-down list, a hint will appear with documentation for the service/port/operation (if the documentation is provided in the WSDL file). After specifying the service, port and operation, all the parameters of the operation will be read. If there are some parameters, they will be displayed in a table. The first two columns of the table are not editable. They are read from the WSDL file. The next 2 columns are editable and allow you to specify parameter prompts and default values. The last column contains a checkbox which allows you to choose whether the default parameter value will always be used or not. This means that this parameter value will be fixed and you will not be prompted for it. All the parameters that don’t have this checkbox checked will be used as report/chart parameters.

The *Setup Data Types* button is only available when editing the SOAP View from the Data Source Manager and allows you to adjust data types when necessary. In order to verify result from the SOAP response, click the *Preview Result* button. All the default values will then be tested to see if they have proper data type or not. In case they do not match, you will be prompted to adjust them. After that, you will get the setup data types dialog (the same as for XML data source). If the data source is parameterized, you will get the parameter prompt dialog before specifying data types. Please note that in order to generate the XML schema properly, you have to specify existing parameter values.

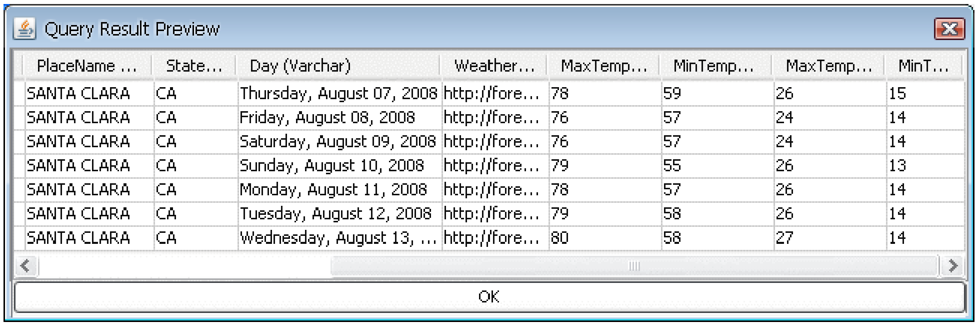


- GetWeatherByPlaceNameResponse
  - GetWeatherByPlaceNameResult
    - Latitude
    - Longitude
    - AllocationFactor
    - FipsCode
    - PlaceName
    - StateCode
    - Status
    - Details
      - WeatherData
        - Day
        - WeatherImage
        - MaxTemperatureF
        - MinTemperatureF
        - MaxTemperatureC
        - MinTemperatureC

Data Type: float

Setup XML Schema Data Types dialog

You can setup data types from this dialog. The behavior is exactly the same as for XML data source with DTD schema (see Section 1.3.3 - Data from XML and XBRL Files). Once you finis specifying data types, click the *OK* button. A dialog will open showing you the result preview.



PlaceName ...	State...	Day (Varchar)	Weather...	MaxTemp...	MinTemp...	MaxTemp...	MinT...
SANTA CLARA	CA	Thursday, August 07, 2008	http://fore...	78	59	26	15
SANTA CLARA	CA	Friday, August 08, 2008	http://fore...	76	57	24	14
SANTA CLARA	CA	Saturday, August 09, 2008	http://fore...	76	57	24	14
SANTA CLARA	CA	Sunday, August 10, 2008	http://fore...	79	55	26	13
SANTA CLARA	CA	Monday, August 11, 2008	http://fore...	78	57	26	14
SANTA CLARA	CA	Tuesday, August 12, 2008	http://fore...	79	58	26	14
SANTA CLARA	CA	Wednesday, August 13, ...	http://fore...	80	58	27	14

Query Result Preview dialog

Clicking the *OK* button in this dialog will take you back to the Setup SOAP View dialog. Once you finish specifying all necessary information, click the *OK* button in the dialog. A new node will be added under your SOAP data source in the Data Source Manager and can now be used to create a report or chart.

1.3.8. Data from Salesforce

The Salesforce data source is designed for existing Salesforce users who want to display their Salesforce data in EspressReport. The connection to the Salesforce server is established via SOAP using Salesforce Partner WSDL (version 13.0). Users communicate with Salesforce server by SOQL (Salesforce Object Query Language) queries. Please note that users must have valid Salesforce accounts with username and password to work with this data source. Moreover, users who use the EspressReport Salesforce data source must have access to Salesforce account from trusted networks. To add your IP address to the trusted IP list, you have to activate your computer as described below.

For more information about SOQL queries and activating Salesforce user's accounts from trusted networks, please visit the following Salesforce sites:

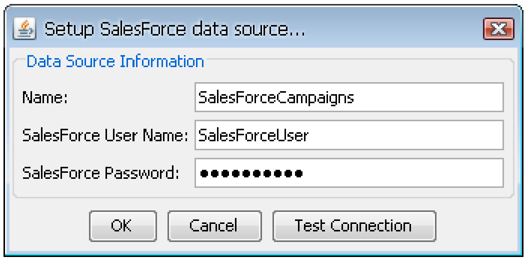
SOQL queries

https://developer.salesforce.com/docs/atlas.en-us.soql_sosl.meta/soql_sosl/sforce_api_calls_soql_sosl_intro.htm

Activating Salesforce user's accounts

https://help.salesforce.com/s/articleView?id=sf.security_networkaccess.htm

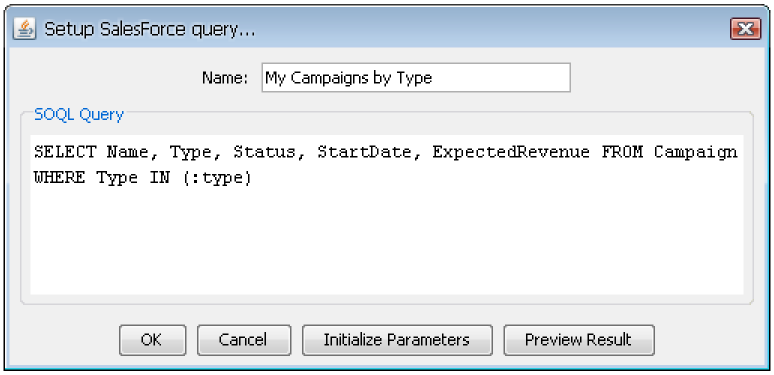
To set up a Salesforce data source, select the *SalesForce* node in the Data Source Manager and click the *Add* button. A dialog will open prompting you to specify a display name for the data source, user name and password to your Salesforce account. Once you specify the connection information, you can test the connection to your Salesforce account by clicking the *Test Connection* button. This will test the connection using the information you've provided and report any problems.



Name: SalesForceCampaigns
SalesForce User Name: SalesForceUser
SalesForce Password: ●●●●●●●●●●●

Setup SalesForce Data Source Dialog

Once you add a Salesforce data source, a new node will appear in the Data Source Manager window. To add a new Salesforce query, click the *Add* button. A new dialog will open prompting you to specify a query name and SOQL query.



Name: My Campaigns by Type

```
SELECT Name, Type, Status, StartDate, ExpectedRevenue FROM Campaign
WHERE Type IN (:type)
```

Setup SalesForce Query Dialog

Please note that only child-to-parent relationship queries are supported in the current EspressReport version. You cannot use parent-to-child queries (using nested SOQL queries). For more information about Salesforce relationship queries and their syntaxes, please visit the following Salesforce site: https://developer.salesforce.com/docs/atlas.en-us.soql_sosl.meta/soql_sosl/sforce_api_calls_soql_relationships.htm

Moreover, this dialog allows you to initialize query parameters in case that your query contains single value or multi value parameters. A parameter is specified within an SOQL statement using the ":" character. Generally the parameter is placed in WHERE clause of an SOQL Select statement. For example, the following SOQL statement

```
Select Name, Type, Status, ActualCost From Campaign Where Name
 = :CampaignName
```

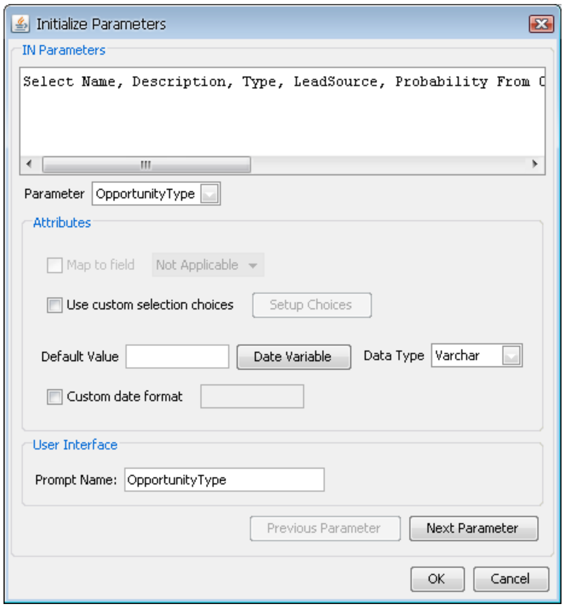
specifies a single value parameter called *CampaignName*. You would then be able to enter a campaign name at run-time and only retrieve data for that campaign.

Another example of SOQL statement shows using of multi value parameters that take an array of values as the input rather than single values.

```
Select Name, Description, Type, LeadSource, Probability From Opportunity
 Where Type IN (:OpportunityType) And LeadSource IN (:OppLeadSource)
```

The statement specifies two multi value parameters called *OpportunityType* and *OppLeadSource*. You would then be able to specify opportunity types and lead sources at run-time and you will only retrieve data according to specified parameters values.

In order to initialize SOQL query parameters, click the *Initialize Parameters* button. The initialize parameters dialog will then appear allowing you to specify parameters mapping.



Select Name, Description, Type, LeadSource, Probability From O

Parameter: OpportunityType

□ Map to field: Not Applicable

□ Use custom selection choices

Default Value:

Data Type: Varchar

□ Custom date format

Prompt Name: OpportunityType

Initialize Parameters Dialog

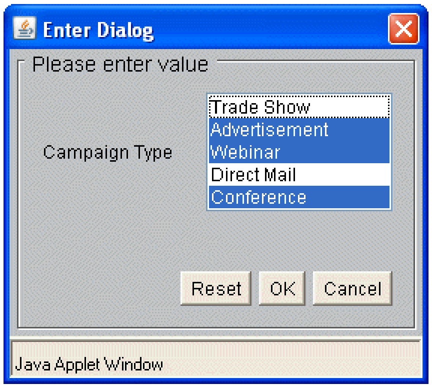
From this dialog, you can specify the following options:

Map to field: This allows you to specify a field from the Salesforce data source whose values will be used for the parameter input. Selecting this option modifies the parameter prompt that you will get when previewing or running the report/chart. If you map the parameter to a Salesforce field, you will be prompted with a drop-down list of distinct values from which you can select a parameter value. If you do not map, you will have to type in specific parameter value.

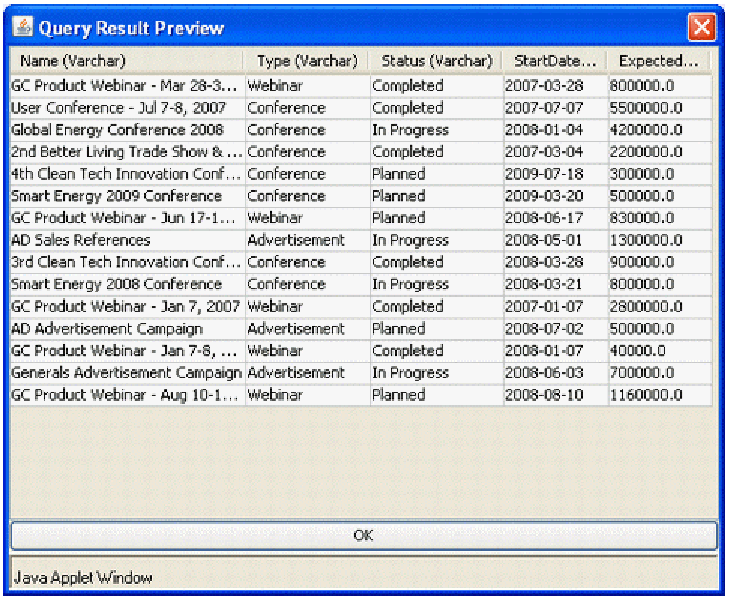
Use custom selection choices: Rather than having a drop-down menu with all the distinct column values, you can build a custom list of parameter values. To set up the list, select this option and click the *Setup Choices* button. This will launch a new dialog allowing you to create a list of choices.

The rest of the options are basically same as for database query parameters. For further information about initializing database query parameters, see Section 1.3.2.2.2.2 - Initializing Query Parameters. Once you specify mapping for all available parameters, click the *OK* button and you will be taken back to the Setup Salesforce Query dialog.

From the Setup SalesForce Query dialog, you can also preview the query result using the *Preview Result* button to verify output from your query. In case you have a parameterized query, the parameter prompt dialog will appear prompting you to specify parameter values. Once you specify the parameter values, click the *OK* button and the query result preview dialog will appear.



Parameter Prompt



Name (Varchar)	Type (Varchar)	Status (Varchar)	StartDate...	Expected...
GC Product Webinar - Mar 28-3...	Webinar	Completed	2007-03-28	800000.0
User Conference - Jul 7-8, 2007	Conference	Completed	2007-07-07	5500000.0
Global Energy Conference 2008	Conference	In Progress	2008-01-04	4200000.0
2nd Better Living Trade Show & ...	Conference	Completed	2007-03-04	2200000.0
4th Clean Tech Innovation Conf...	Conference	Planned	2009-07-18	300000.0
Smart Energy 2009 Conference	Conference	Planned	2009-03-20	500000.0
GC Product Webinar - Jun 17-1...	Webinar	Planned	2008-06-17	830000.0
AD Sales References	Advertisement	In Progress	2008-05-01	1300000.0
3rd Clean Tech Innovation Conf...	Conference	Completed	2008-03-28	900000.0
Smart Energy 2008 Conference	Conference	In Progress	2008-03-21	800000.0
GC Product Webinar - Jan 7, 2007	Webinar	Completed	2007-01-07	2800000.0
AD Advertisement Campaign	Advertisement	Planned	2008-07-02	500000.0
GC Product Webinar - Jan 7-8, ...	Webinar	Completed	2008-01-07	40000.0
Generals Advertisement Campaign	Advertisement	In Progress	2008-06-03	700000.0
GC Product Webinar - Aug 10-1...	Webinar	Planned	2008-08-10	1160000.0

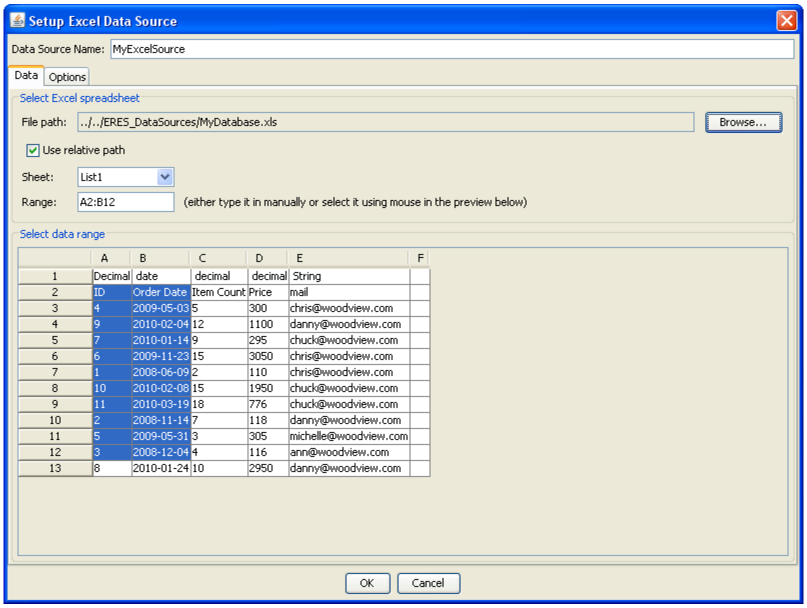
Query Result Preview Dialog

From this dialog, you can verify the query output. Clicking the *OK* button will take you back to the Setup SalesForce Query Dialog.

Once you specify the query, click the *OK* button. The query will then be added as a new node under your Salesforce data source in the Data Source Manager and can now be used to create a report or chart.

1.3.9. Data from Excel files

EspressReport also allows you to design reports using data retrieved from Excel files. To add an Excel file as a data source, select the ExcelFiles node in the Data Source Manager and click the *Add* button. A dialog will open prompting you to specify the data source name and to select the Excel file from which the data should be imported.



Data Source Name: MyExcelSource

File path: ../../ERES_DataSources/MyDatabase.xls

☑ Use relative path

Sheet: List1

Range: A2:B12 (either type it in manually or select it using mouse in the preview below)

	A	B	C	D	E	F
1	Decimal	date	decimal	decimal	String	
2	ID	Order Date	Item Count	Price	mail	
3	4	2009-05-03	5	300	chris@woodview.com	
4	9	2010-02-04	12	1100	danny@woodview.com	
5	7	2010-01-14	9	295	chuck@woodview.com	
6	6	2009-11-23	15	3050	chris@woodview.com	
7	1	2008-06-09	2	110	chris@woodview.com	
8	10	2010-02-08	15	1950	chuck@woodview.com	
9	11	2010-03-19	18	776	chuck@woodview.com	
10	2	2008-11-14	7	118	danny@woodview.com	
11	5	2009-05-31	3	305	michelle@woodview.com	
12	3	2008-12-04	4	116	ann@woodview.com	
13	8	2010-01-24	10	2950	danny@woodview.com	

Setup Excel Data Source dialog - Data

After selecting the Excel file, the imported data will be previewed in the dialog. If the checkbox *Use Relative Path* is checked, the file path to the selected Excel file will be set as relative to the EspressReport installation directory. Otherwise the full path will be used. In case the Excel file is stored on a different disk drive from the one which EspressReport is installed on, this option is not available. You can select the sheet (if there is more than one in the file) and the cells which are relevant for the data source being designed using your mouse or by specifying the range in the *Range* box (for instance, you can specify that your source will use the data from the columns A and B and from the rows 2 to 12 by typing **`A2:B12`** in the *Range* box; this is the format which is also used for ranges in MS Excel). You can also select the data by clicking the row header or the column header. Hold **Ctrl** or **Shift** to select more rows or columns. Click the top left corner to select all cells. Again, this behavior is similar to MS Excel.

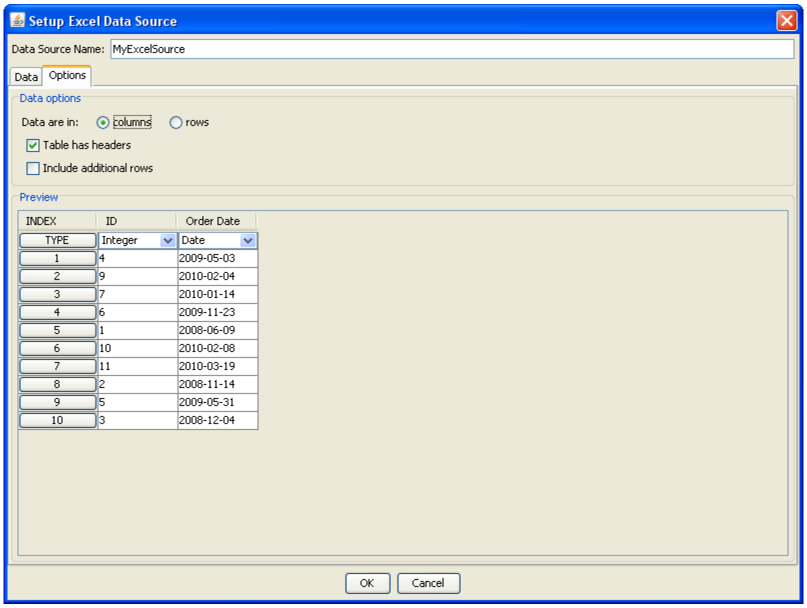
Please note that EspressReport can process both `*.xls` files and `*.xlsx` files. The `*.xlsx` files are used in Microsoft Excel 2007 and newer and they are based on Open XML.

EspressReport can also process basic Excel formulas. If it is not able to process the entered formula, an error message will be displayed.

Empty rows (in case the data are in columns) or columns (in case the data are in rows) at the end of the sheet are automatically removed from the data source, even in case they have been selected.

Click on the *Options* tab to specify whether the data are in columns or in rows to get the data structure correctly. You can also specify whether table headers are included in your data selection. The option *Include additional rows* or *Include additional columns* (depending on whether the data are in columns or rows) allows you to automatically include data rows or columns into the Excel file after the data source has been created without the need to change the data source in the Data Source Manager manually.

The bottom part of the dialog on the *Options* tab shows you the data source content following the current configuration. You can change the data type for each column of the data source there, if desired. The data types are detected automatically, therefore changing the type should not be necessary in most cases.


Setup Excel Data Source
Data Source Name: MyExcelSource
Data
Options
Data options
Data are in: ● columns ○ rows
☑ Table has headers
☐ Include additional rows
Preview
OK
Cancel

INDEX	ID	Order Date
TYPE	Integer	Date
1	4	2009-05-03
2	9	2010-02-04
3	7	2010-01-14
4	6	2009-11-23
5	1	2008-06-09
6	10	2010-02-08
7	11	2010-03-19
8	2	2008-11-14
9	5	2009-05-31
10	3	2008-12-04

Setup Excel Data Source dialog - Options

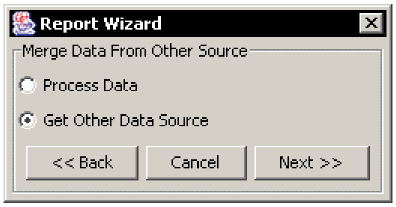
Click *OK* to save the data source when the configuration is finished.

1.3.10. Using Data for Reports

1.3.10.1. Using Multiple Data Sources

Once you select the data source you want to use and click the *Next* button, the next screen will display first twenty records from the data source (With the exception of data views, which will require you to select fields and set conditions first). You can display all of the records by checking the *Show All Records* box.

EspressReport allows you to construct reports from multiple data sources. Once you select your first source and click the *Next* button from the data table window, you will be presented with a dialog asking if you would like to process the current data or select another data source.

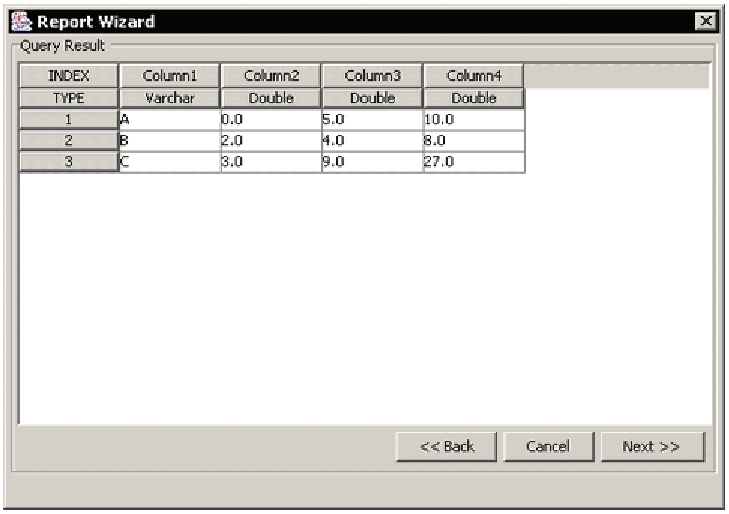


Additional Data Source Dialog

If you select *Process Data* and click the *Next* button, you will continue to the next step in the chart wizard. If you select *Get Other Data Source*, you will return to the Data Source Manager to select another data source for the chart. You can repeat this process to select as many sources as you want.

Multiple data sources are combined sideways in the data table. This means that the columns from the second (or third, fourth, etc.) data sources are placed to the right next to the columns from the first data source. If the columns from one data source have more rows than the other ones, null values will be placed in extra rows.

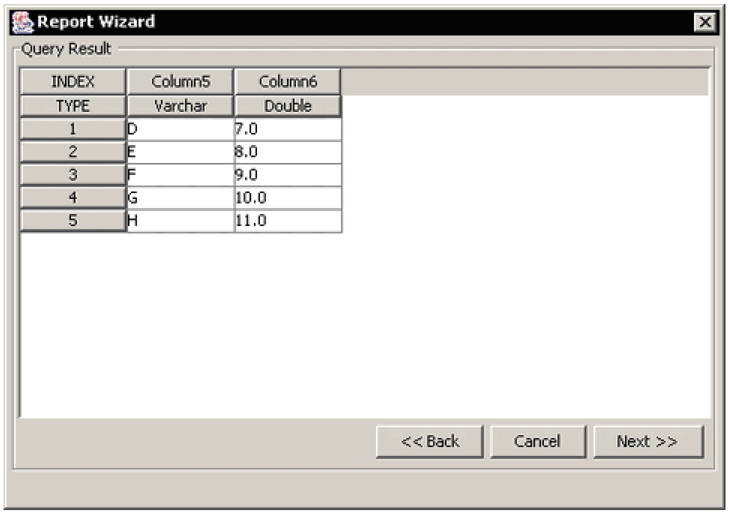
For example, assuming you want to use two data sources to create a report. The data table generated by the first source will look like this:



INDEX	Column1	Column2	Column3	Column4
TYPE	Varchar	Double	Double	Double
1	A	0.0	5.0	10.0
2	B	2.0	4.0	8.0
3	C	3.0	9.0	27.0

Data From the First Data Source

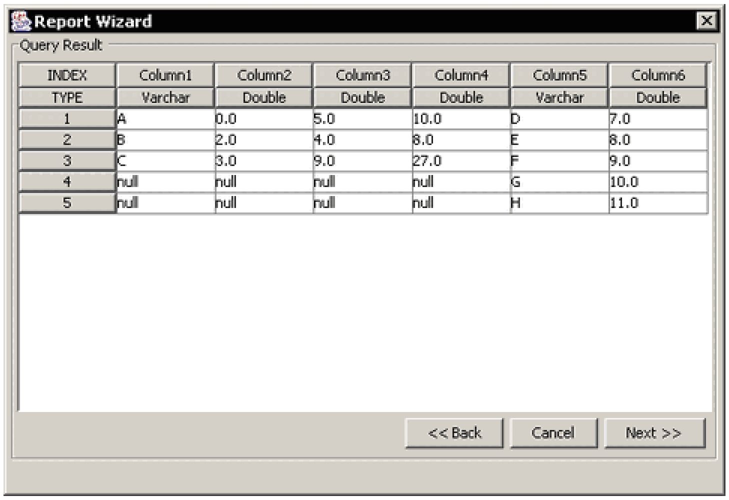
The data from the second source will look like this:



INDEX	Column5	Column6
TYPE	Varchar	Double
1	D	7.0
2	E	8.0
3	F	9.0
4	G	10.0
5	H	11.0

Data From the Second Data Source

When you combine two data sources, you will get the following data table:



INDEX	Column1	Column2	Column3	Column4	Column5	Column6
TYPE	Varchar	Double	Double	Double	Varchar	Double
1	A	0.0	5.0	10.0	D	7.0
2	B	2.0	4.0	8.0	E	8.0
3	C	3.0	9.0	27.0	F	9.0
4	null	null	null	null	G	10.0
5	null	null	null	null	H	11.0

Data From Combined Sources

As you can see, the two data sources have been placed side by side. Since the second source has more rows of data than the first, additional rows of null data were added.

Note

You cannot use parameterized data sources to create a multiple data source.

You can also use multiple data sources in reports by adding sub-reports. For more information about sub-reports, please see Section 1.11 - Sub-Reports.

1.3.10.2. Change Data Source

At any point during report design, you can change the template's data source. Since chart templates are saved with their data source information, you must use the option within report designer to change the template's data source. Simply altering a data source in the registry will not affect the template unless you use the data source updating feature (For more information about this, see Section 1.3.11 - Data Source Updating). To change a template's data source, select Data → Change Data Source or click the *Change Data Source* button on the toolbar. This will bring up the Data Source Manager allowing you to select a new data source or modify an existing one.

If the new data source is significantly different from the previous data source in number of columns or data types, you might need to re-map the data to the report. This is necessary because some layouts or computations may no longer work correctly with major changes to the structure of the report's data. For more information about layouts and data mapping, see Section 1.4 - Report Types and Data Mapping.

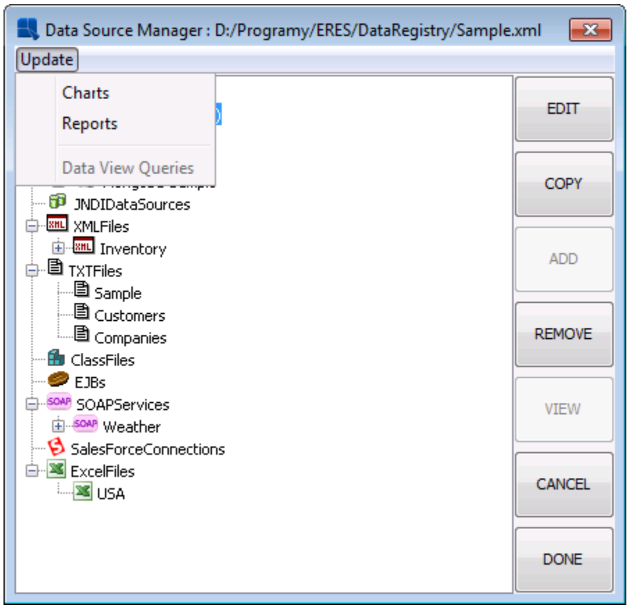
1.3.11. Data Source Updating

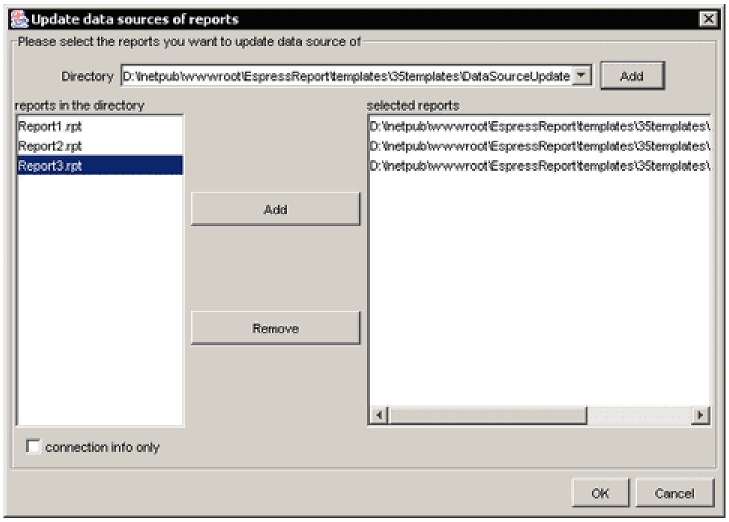
There are many circumstances where you will want to move a group of templates or a complete installation of EspressReport from one location to another. For example an application can move from a development to a production environment. In each environment, the location and connection information for data sources may be different. In this scenario, updating report templates one by one as detailed in the previous section isn't a feasible way to change the connection information for a large number of templates. Instead, EspressReport allows you to quickly update a group of templates based on information in the data registry.

Although report templates maintain an internal copy of the data source information (allowing them to be deployed independently), they also maintain information (location & source) about the data registry from which they were created. Hence, when you modify a report's query (as detailed in Section 1.3.2.4 - Editing Queries), you have the option to save the changes back to the data registry. In order to use this feature, you will have to keep your data registry up to date. This means that query changes should be saved back to the registry and changes in data view structure should be propagated to data view queries as detailed in Section 1.3.2.3.2 - Updating Data View Queries.

To use this feature, first make any modifications to the data registry that you want to propagate to the report templates. These modifications can include database connection information, file locations for text, XML data files, and even changes to data views or queries that you want to pass to the templates. Note that if you're moving reports between installations, the data registry file will need to be moved to the same relative location in the new installation. In addition, the associated query files for that registry (`.qry`/`.dvw`/`.ddt`) will need to be moved to `/queries/` directory of the new installation (Query file names begin with the name of the registry).

From the data source manager, select Update → Reports. This will bring up a dialog allowing you to select which reports you want to update.




Update data sources of reports
Please select the reports you want to update data source of
Add
Remove
OK
Cancel

Directory D:\inetpub\wwwroot\EspressReport\templates\35templates\DataSourceUpdate

reports in the directory

Report1.rpt
Report2.rpt
Report3.rpt

selected reports

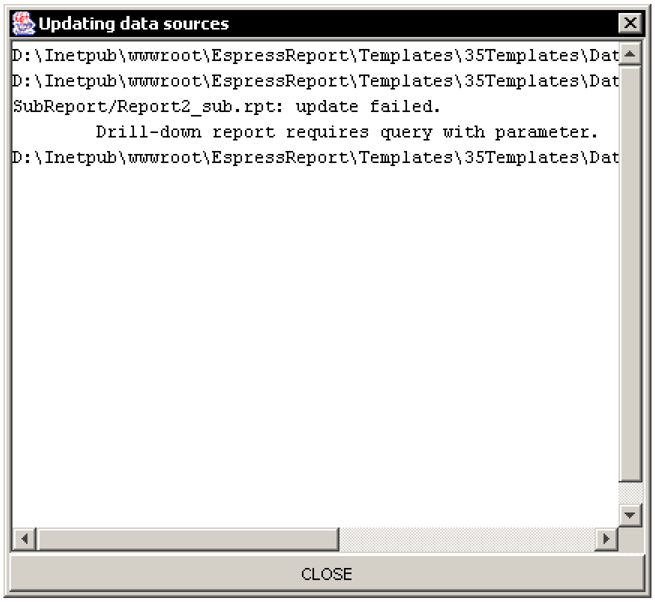
D:\inetpub\wwwroot\EspressReport\templates\35templates\
D:\inetpub\wwwroot\EspressReport\templates\35templates\
D:\inetpub\wwwroot\EspressReport\templates\35templates\

☐ connection info only

Select Reports for Data Source Updating

To select reports to update, first browse to the directory that contains the reports. After you select a directory, any report templates will appear in the left side of the dialog. You can select any templates you want to update and click the *Add* button. You can navigate to as many different directories as you want to select templates. Selecting the *Connection Info Only* option will only update the connection information (database URL, driver, username, password, and locations for XML files, text files and Java classes). Queries and data view information will not be updated in the templates.

Once you finish selecting the templates you want to update, click *OK* and the updating process will begin. A dialog will display the current progress and any errors.



```
D:\Inetpub\wwwroot\EspressReport\Templates\35Templates\Dat
D:\Inetpub\wwwroot\EspressReport\Templates\35Templates\Dat
SubReport/Report2_sub.rpt: update failed.
        Drill-down report requires query with parameter.
D:\Inetpub\wwwroot\EspressReport\Templates\35Templates\Dat
```

Updating Data Sources Progress/Log Dialog

A log file named `UpdateDataSources` will also be written in the root directory of the installation with the contents of the progress screen. Reports that fail the updating for various reasons can be updated manually using the option in ReportDesigner.

Note

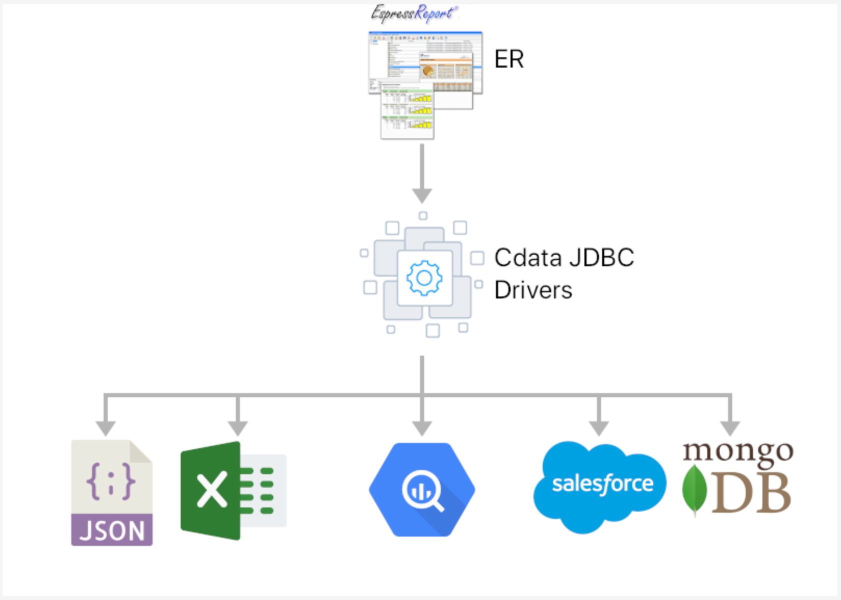
Only reports for the current data registry will be modified. If you select reports that do not retrieve their data from a source in the current registry, they will be ignored.

This feature can also be used to update chart templates that have an independent data source. If you launch the data registry from within the Chart Designer interface, the *Chart* option on the Update menu will become active. By selecting this option you can select a group of charts that have been developed using the registry and update their data sources in the same way.

1.3.12. CData JDBC drivers

If you want to connect to a data source driver that doesn’t ship with EspressReport, CData JDBC drivers are an interesting option.

CData JDBC drivers can be also used to increase capabilities of non-database data sources like Excel or JSON.



Tip

Although CData JDBC drivers are a 3rd party commercial product, you can install a free trial version to test the product before purchasing.

Note

CData JDBC drivers are a 3rd party product. If you want to use the drivers, you have to purchase them at https://www.cdata.com

1.3.12.1. Supported CData Drivers

Currently, we support the following CData JDBC drivers:

- Salesforce
- BigQuery
- Excel
- JSON
- MongoDB
- Kintone

Other CData JDBC drivers might work too but we can not guarantee full functionality for unsupported drivers. If you require a connection to a driver that is not listed here, please contact us at `<support@quadbase.com>`

1.3.12.1.1. Excel

Download: https://www.cdata.com/drivers/excel/jdbc

Official CData Documentation (for the JDBC driver only): https://cdn.cdata.com/help/RXF/jdbc

After downloading the driver, see the following chapters: Section 1.3.12.2 - CData JDBC driver installation, Section 1.3.12.3 - Deploying the CData JDBC Driver in EspressReport and Section 1.3.12.4 - Using the CData JDBC drivers in DataSource Manager

The CData Excel driver allows you to run SQL queries on top of an Excel file just like it was a database. This allows you to use the QueryBuilder, build Data Views in a graphical user interface as well as write SQL queries manually (in Query Builder).

> **Tip**
>
> You can also write queries with parameters and multi-value parameters using Excel files as the data source.

Since Excel does not have data types set for columns as a normal database has, determining the data type for the columns can be a bit tricky at times.By default, we added the following parameters to the CData Excel driver connection URL:

```
TypeDetectionScheme=RowScan;
RowScanDepth=10;
```

This makes the CData Excel driver scan the first ten rows when loading the selected Excel file and detect the data source for each column automatically based on the data in the first ten rows.

However, there are multiple options of how to determine the data type for each column.

For more options, read the CData JDBC Excel driver documentation about the TypeDetectionScheme parameter: https://cdn.cdata.com/help/RXH/jdbc/RSBExcel_p_TypeDetectionScheme.htm

You can change the TypeDetectionScheme value in the Setup Database… dialog in Data Source Manager in the URL field (the field where you entered the Excel file path).

1.3.12.1.2. JSON

Download: https://www.cdata.com/drivers/json/jdbc

Official CData Documentation (for the JDBC driver only): https://cdn.cdata.com/help/DJF/jdbc

After downloading the driver, see the following chapters: Section 1.3.12.2 - CData JDBC driver installation, Section 1.3.12.3 - Deploying the CData JDBC Driver in EspressReport and Section 1.3.12.4 - Using the CData JDBC drivers in DataSource Manager

The CData json driver allows you to run SQL queries on top of a json file just like it was a database. This allows you to use the QueryBuilder, build Data Views in a graphical user interface as well as write SQL queries manually (in Query Builder).

> **Tip**
>
> You can also write queries with parameters and multi-value parameters using JSON files as the data source.

1.3.12.1.3. Salesforce

To connect to Salesforce:

Download: https://www.cdata.com/drivers/salesforce/jdbc

Official CData Documentation (for the JDBC driver only): https://cdn.cdata.com/help/RFF/jdbc

After downloading the driver, see the following chapters: Section 1.3.12.2 - CData JDBC driver installation, Section 1.3.12.3 - Deploying the CData JDBC Driver in EspressReport and Section 1.3.12.4 - Using the CData JDBC drivers in DataSource Manager

1.3.12.1.4. MongoDB

To connect to MongoDB:

Download: https://www.cdata.com/drivers/mongodb/jdbc

A bug fix was added to newer MongoDB driver from https://cdatabuilds.s3.amazonaws.com/support/DGR-JV_8598.exe

Official CData Documentation (for the JDBC driver only): https://cdn.cdata.com/help/DGF/jdbc

After downloading the driver, see the following chapters: Section 1.3.12.2 - CData JDBC driver installation, Section 1.3.12.3 - Deploying the CData JDBC Driver in EspressReport and Section 1.3.12.4 - Using the CData JDBC drivers in DataSource Manager

1.3.12.1.5. BigQuery

To connect to BigQuery:

Download: https://www.cdata.com/drivers/bigquery/jdbc

Official CData Documentation (for the JDBC driver only): https://cdn.cdata.com/help/DBF/jdbc

After downloading the driver, see the following chapters: Section 1.3.12.2 - CData JDBC driver installation, Section 1.3.12.3 - Deploying the CData JDBC Driver in EspressReport and Section 1.3.12.4 - Using the CData JDBC drivers in DataSource Manager

1.3.12.1.6. Kintone

To connect to Kintone:

Download: https://www.cdata.com/drivers/kintone/jdbc

Official CData Documentation (for the JDBC driver only): https://cdn.cdata.com/help/DBF/jdbc

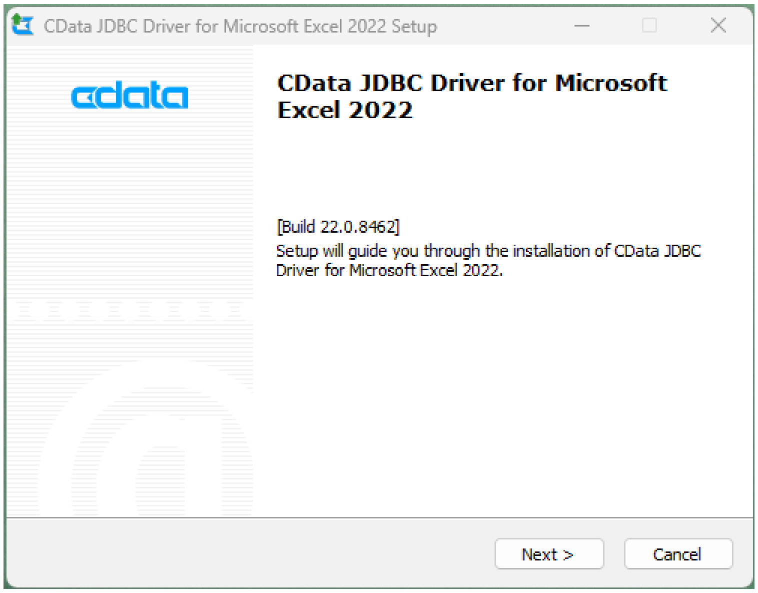
After downloading the driver, see the following chapters: Section 1.3.12.2 - CData JDBC driver installation, Section 1.3.12.3 - Deploying the CData JDBC Driver in EspressReport and Section 1.3.12.4 - Using the CData JDBC drivers in DataSource Manager

1.3.12.2. CData JDBC driver installation

The installation process of all CData JDBC drivers is basically the same for all the supported data sources. We'll guide you through the process of installing the CData Excel JDBC driver for example.

First, install the CData JDBC Driver of your choice. You will find the links to each supported CData JDBC driver in the chapters below.

After downloading the installer, proceed with the setup according to the dialogs.



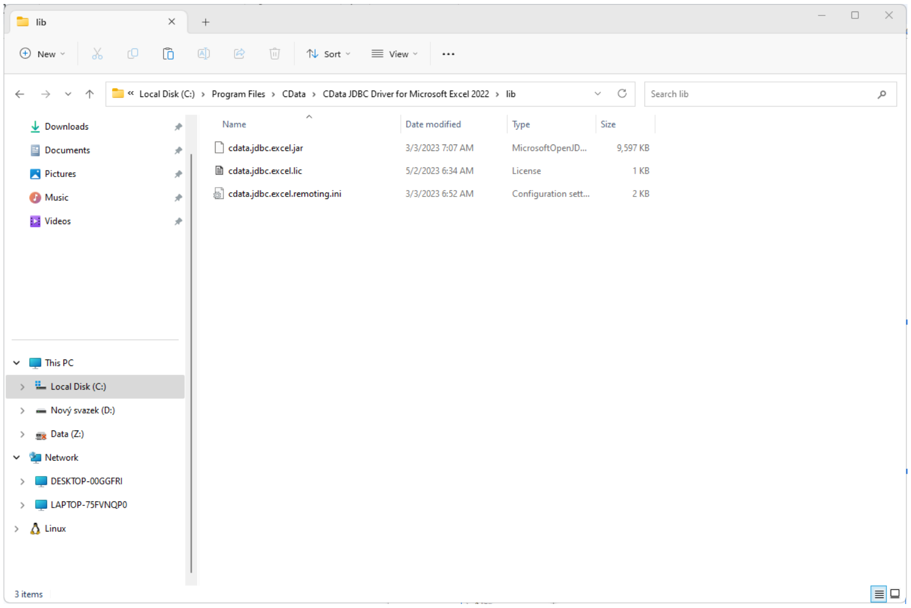
CData JDBC Driver for Microsoft Excel 2022 Setup

CData JDBC Driver for Microsoft Excel 2022

[Build 22.0.8462]
Setup will guide you through the installation of CData JDBC Driver for Microsoft Excel 2022.

In the following dialog, you can choose to either install a paid version of the product or a free 30-day trial version. In this example, we'll use the trial version.

When the setup is complete, JDBC Driver will be created in "CData JDBC Driver for Microsoft Excel 2022\lib".



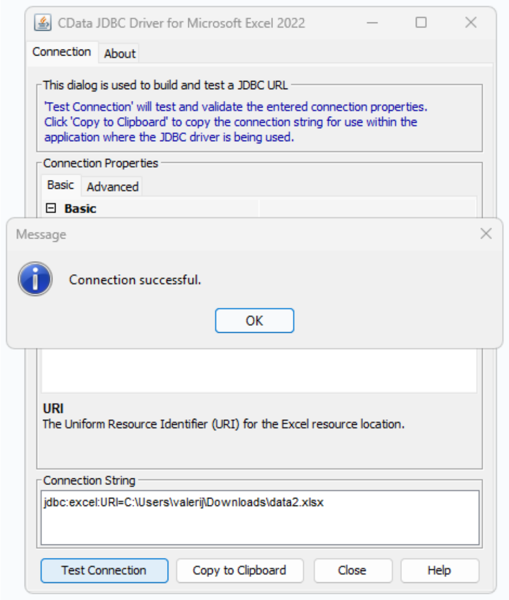
Name	Date modified	Type	Size
cdata.jdbc.excel.jar	3/3/2023 7:07 AM	MicrosoftOpenJD...	9,597 KB
cdata.jdbc.excel.lic	5/2/2023 6:34 AM	License	1 KB
cdata.jdbc.excel.remoting.ini	3/3/2023 6:52 AM	Configuration sett...	2 KB

Tip

Executing "cdata.jdbc.excel.jar" will launch the connection test tool. You can use this tool to test or troubleshoot the CData driver.

Tip

The connection test tool also displays you the connection URL that can be used in EspressReport Data Source Manager.



This dialog is used to build and test a JDBC URL

'Test Connection' will test and validate the entered connection properties. Click 'Copy to Clipboard' to copy the connection string for use within the application where the JDBC driver is being used.

Connection successful.

URI
The Uniform Resource Identifier (URI) for the Excel resource location.

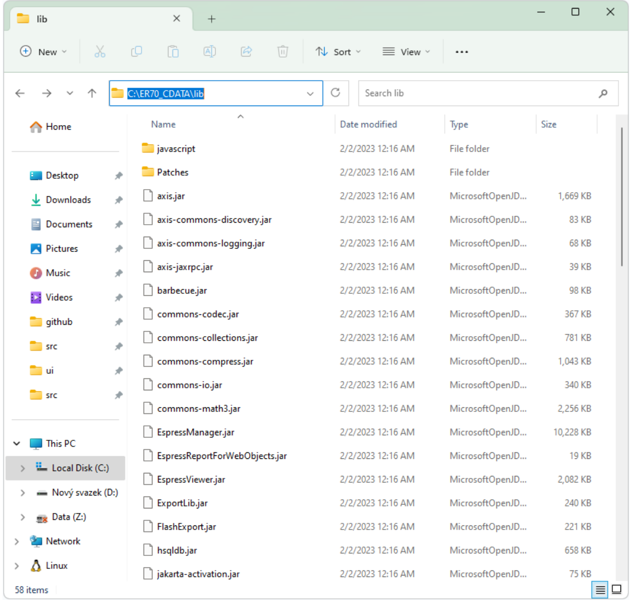
Connection String

```
jdbc:excel:URI=C:\Users\valerij\Downloads\data2.xlsx
```

1.3.12.3. Deploying the CData JDBC Driver in EspressReport

Locate the "CData Installation Directory"\CData JDBC Driver for Microsoft Excel 2023\lib (for example: C:\Program Files\CData\CData JDBC Driver for Microsoft Excel 2023\lib) on your hard drive.The directory should contain three files: cdata.jdbc.excel.jar, cdata.jdbc.excel.lic, cdata.jdbc.excel.remoting

Copy the three files to ER/lib/



C:\ER70_CDATA\lib

Search lib

Name	Date modified	Type	Size
javascript	2/2/2023 12:16 AM	File folder	
Patches	2/2/2023 12:16 AM	File folder	
axis.jar	2/2/2023 12:16 AM	MicrosoftOpenJD...	1,669 KB
axis-commons-discovery.jar	2/2/2023 12:16 AM	MicrosoftOpenJD...	83 KB
axis-commons-logging.jar	2/2/2023 12:16 AM	MicrosoftOpenJD...	68 KB
axis-jaxrpc.jar	2/2/2023 12:16 AM	MicrosoftOpenJD...	39 KB
barbecue.jar	2/2/2023 12:16 AM	MicrosoftOpenJD...	98 KB
commons-codec.jar	2/2/2023 12:16 AM	MicrosoftOpenJD...	367 KB
commons-collections.jar	2/2/2023 12:16 AM	MicrosoftOpenJD...	781 KB
commons-compress.jar	2/2/2023 12:16 AM	MicrosoftOpenJD...	1,043 KB
commons-io.jar	2/2/2023 12:16 AM	MicrosoftOpenJD...	340 KB
commons-math3.jar	2/2/2023 12:16 AM	MicrosoftOpenJD...	2,256 KB
EspressManager.jar	2/2/2023 12:16 AM	MicrosoftOpenJD...	10,228 KB
EspressReportForWebObjects.jar	2/2/2023 12:16 AM	MicrosoftOpenJD...	19 KB
EspressViewer.jar	2/2/2023 12:16 AM	MicrosoftOpenJD...	2,082 KB
ExportLib.jar	2/2/2023 12:16 AM	MicrosoftOpenJD...	240 KB
FlashExport.jar	2/2/2023 12:16 AM	MicrosoftOpenJD...	221 KB
hsqldb.jar	2/2/2023 12:16 AM	MicrosoftOpenJD...	658 KB
jakarta-activation.jar	2/2/2023 12:16 AM	MicrosoftOpenJD...	75 KB

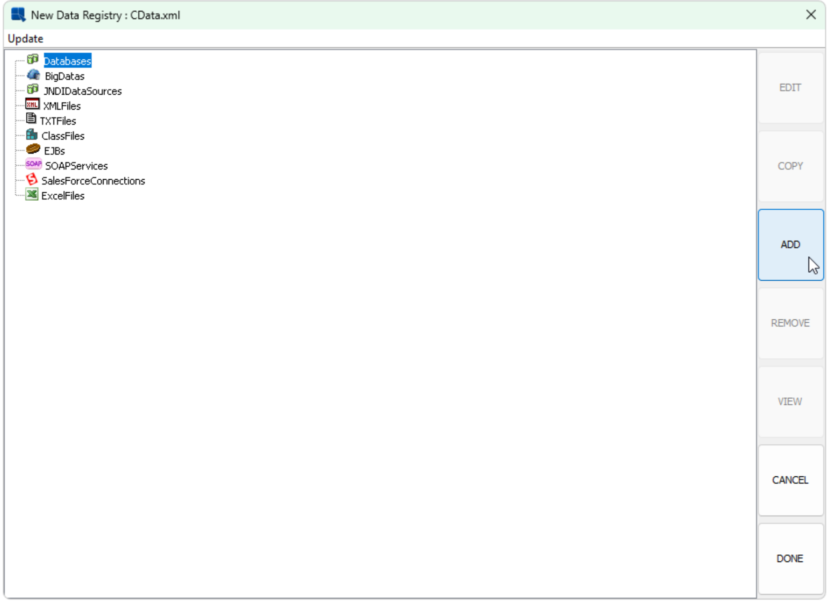
58 items

After you've copied the files, restart The EspressManager (if it is running).

1.3.12.4. Using the CData JDBC drivers in DataSource Manager

Launch the DataSource Manager and open a data registry in it (existing one or a new one).

Select the “Databases” option in the tree-list and press the “ADD” Button.

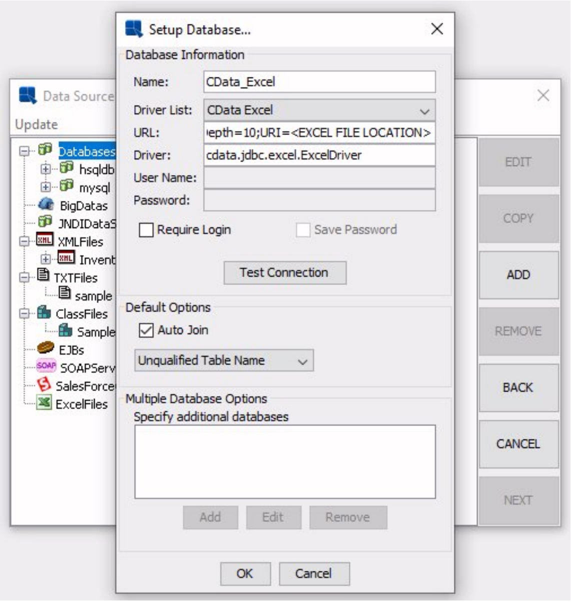


New Data Registry : CData.xml

Update

- Databases
- BigDatas
- JNDIDataSources
- XMLFiles
- TXTFiles
- ClassFiles
- EJBs
- SOAPServices
- SalesForceConnections
- ExcelFiles

In the “Driver List:” drop-down menu, select “CData Excel” (or any other CData JDBC driver you might be installing).



Setup Database...

Database Information

Name: CData_Excel

Driver List: CData Excel

URL: epth=10;URI=<EXCEL FILE LOCATION>

Driver: cdata.jdbc.excel.ExcelDriver

User Name:

Password:

□ Require Login □ Save Password

Default Options

☑ Auto Join

Unqualified Table Name

Multiple Database Options

Specify additional databases

In the “URL:” text field, replace the placeholders (like "EXCEL FILE LOCATION") with real values.

Tip

Alternatively, you can replace the text in the “URL:” text field with the connection string obtained by the CData Test Connection tool described in the previous chapter.

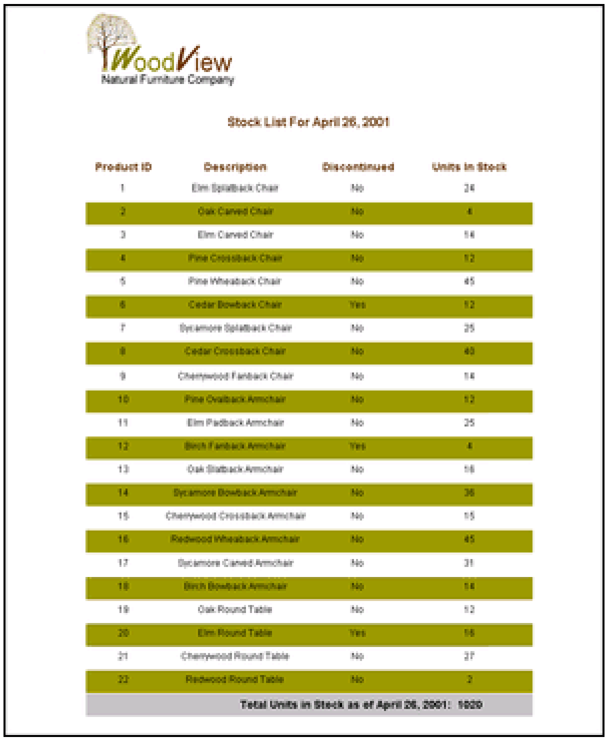
Click OK. You’re done. You can start using Excel files as if they were a database.

1.4. Report Types and Data Mapping

EspressReport supports five basic report types: simple columnar, summary break, crosstab, master & details, and mailing label. Each report type has slightly different mapping options and can be used to generate a different style of report.

1.4.1. Simple Columnar Report

The simple columnar report is the most basic report type supported by EspressReport. It displays columnar data in a single table without any grouping or breaks.



WoodView
Natural Furniture Company

Stock List For April 26, 2001

Product ID	Description	Discontinued	Units In Stock
1	Elm Splatback Chair	No	24
2	Oak Carved Chair	No	4
3	Elm Carved Chair	No	14
4	Pine Crossback Chair	No	12
5	Pine Wheatback Chair	No	45
6	Cedar Bowback Chair	Yes	12
7	Sycamore Splatback Chair	No	25
8	Cedar Crossback Chair	No	40
9	Cherrywood Fanback Chair	No	14
10	Pine Ovalback Armchair	No	12
11	Elm Padback Armchair	No	25
12	Birch Fanback Armchair	Yes	4
13	Oak Slatback Armchair	No	16
14	Sycamore Bowback Armchair	No	36
15	Cherrywood Crossback Armchair	No	15
16	Redwood Wheatback Armchair	No	45
17	Sycamore Carved Armchair	No	31
18	Birch Bowback Armchair	No	14
19	Oak Round Table	No	12
20	Elm Round Table	Yes	16
21	Cherrywood Round Table	No	27
22	Redwood Round Table	No	2

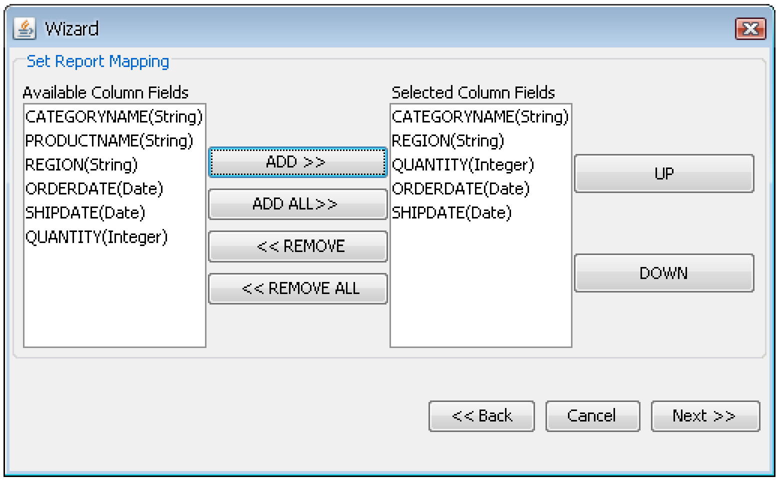
Total Units in Stock as of April 26, 2001: 1020

Simple Columnar Report

1.4.1.1. Data Mapping

The first step in data mapping for this type is to select the columns you want to include in your report from the *Set Report Mapping* window in the Report Wizard. The left side of the window lists all available columns from your query or data file and the right side lists columns that will be added to the report. Click on a column name to select it (**Shift**+**Click** for multiple selections) and press the *ADD* or *REMOVE* buttons to add it or to remove it from the report. The order in which you select columns in this window is the order in which they will appear in your report. You can change the order of the selected columns by highlighting the column you want to move and clicking the *UP* or *DOWN* buttons. Fields can also be moved by clicking and dragging the selected field.

The second step for report mapping is to set the column options. For simple columnar reports there is only one available option. The window displays a table showing all the columns you have selected for the report and all available options. The first field displays column names, the second one displays data type, and the third one displays a check box called *Visible*. Checking or un-checking this box will make the column visible or invisible within the body of the report (All columns are visible by default).

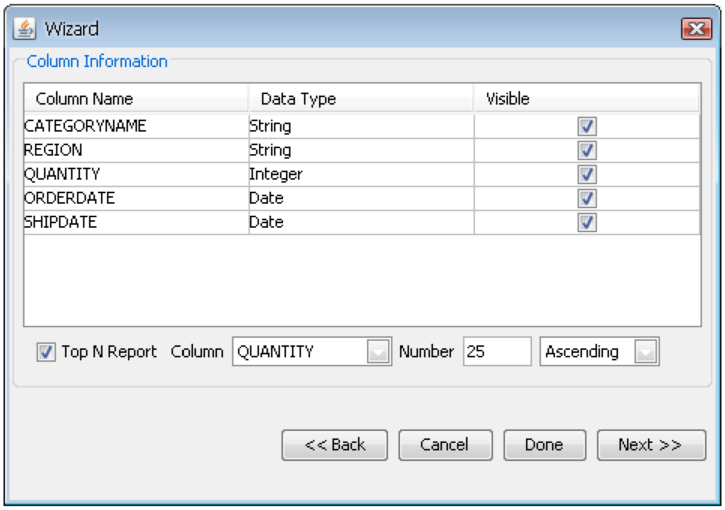


Wizard

Set Report Mapping

Available Column Fields
CATEGORYNAME(String)
PRODUCTNAME(String)
REGION(String)
ORDERDATE(Date)
SHIPDATE(Date)
QUANTITY(Integer)

Selected Column Fields
CATEGORYNAME(String)
REGION(String)
QUANTITY(Integer)
ORDERDATE(Date)
SHIPDATE(Date)



Column Name	Data Type	Visible
CATEGORYNAME	String	☑
REGION	String	☑
QUANTITY	Integer	☑
ORDERDATE	Date	☑
SHIPDATE	Date	☑

☑ Top N Report Column QUANTITY Number 25 Ascending

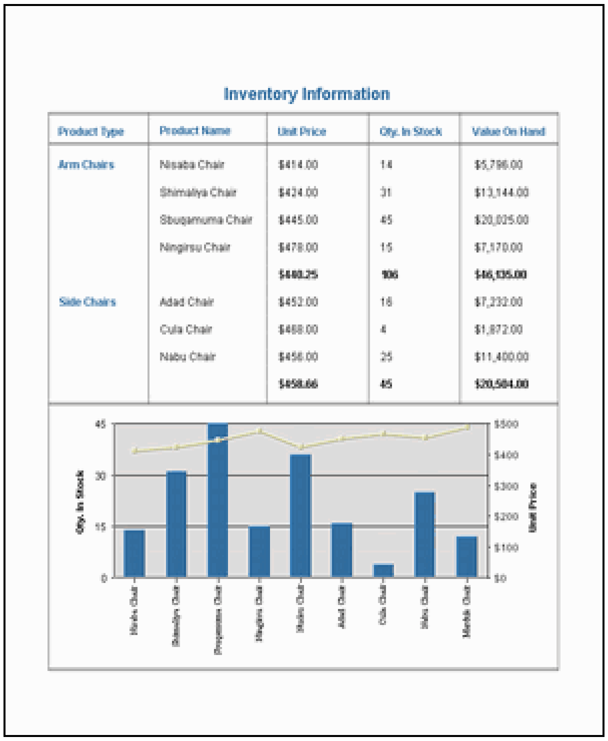
Simple Columnar Report Mapping

1.4.1.1.1. Top N Report

You can also create a Top N report using the columnar format. A Top N report will order and show you the highest set of values based on a particular column in the report. For example, you want to show top five customers based on total sales. To do this, check the *Top N Report* checkbox at the bottom of the data mapping window. You can then specify which column you want to use as the measure (for the above example, it would be total sales column), the number you want to retrieve (i.e. 10 or 20, etc), and whether you want to show the columns in ascending or descending order. The columnar report will then return the number of records specified sorted by the highest value for the specified column.

1.4.2. Summary Break Report

Like the simple columnar report, the summary break report type takes columnar data and presents it in a tabular form. However, unlike the simple columnar report, it allows you to break data into sections and insert summary fields.



Inventory Information

Product Type	Product Name	Unit Price	Qty. In Stock	Value On Hand
Arm Chairs	Nisaba Chair	$414.00	14	$5,796.00
	Shimaliya Chair	$424.00	31	$13,144.00
	Sbugamuma Chair	$445.00	45	$20,025.00
	Ningirsu Chair	$478.00	15	$7,170.00
		$440.25	**106**	**$46,135.00**
Side Chairs	Adad Chair	$452.00	16	$7,232.00
	Cula Chair	$468.00	4	$1,872.00
	Nabu Chair	$456.00	25	$11,400.00
		$458.66	**45**	**$20,504.00**

Summary Break Report

1.4.2.1. Data Mapping

The first step in data mapping for this type is to select the columns you want to include in your report from the `Set Mapping` window in the Report Wizard. The left side of the window lists all available columns from your query or data file and the right side lists the columns that will be displayed in the report. Click on a column name to select it (**SHIFT**+**Click** for multiple selections) and press the *ADD* or *REMOVE* buttons to add it or remove it from the report. The order in which you select columns in this window is the order in which they will appear in your report. You can change the order of the selected columns by highlighting the column you want to move and clicking the *UP* or *DOWN* buttons. Fields can also be moved by clicking and dragging the selected field.

The second step in data mapping is to set column options. There are five column options available for summary break reports. The window displays a table showing all of the columns that you have selected for your report and all available options. The first field displays column names, the second one displays data type, and the third one displays a check box marked `Visible`. Checking or un-checking this box will make the column visible or invisible within the body of the report (All columns are visible by default). The fourth field is the row break field. Checking this box indicates that the report will break and insert column summaries every time the selected column field changes. By default, the first column selected for the report is the break field. The fifth field allows you to select aggregation. If the *Perform Column Aggregation* checkbox at the bottom of the window is checked, the selected aggregation will be performed on the entire column and the report will only contain summarized data. If the box is not checked, the report will display all data and only insert aggregations as summaries after each row break.

The drop-down menu allows you to select the aggregation operation to be performed for that column. The aggregation operations that can be performed are: `none`, `sum`, `maximum`, `minimum`, `count`, `average`, `first`, `last`, `sum of squares`, `variance`, `standard deviation`, and `count distinct`. If you select to perform column aggregation, you must select an aggregation option for each column. Columns selected as `row break` fields cannot be aggregated.

For example, suppose we have the following data table:

Order#	Product	Quantity
12	Chair	2
12	Table	3

Order#	Product	Quantity
14	Cabinet	2
14	Table	5

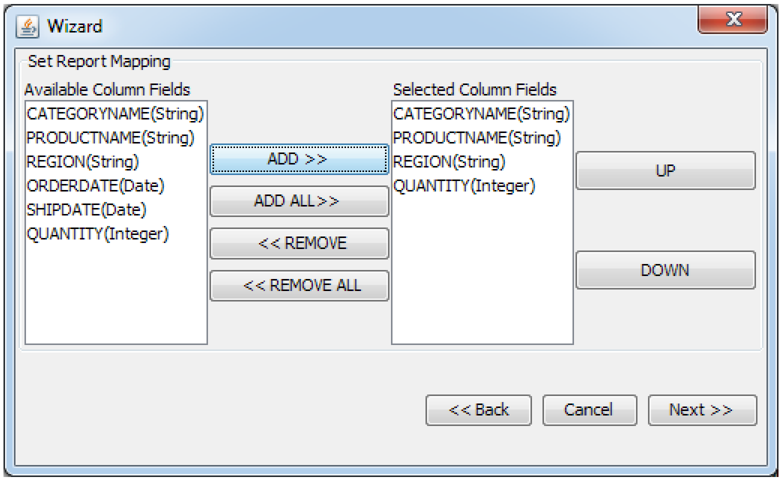
Setting `Order #` as the row break field and the aggregation on `Quantity` to `Sum` without checking *Perform Column Aggregation* will produce the following report:

Order#	Product	Quantity
12	Chair	2
	Table	3
		5
14	Cabinet	2
	Table	5
		7

However, selecting *Perform Column Aggregation* and changing the `Product` aggregation to `Count` will produce the following report:

Order#	Product	Quantity
12	2	5
14	2	7
	2	**5**

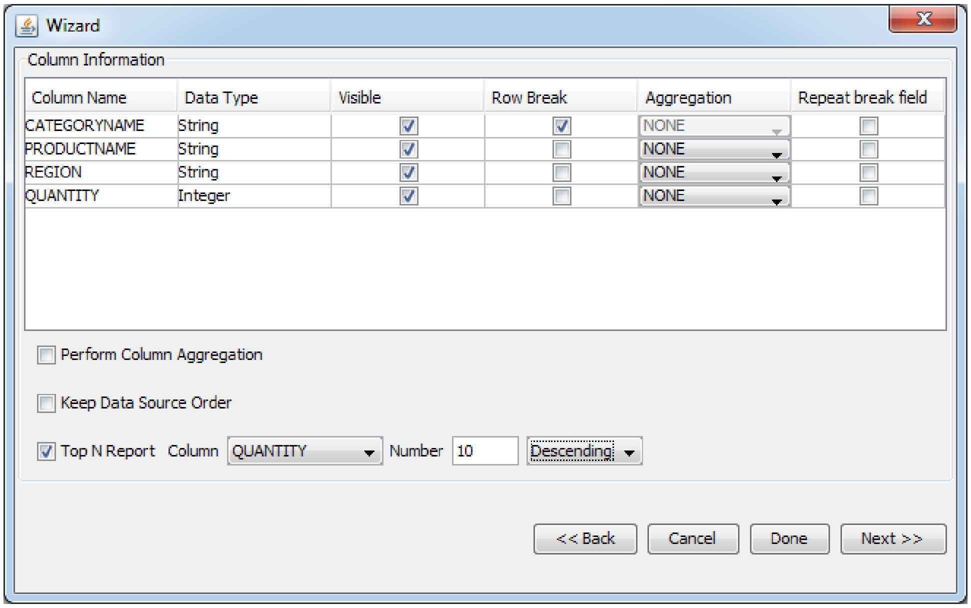
The sixth field allows you to select whether to repeat a break field or not. By default, a row break column will only print each distinct value in the column once (i.e. once for each group). Selecting this option will cause the break fields to repeat for each row of data in the group.



Available Column Fields: CATEGORYNAME(String), PRODUCTNAME(String), REGION(String), ORDERDATE(Date), SHIPDATE(Date), QUANTITY(Integer)

Selected Column Fields: CATEGORYNAME(String), PRODUCTNAME(String), REGION(String), QUANTITY(Integer)

Summary Break Report Mapping



Column Name	Data Type	Visible	Row Break	Aggregation	Repeat break field
CATEGORYNAME	String	☑	☑	NONE	□
PRODUCTNAME	String	☑	□	NONE	□
REGION	String	☑	□	NONE	□
QUANTITY	Integer	☑	□	NONE	□

□ Perform Column Aggregation

□ Keep Data Source Order

☑ Top N Report Column QUANTITY Number 10 Descending

Summary Break Report Mapping

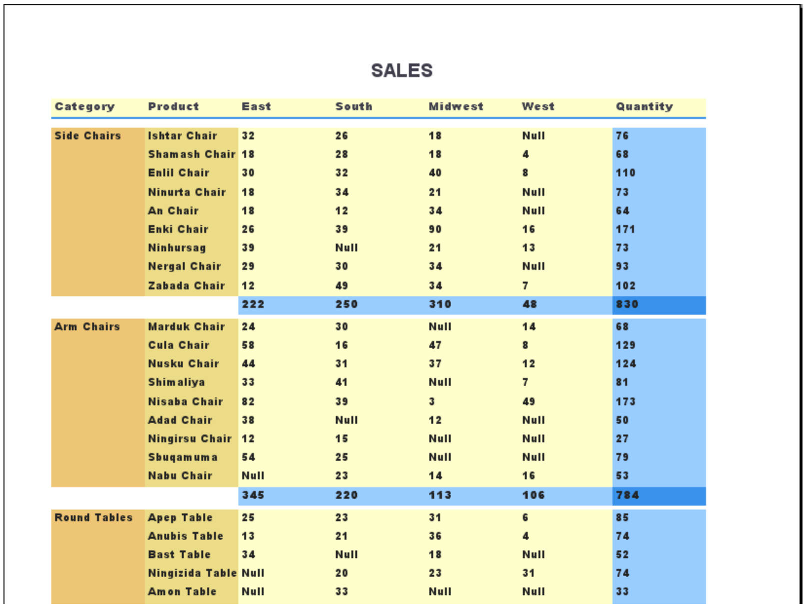
If you want to keep the data ordering specified in the data source, select the *Keep Data Source Order* option. To learn more about this feature, see Section 1.4.9 - Keep Data Source Order.

1.4.2.1.1. Top N Report

You can also create a Top N report using the summary break format. A Top N report will order and show you to highest set of values based on a particular column in the report. For summary break reports, you can also specify to show the highest values for each group. For example, you may want to show the top five customers in each region based on total sales. To do this, check the *Top N Report* checkbox at the bottom of the data mapping window. You can then specify which column you want to use as the measure (for the above example, it would be total sales column), the number that you want to retrieve (i.e. 10 or 20, etc), and whether you want to display the columns in ascending or descending order. The summary break report will then return the number of records specified (or number of records specified for each row break) sorted by the highest value for the specified column.

1.4.3. Crosstab Report

A crosstab report is a report format that shows and summarizes columnar data in a matrix-like form. Crosstab reports often resemble spreadsheets. Both rows and columns are summarized, allowing multi-dimensional data to be displayed in a two-dimensional format.



SALES

Category	Product	East	South	Midwest	West	Quantity
Side Chairs	Ishtar Chair	32	26	18	Null	76
	Shamash Chair	18	28	18	4	68
	Enlil Chair	30	32	40	8	110
	Ninurta Chair	18	34	21	Null	73
	An Chair	18	12	34	Null	64
	Enki Chair	26	39	90	16	171
	Ninhursag	39	Null	21	13	73
	Nergal Chair	29	30	34	Null	93
	Zabada Chair	12	49	34	7	102
		222	250	310	48	830
Arm Chairs	Marduk Chair	24	30	Null	14	68
	Cula Chair	58	16	47	8	129
	Nusku Chair	44	31	37	12	124
	Shimaliya	33	41	Null	7	81
	Nisaba Chair	82	39	3	49	173
	Adad Chair	38	Null	12	Null	50
	Ningirsu Chair	12	15	Null	Null	27
	Sbuqamuma	54	25	Null	Null	79
	Nabu Chair	Null	23	14	16	53
		345	220	113	106	784
Round Tables	Apep Table	25	23	31	6	85
	Anubis Table	13	21	36	4	74
	Bast Table	34	Null	18	Null	52
	Ningizida Table	Null	20	23	31	74
	Amon Table	Null	33	Null	Null	33

Crosstab Report

The actual report is only constructed during running or preview. Since the crosstab table is essentially constructed from scratch every time the report is run, it is easier to create a smoothly collapsing and expanding crosstab table.

1.4.3.1. Data Mapping

Report mapping for crosstab reports is more complicated than other report types and may require some planning to be able to execute it correctly.

The first step in data mapping for this type is to select the columns you want to include in your report from the *Set Report Mapping* window in the Wizard. The left side of the window lists all available columns from your query or data file and the right side lists the columns that will be displayed in the report. Click on a column name to select it (**Ctrl+click** for multiple selections) and press the *ADD* or *REMOVE* buttons to add it or remove it from the report. The order in which you select columns in this window is the order in which they will appear in your report. Once you select the columns, click the *Next* button.

The second step in data mapping is to set column options.

The top of the column options window displays a table showing all of the columns that you have selected for your report and all available options.

- The first field displays the *Column Name*.
- The second field displays the *Data Type*.
- The third field displays a checkbox marked *Visible*. Checking or un-checking this box will make the column visible or invisible within the body of the report.
- The fourth field is the *Row Break* field. Selecting a column as the row break field will cause the report to insert a new crosstab row for each unique entry in the selected column.
- The fifth field is the *Column Break* field. Selecting a column as a column break field will cause the report to create a new column for each unique entry in the database.
- The sixth field is the *Column Break Value* field. Columns that you select as column break values become the fields that are summarized in the report.
- The seventh field allows you to select column *Aggregation*. A drop-down menu allows you to select the aggregation to be performed for that column. The aggregation operations that can be performed are: *sum*, *maximum*,

minimum, *count*, *average*, *first*, *last*, *sum of squares*, *variance*, *standard deviation*, and *count distinct*. Columns that have been selected as *Row Break* or *Column Break* fields cannot be aggregated.

- The eighth field *Order* allows you to specify ordering for the column break column. This order will determine how the crosstab columns are drawn from left to right. Ordering options are *not sorted*, *ascending*, and *descending*.

> ### Note
>
> The aggregation that will be performed depends on the aggregation of the *Column Break Value* column(s). For example: if you use the *MAX* aggregation, the greatest value from the row will be displayed in the *Grand Total* column.

Keep Data Source Order: If you want to keep the data ordering specified in the data source, select this option. To learn more about this feature, see Section 1.4.9 - Keep Data Source Order.

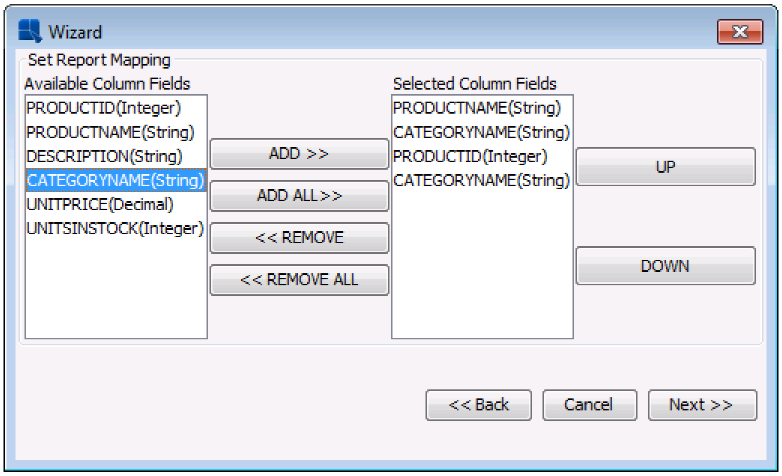
Show Column Aggregation Grand Total: If this option is enabled, a column will be added into the report. Each row of the column will aggregate all values from the corresponding row into a single value.

Column Aggregation Subtotal: Subtotal column is very similar to the *Grand Total* column, except it doesn't aggregate the whole row, but each one of the *Column Break* group. This option allows you to disable the subtotal column or set its position (right or left to the *Column Break* group).

Row Aggregation: This is also very similar to the *Grand Total* column. The main difference is that this option doesn't aggregate rows, but columns. Also, it doesn't add a column to the report, but it adds a row instead. Use this options to disable the *Row Aggregation* row, or to adjust it's position (on top of the table or below the table).

Align Column Break Value: If you select more than one *Column Break Value* columns, use this option to choose whether the values will be aligned in rows or columns.

For example, assume we have following crosstab mapping:



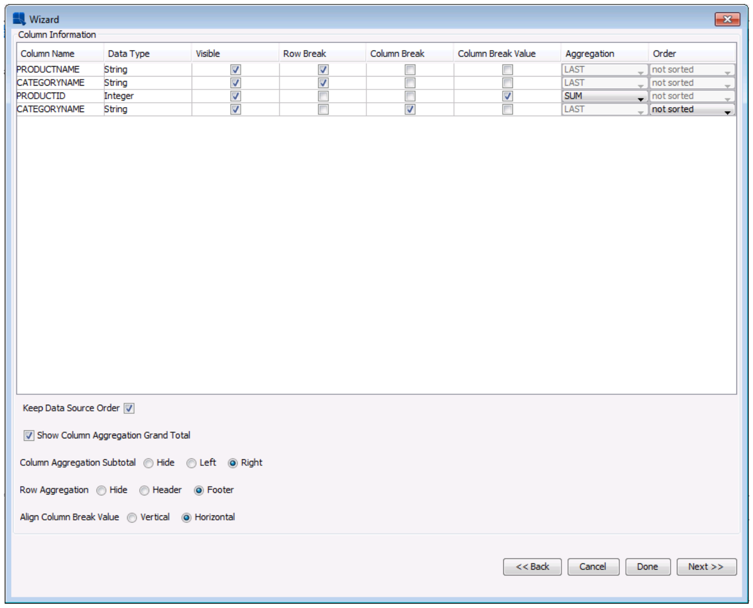
Set Report Mapping

Available Column Fields

PRODUCTID(Integer)
PRODUCTNAME(String)
DESCRIPTION(String)
CATEGORYNAME(String)
UNITPRICE(Decimal)
UNITSINSTOCK(Integer)

Selected Column Fields

PRODUCTNAME(String)
CATEGORYNAME(String)
PRODUCTID(Integer)
CATEGORYNAME(String)



Column Information

Column Name	Data Type	Visible	Row Break	Column Break	Column Break Value	Aggregation	Order
PRODUCTNAME	String	☑	☑	□	□	LAST	not sorted
CATEGORYNAME	String	☑	☑	□	□	LAST	not sorted
PRODUCTID	Integer	☑	□	□	☑	SUM	not sorted
CATEGORYNAME	String	☑	□	☑	□	LAST	not sorted

Keep Data Source Order ☑

☑ Show Column Aggregation Grand Total

Column Aggregation Subtotal ○ Hide ○ Left ● Right

Row Aggregation ○ Hide ○ Header ● Footer

Align Column Break Value ○ Vertical ● Horizontal

Crosstab Fixed-Field Mapping

Setting `CATEGORYNAME` and `PRODUCTNAME` as **`Row Break`** fields, `PRODUCTID` as a **`Column Break Value`** field with an aggregation of **`SUM`**, and `CATEGORYNAME` as a **`Column Break`** field with the *Show Column Aggregation Grand Total* option enabled will produce a report from the beginning of the *Crosstab Report* section.

You can easily change the crosstab settings using the *Change Data Mapping* option under the *Data* menu. This will open the crosstab report mapping dialog again. To learn more about this feature, see Section 1.4.8 - Change Data Mapping.

The following table shows various formatting in crosstab reports:

	Column Aggregation Subtotal	**Row Aggregation**	**Align Column Break Value**	**Template**
Report A [https://data.quadbase.com/Docs71/help/manual/code/export/Fixed_field_Crosstab_A.pdf]	Right	Footer	Horizontal	Template A [https://data.quadbase.com/Docs71/help/manual/code/templates/Fixed_field_Crosstab_A.pak]
Report B [https://data.quadbase.com/Docs71/help/manual/code/export/Fixed_field_Crosstab_B.pdf]	Right	Header	Horizontal	Template B [https://data.quadbase.com/Docs71/help/manual/code/templates/Fixed_field_Crosstab_B.pak]
Report C [https://data.quadbase.com/Docs71/help/manual/code/ex-	Left	Footer	Horizontal	Template C [https://data.quadbase.com/Docs71/help/manual/code/tem-

	Column Aggregation Subtotal	Row Aggregation	Align Column Break Value	Template
port/Fixed_field_Crosstab_C.pdf]				plates/Fixed_field_Crosstab_C.pak]
Report D [https://data.quadbase.com/Docs71/help/manual/code/export/Fixed_field_Crosstab_D.pdf]	Left	Header	Vertical	Template D [https://data.quadbase.com/Docs71/help/manual/code/templates/Fixed_field_Crosstab_D.pak]
Report E [https://data.quadbase.com/Docs71/help/manual/code/export/Fixed_field_Crosstab_E.pdf]	Left	None	Vertical	Template E [https://data.quadbase.com/Docs71/help/manual/code/templates/Fixed_field_Crosstab_E.pak]

1.4.3.2. Transposing Data

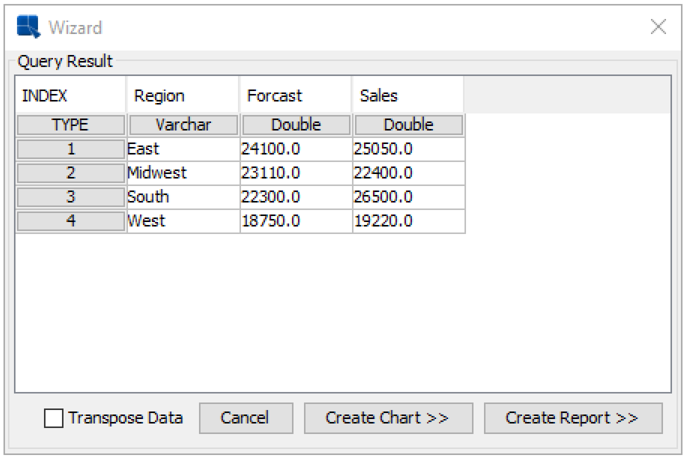
The crosstab report also provides a unique feature that allows users to create a transposed report. Data transposition allows users to create a report presentation where the input data is essentially rotated by 90 degrees. Columns become rows and rows become columns. For example, you have a following set of data:

Region	Forcast	Sales
East	24100	25050
Midwest	23110	22400
South	22300	26500
West	18750	19220

The data transposition feature can be used to create a report that will look like this:

	East	Midwest	South	West
Forcast	24100	23110	22300	18750
Sales	25050	22400	26500	19220

To transpose the data, click the *Transpose Data* option at the bottom of the result screen that first appears when you select a data source for the report. The following dialog shows the initial result set from the sample data in the example above.

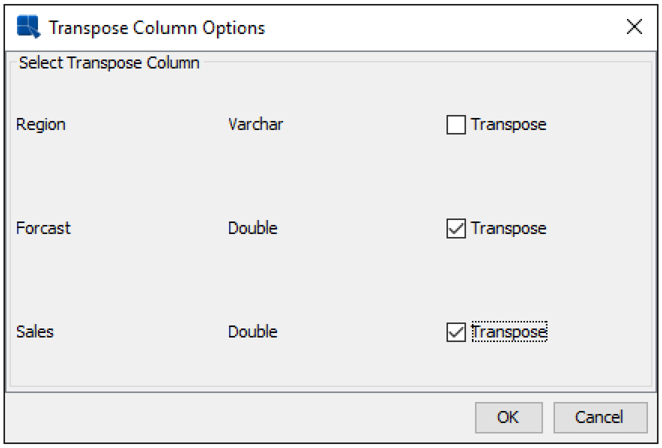


INDEX	Region	Forcast	Sales
TYPE	Varchar	Double	Double
1	East	24100.0	25050.0
2	Midwest	23110.0	22400.0
3	South	22300.0	26500.0
4	West	18750.0	19220.0

□ Transpose Data

Data Table Dialog

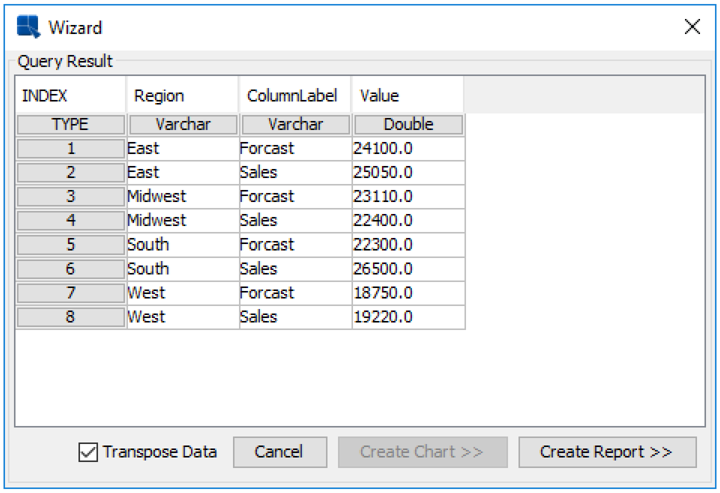
When you check the option at the bottom, a new dialog will appear allowing you to select the columns that you want to transpose. In this example, you would select the `Forecast` and `Sales` columns. Note that in order to perform transposition, the selected columns must have the same data type.



Region	Varchar	□ Transpose
Forcast	Double	☑ Transpose
Sales	Double	☑ Transpose

Select Transpose Dialog

After making your selections, click the *OK* button to apply the changes. You will see the transposition in the data table dialog. The Sales and Forecast headers are transposed into a column called `ColumnLabel` and their values are merged into a column called `Value`. Now you can map this result set using a crosstab report where `ColumnLabel` is the Row Break, `Region` is the Column Break, and `Value` is the Column Break Value to create the final transposed report layout.

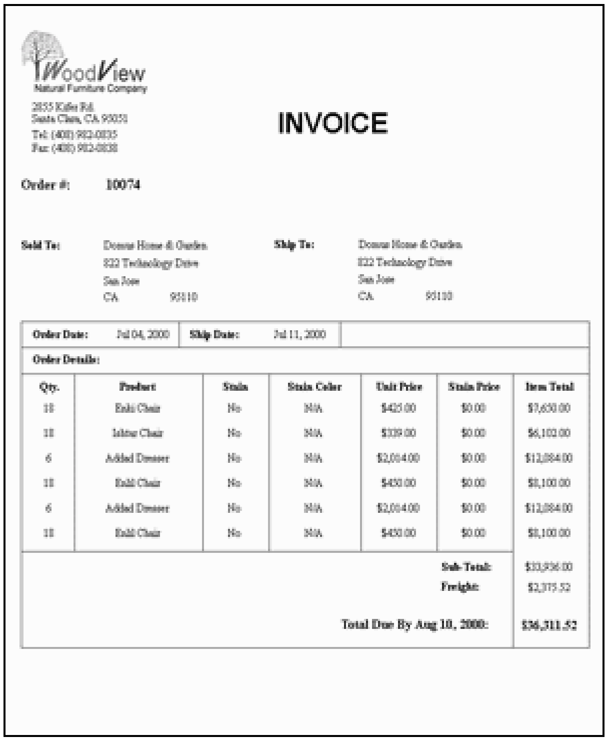


INDEX	Region	ColumnLabel	Value
TYPE	Varchar	Varchar	Double
1	East	Forcast	24100.0
2	East	Sales	25050.0
3	Midwest	Forcast	23110.0
4	Midwest	Sales	22400.0
5	South	Forcast	22300.0
6	South	Sales	26500.0
7	West	Forcast	18750.0
8	West	Sales	19220.0

Data After Transposition

1.4.4. Master & Details Report

A Master & Details report is a set of tabular data that is grouped according to a master field. This report type is most commonly used when you have fields in your data table that have a one to many relationship. A good example of this is an invoice. For each order number in a database there will be customer information and several items with pricing information. A Master & Details report would be used to group the information according to order number.


WoodView
Natural Furniture Company
2855 Kifer Rd.
Santa Clara, CA 95051
Tel: (408) 982-0835
Fax: (408) 982-0838

INVOICE

Order #: 10074

Sold To: Donnas Home & Garden
822 Technology Drive
San Jose
CA 95110

Ship To: Donnas Home & Garden
822 Technology Drive
San Jose
CA 95110

Order Date: Jul 04, 2000 Ship Date: Jul 11, 2000

Order Details:

Qty.	Product	Stain	Stain Color	Unit Price	Stain Price	Item Total
18	Enki Chair	No	N/A	$425.00	$0.00	$7,650.00
18	Ishtar Chair	No	N/A	$339.00	$0.00	$6,102.00
6	Akkad Dresser	No	N/A	$2,014.00	$0.00	$12,084.00
18	Enlil Chair	No	N/A	$450.00	$0.00	$8,100.00
6	Akkad Dresser	No	N/A	$2,014.00	$0.00	$12,084.00
18	Enlil Chair	No	N/A	$450.00	$0.00	$8,100.00

Sub-Total:	$53,936.00
Freight:	$2,375.52
Total Due By Aug 10, 2000:	$56,311.52

Master & Details Report

1.4.4.1. Data Mapping

The first step in data mapping for this type is to select the columns you want to include in your report from the *Set Mapping Window* in the Report Wizard. The left side of the window lists all available columns from your query or data file and the right side lists columns that will be displayed in the report. Click on a column name to select it (**Ctrl**+**click** for multiple selections) and press the *ADD* or *REMOVE* buttons to add it or remove it from the report. The order in which you select columns in this window is the order in which they will appear in your report.

The second step in data mapping is to set the column options. There are three column options available for Master & Details reports. There is a drop-down menu at the bottom of the column options window. It is labeled `primary key`. The drop-down menu contains all of the columns that you have selected for the report. Selecting a column as the `primary key` will group the report according to that column. A new group will be created every time the value in the selected column changes.

The top of the column options window displays a table showing all of the columns that you have selected for your report and the available options. The first field displays column names, the second one displays data type, and the third one displays a checkbox marked *Visible*. Checking or un-checking this box will make the column visible or invisible within the body of the report. The fourth field is the `Master field` field. Selecting a column as a Master field will place the column value in the column header instead of the data section of the report.

A checkbox at the bottom of the data mapping window allows you to change the layout of the report. Checking the *Side-By-Side Layout* box will arrange the report so that the master section is displayed next to the details section rather than above it.

For example, suppose we have the following data table:

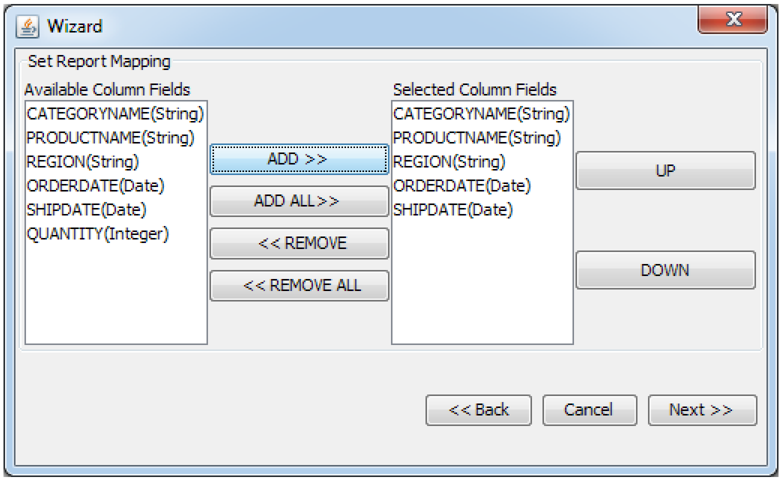
Order #	Customer Name	Product	Unit Price	Quantity
12	Paul Campbell	Chair	$24.95	4
12	Paul Campbell	Table	$127.50	1
14	Sally Hayes	Cabinet	$227.25	2
14	Sally Hayes	Chair	$24.95	2
14	Sally Hayes	Table	$127.50	1

Setting `Order #` as the **`primary key`** and `Customer Name` as a **`Master field`** without using side-by-side layout will result in the following report:

Order #	**12**	
Customer Name	**Paul Campbell**	
Product	**Unit Price**	**Quantity**
Chair	$24.95	4
Table	$127.50	1
Order #	**14**	
Customer Name	**Sally Hayes**	
Product	**Unit Price**	**Quantity**
Cabinet	$227.25	2
Chair	$24.95	2
Table	$127.50	1

However, the same report with side-by-side layout would look like this:

		Product	UnitPrice	Quantity
Order #:	**12**	Chair	$24.95	4
Customer Name:	**Paul Campbell**	Table	$127.50	1
Order#:	**14**	Cabinet	$227.25	2
Customer Name:	**Sally Hayes**	Chair	$24.95	2
		Table	$127.50	1



Set Report Mapping

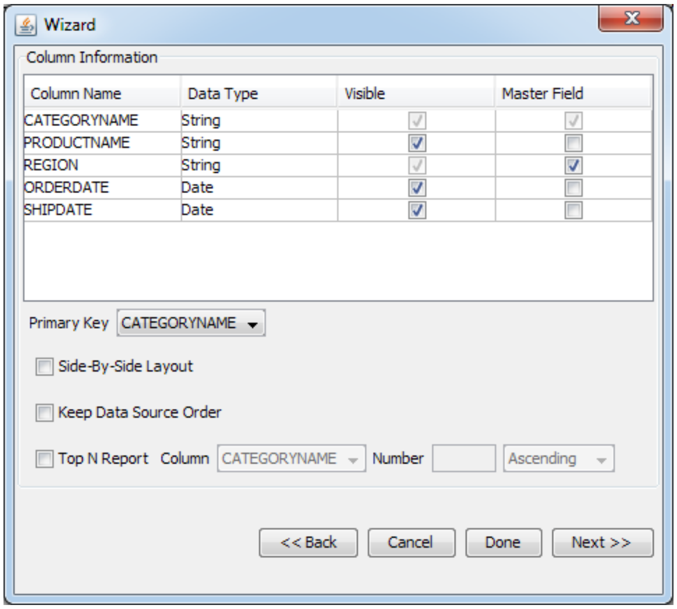
Available Column Fields

CATEGORYNAME(String)
PRODUCTNAME(String)
REGION(String)
ORDERDATE(Date)
SHIPDATE(Date)
QUANTITY(Integer)

Selected Column Fields

CATEGORYNAME(String)
PRODUCTNAME(String)
REGION(String)
ORDERDATE(Date)
SHIPDATE(Date)

Master & Details Report Mapping



Column Information

Column Name	Data Type	Visible	Master Field
CATEGORYNAME	String	☑	☑
PRODUCTNAME	String	☑	□
REGION	String	☑	☑
ORDERDATE	Date	☑	□
SHIPDATE	Date	☑	□

Primary Key CATEGORYNAME

□ Side-By-Side Layout

□ Keep Data Source Order

□ Top N Report Column CATEGORYNAME Number Ascending

Master & Details Report Mapping

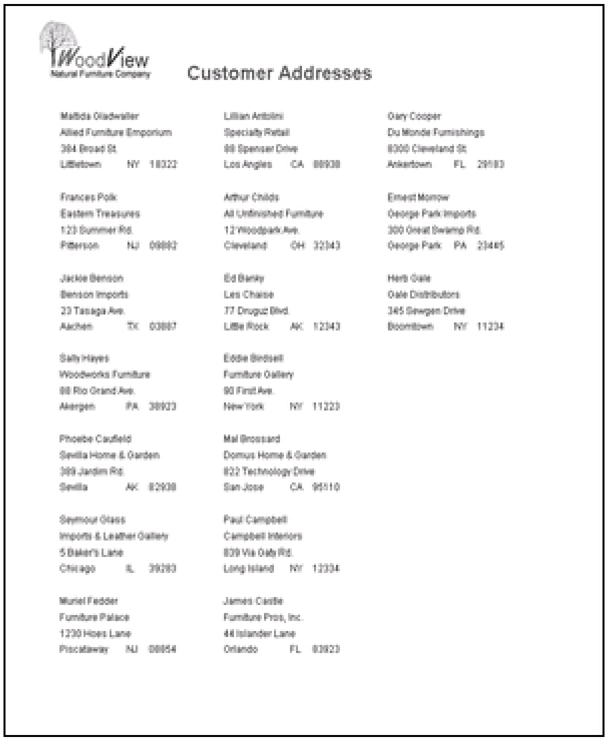
If you want to keep the data ordering specified in the data source, select the *Keep Data Source Order* option. To learn more about this feature, see Section 1.4.9 - Keep Data Source Order.

1.4.4.1.1. Top N Report

You can also create a Top N report using the Master & Details format. A Top N report will order and show you the highest set of values based on a particular column in the report. For Master & Details reports you can also specify to show the highest values for each group. For example, you may want to show the top five customers in each region based on total sales. To do this, check the *Top N Report* checkbox at the bottom of the data mapping window. You can then specify which column you want to use as the measure (for the above example it would be total sales column), the number that you want to retrieve (i.e. 10 or 20, etc), and whether you want to display the columns in ascending or descending order. The master & details report will then return the number of records specified (or number of records specified for each row break) sorted by the highest value for the specified column.

1.4.5. Mailing Label Report

A mailing label report is similar to a simple columnar report; however, it pre-arranges data in a way that allows you to create mailing labels. Data is arranged within the data table vertically and is wrapped automatically, making it easy to create a report showing an address list for mailing labels.



Customer Addresses

Matilda Gladwaller
Allied Furniture Emporium
394 Broad St.
Littletown NY 18322

Frances Polk
Eastern Treasures
123 Summer Rd.
Patterson NJ 09882

Jackie Benson
Benson Imports
23 Tasaga Ave.
Aachen TX 03887

Sally Hayes
Woodworks Furniture
88 Rio Grand Ave.
Akergen PA 38923

Phoebe Caulfield
Sevilla Home & Garden
389 Jardim Rd.
Sevilla AK 82938

Seymour Glass
Imports & Leather Gallery
5 Baker's Lane
Chicago IL 39283

Muriel Fedder
Furniture Palace
1230 Hoes Lane
Piscataway NJ 08854

Lillian Antolini
Specialty Retail
88 Spenser Drive
Los Angles CA 88938

Arthur Childs
All Unfinished Furniture
12 Woodpark Ave.
Cleveland OH 32343

Ed Banky
Les Chaise
77 Druguz Blvd.
Little Rock AK 12343

Eddie Birdsell
Furniture Gallery
90 First Ave.
New York NY 11223

Mal Brossard
Domus Home & Garden
822 Technology Drive
San Jose CA 95110

Paul Campbell
Campbell Interiors
839 Via Gaty Rd.
Long Island NY 12334

James Castle
Furniture Pros, Inc.
44 Islander Lane
Orlando FL 83923

Gary Cooper
Du Monde Furnishings
8300 Cleveland St.
Ankertown FL 29183

Ernest Morrow
George Park Imports
300 Great Swamp Rd.
George Park PA 23445

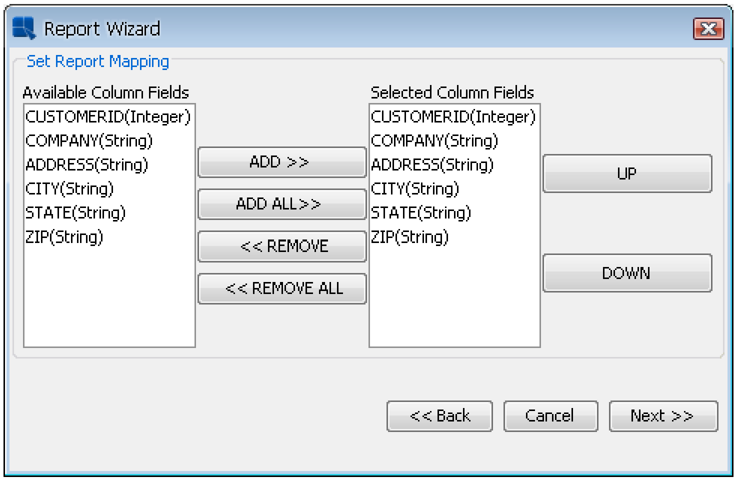
Herb Gale
Gale Distributors
345 Sewgen Drive
Boomtown NY 11234

Mailing Label Report

1.4.5.1. Data Mapping

Data Mapping for this type is exactly the same as for simple columnar report. The first step is to select the columns you want to include in your report from the *Set Report Mapping* window in the Report Wizard. The left side of the window lists all available columns from your query or data file and the right side lists columns that will be displayed in the report. Click on a column name to select it (**SHIFT+Click** for multiple selections) and press the *ADD* or *REMOVE* buttons to add it or remove it from the report. The order in which you select columns in this window is the order in which they will appear in your report.

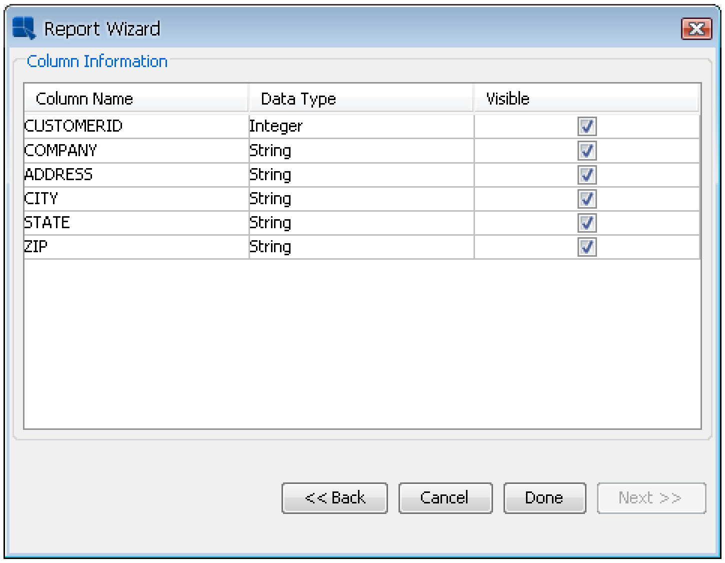
The second step for data mapping is to set the column options. For mailing label reports there is only one available option. The window displays a table showing all of the columns you have selected for the report and all available options. The first field displays column names, the second one displays data type, and the third one displays a checkbox marked *visible*. Checking or un-checking this box will make the column visible or invisible within the body of the report (all columns are visible by default).



Available Column Fields
CUSTOMERID(Integer)
COMPANY(String)
ADDRESS(String)
CITY(String)
STATE(String)
ZIP(String)

Selected Column Fields
CUSTOMERID(Integer)
COMPANY(String)
ADDRESS(String)
CITY(String)
STATE(String)
ZIP(String)

Mailing Label Report Mapping



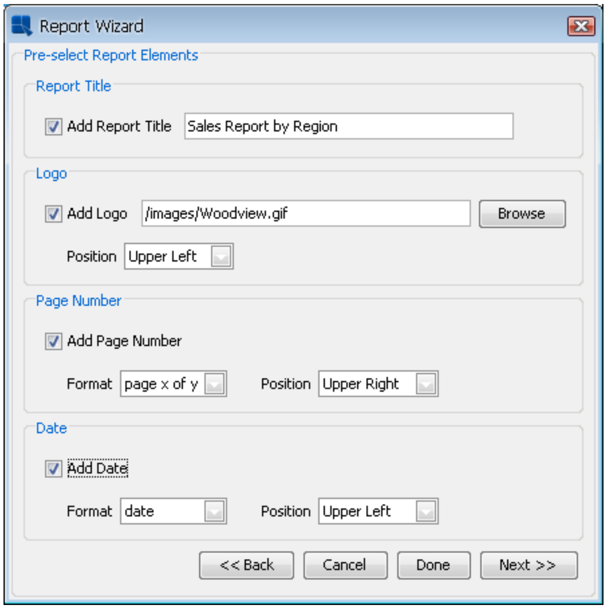
Column Name	Data Type	Visible
CUSTOMERID	Integer	☑
COMPANY	String	☑
ADDRESS	String	☑
CITY	String	☑
STATE	String	☑
ZIP	String	☑

Mailing Label Report Mapping

1.4.6. Additional Formatting Options

After you finish specifying mapping options, you can dismiss the Wizard and begin editing/modifying the report. To exit the Wizard, click the *Done* button in the last data mapping window. This will take you back to the main designer window where a blank (unformatted) report will be generated based on your mapping specifications.

Instead of generating an unformatted report, EspressReport also provides several additional options in the Wizard that allows you to automatically place some elements in the report, as well as provide a default style for the report elements. To continue on in the Wizard, click the *Next* button in the last data mapping window. This will bring up a dialog prompting you to add several elements to the report.



☑ Add Report Title: Sales Report by Region

☑ Add Logo: /images/Woodview.gif

Position: Upper Left

☑ Add Page Number

Format: page x of y

Position: Upper Right

☑ Add Date

Format: date

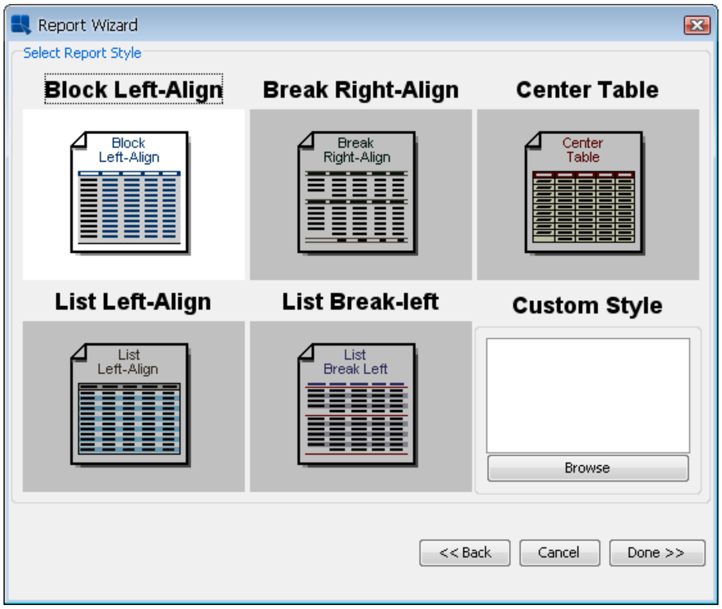
Position: Upper Left

Add Elements Dialog

This dialog allows you to specify a report title and a logo for the report header. If you select to add a report title, you will be able to type in the title text. If you specify to add a logo, you can specify the location of the image file either using a file or URL path. You can also specify whether to place the logo in the upper right or upper left corner of the page.

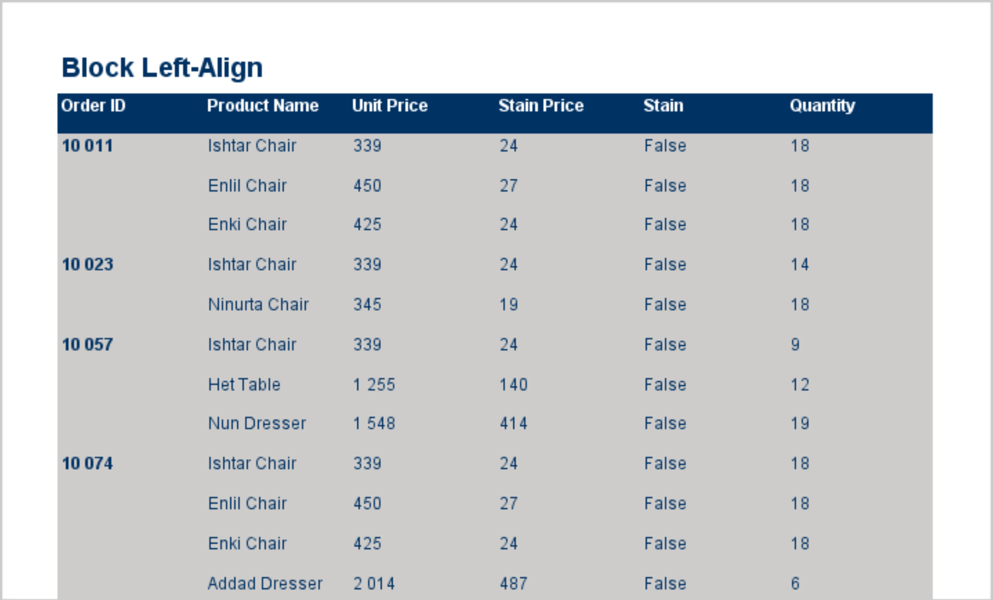
You can also specify to add page numbers and date to the page headers and footers. You can select format and position of the elements either in the page header or footer and align them to the right, left, or center of the page.

After you finish specifying the elements, you can again click the *Done* button to dismiss the Wizard and begin editing/modifying the report with the added elements. If you click *Next*, you will be taken back to the last dialog in the Report Wizard. This dialog allows you to select a style for the report.



Report Style Dialog

From here you can pick a pre-defined style or a custom style you want to use for the report. Selecting a style will apply formats to the report elements, as well as change the default attributes of new report elements. Below are examples of each of the pre-defined report styles when applied to the same summary break report:

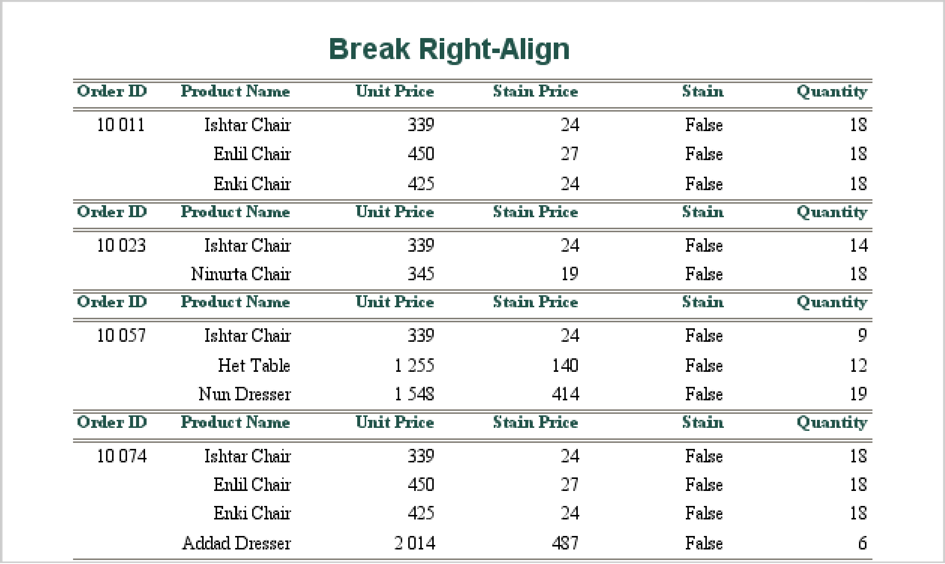


Block Left-Align

Order ID	Product Name	Unit Price	Stain Price	Stain	Quantity
10 011	Ishtar Chair	339	24	False	18
	Enlil Chair	450	27	False	18
	Enki Chair	425	24	False	18
10 023	Ishtar Chair	339	24	False	14
	Ninurta Chair	345	19	False	18
10 057	Ishtar Chair	339	24	False	9
	Het Table	1 255	140	False	12
	Nun Dresser	1 548	414	False	19
10 074	Ishtar Chair	339	24	False	18
	Enlil Chair	450	27	False	18
	Enki Chair	425	24	False	18
	Addad Dresser	2 014	487	False	6

Block Left-Align

This style sets text alignment to the left and draws the report on a gray background with blue headers.

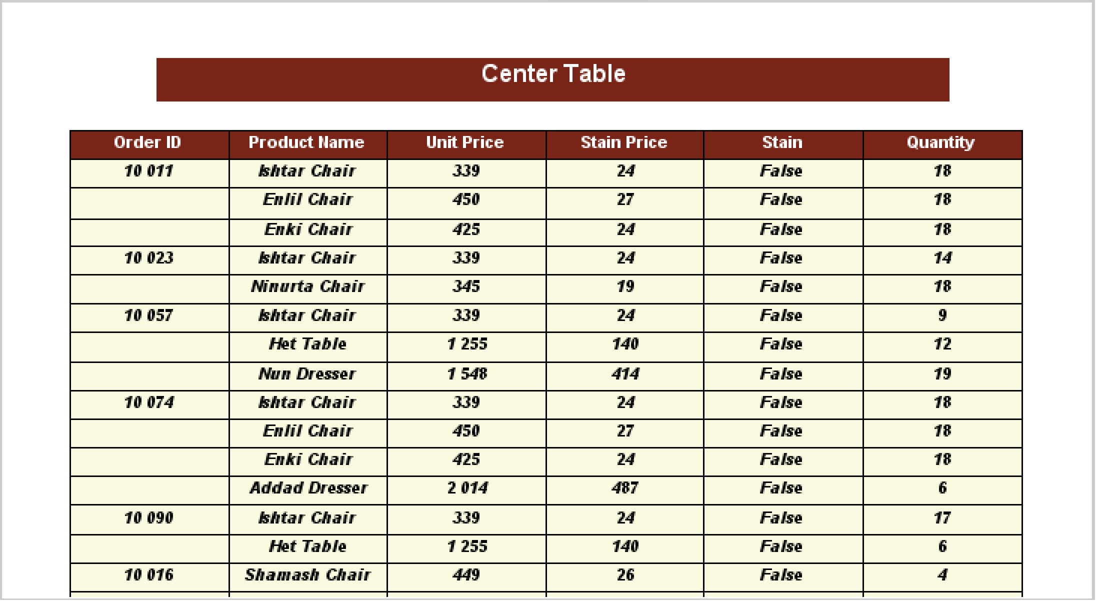


Break Right-Align

Order ID	Product Name	Unit Price	Stain Price	Stain	Quantity
10 011	Ishtar Chair	339	24	False	18
	Enlil Chair	450	27	False	18
	Enki Chair	425	24	False	18
Order ID	Product Name	Unit Price	Stain Price	Stain	Quantity
10 023	Ishtar Chair	339	24	False	14
	Ninurta Chair	345	19	False	18
Order ID	Product Name	Unit Price	Stain Price	Stain	Quantity
10 057	Ishtar Chair	339	24	False	9
	Het Table	1 255	140	False	12
	Nun Dresser	1 548	414	False	19
Order ID	Product Name	Unit Price	Stain Price	Stain	Quantity
10 074	Ishtar Chair	339	24	False	18
	Enlil Chair	450	27	False	18
	Enki Chair	425	24	False	18
	Addad Dresser	2 014	487	False	6

Break Right-Align

This style sets text alignment to the right and draws lines to demarcate the headers.

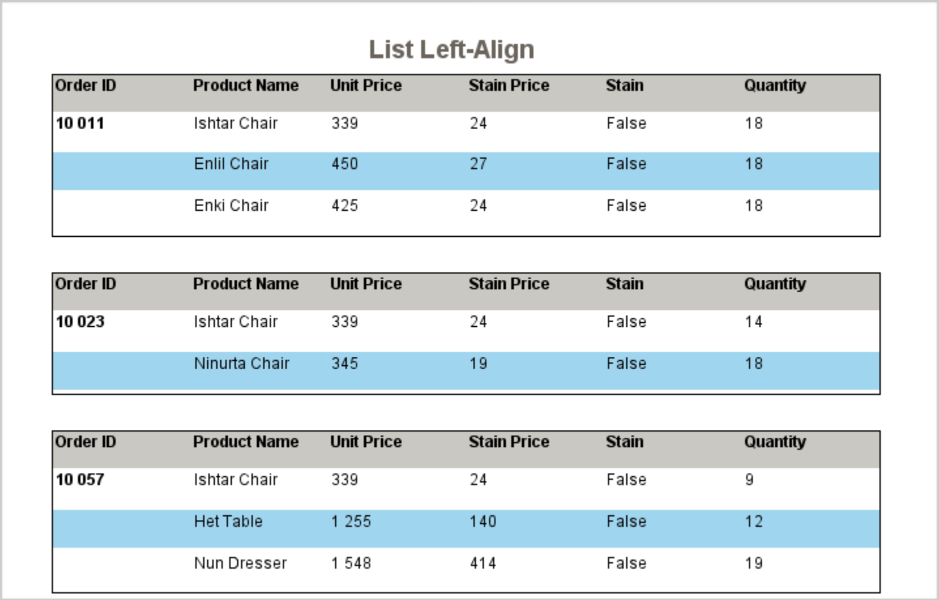


Center Table

Order ID	Product Name	Unit Price	Stain Price	Stain	Quantity
10 011	Ishtar Chair	339	24	False	18
	Enlil Chair	450	27	False	18
	Enki Chair	425	24	False	18
10 023	Ishtar Chair	339	24	False	14
	Ninurta Chair	345	19	False	18
10 057	Ishtar Chair	339	24	False	9
	Het Table	1 255	140	False	12
	Nun Dresser	1 548	414	False	19
10 074	Ishtar Chair	339	24	False	18
	Enlil Chair	450	27	False	18
	Enki Chair	425	24	False	18
	Addad Dresser	2 014	487	False	6
10 090	Ishtar Chair	339	24	False	17
	Het Table	1 255	140	False	6
10 016	Shamash Chair	449	26	False	4

Center Table

This style centers text and draws table borders around the report cells.



List Left-Align

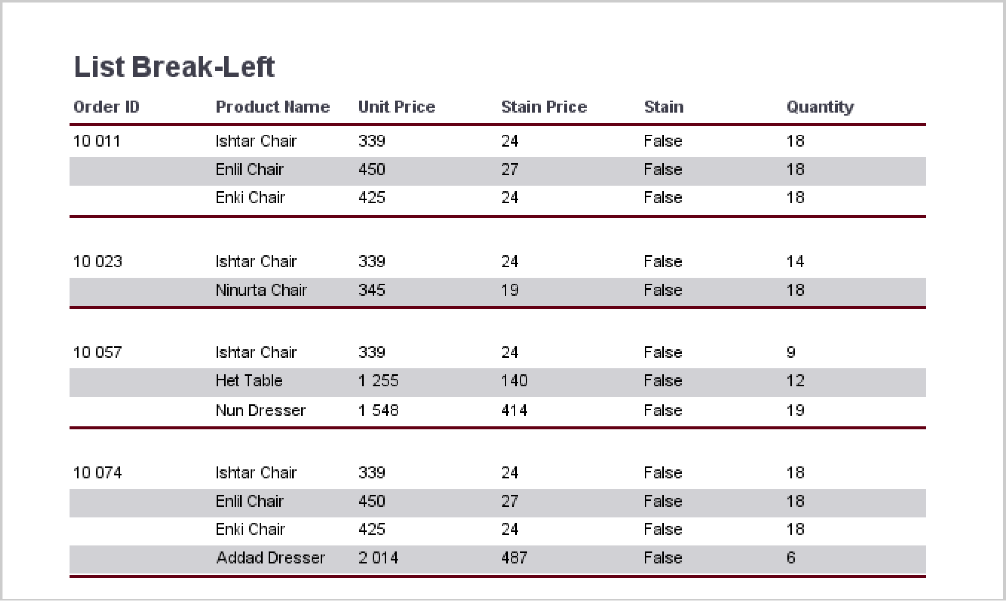
Order ID	Product Name	Unit Price	Stain Price	Stain	Quantity
10 011	Ishtar Chair	339	24	False	18
	Enlil Chair	450	27	False	18
	Enki Chair	425	24	False	18

Order ID	Product Name	Unit Price	Stain Price	Stain	Quantity
10 023	Ishtar Chair	339	24	False	14
	Ninurta Chair	345	19	False	18

Order ID	Product Name	Unit Price	Stain Price	Stain	Quantity
10 057	Ishtar Chair	339	24	False	9
	Het Table	1 255	140	False	12
	Nun Dresser	1 548	414	False	19

List Left-Align

This style sets text alignment to the left and draws alternating color for each row.



List Break-Left

Order ID	Product Name	Unit Price	Stain Price	Stain	Quantity
10 011	Ishtar Chair	339	24	False	18
	Enlil Chair	450	27	False	18
	Enki Chair	425	24	False	18
10 023	Ishtar Chair	339	24	False	14
	Ninurta Chair	345	19	False	18
10 057	Ishtar Chair	339	24	False	9
	Het Table	1 255	140	False	12
	Nun Dresser	1 548	414	False	19
10 074	Ishtar Chair	339	24	False	18
	Enlil Chair	450	27	False	18
	Enki Chair	425	24	False	18
	Addad Dresser	2 014	487	False	6

List Break-Left

This style sets text alignment to the left, adds lines to demarcate column headers, and draws alternating color for each row.

Report styles will only effect the appearance properties of the report elements and will not change the report type or data mapping options that were selected earlier. Once you specify a style, click the *Done* button to dismiss the Wizard and continue to the main designer window.

1.4.6.1. Custom Styles

In addition to the five pre-defined styles included with EspressReport, users can create their own style definitions. Style definitions are special versions of report templates that are saved with a `.stl` extension. Users can create a style definition using any report template.

1.4.6.1.1. Creating a Custom Style

The first step in creating a custom style is to set the global formats. This allows you to control the default look and feel for all elements the user inserts into the report. For more information about global formats, see Section 1.5.9.1 - Global Formatting.

The second step in creating a custom style is to select the look and feel for the elements in each section of the report. To do this, select the report element whose formats you want to apply to all elements in the section when the style is applied and select *Set Attributes as Section Style* from the pop-up menu.

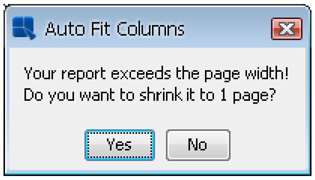
Once you finish setting the attributes, save the report template as a custom style by checking the *Create Style* option in the save as dialog.

1.4.6.1.2. Applying a Custom Style

To select a custom style when creating a report, check the *Use Custom Style* option in the report style dialog and then specify the style (`.stl`) file that you want to use. When the style is applied, the global formats will be applied and the attributes defined for each section will be applied to the elements in the corresponding section in your new report. In addition, any lines and/or images defined in the style will be applied to the new report as well.

1.4.7. Fit Columns to Page Width

Often when you finish with the Report Wizard, your created report will be wider than the default page width (8.5"). When this happens, you will be presented with a warning.



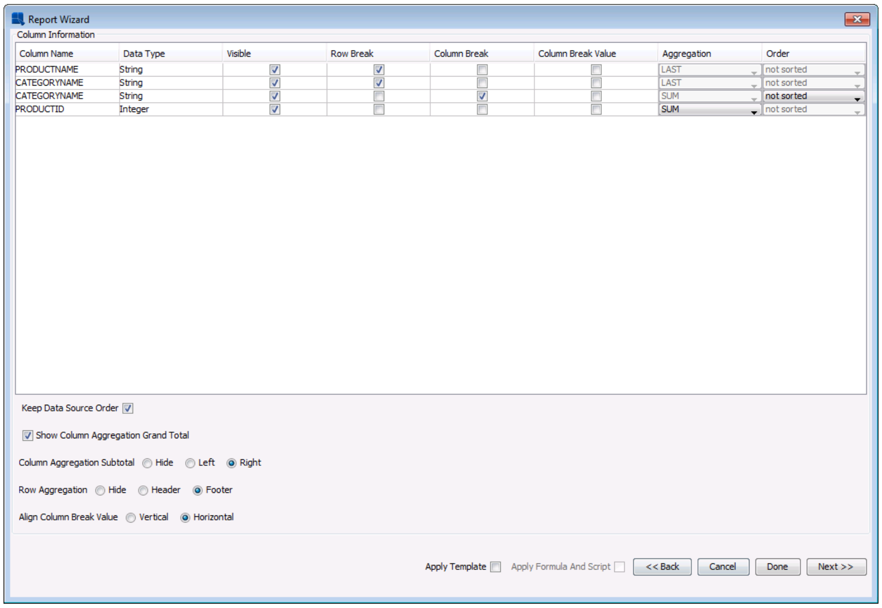
Your report exceeds the page width!
Do you want to shrink it to 1 page?

Auto-Fit Columns Dialog

The dialog gives you an option to shrink the report to fit within the page. If you select *Yes*, all columns will automatically shrink to fit the width of the page. The page boundary is indicated by a vertical bar within the design window. If you select to keep the columns overlapping the page boundary, the overflow will be printed on a new page when you preview the report. You can auto-fit the columns anytime you are editing the report by selecting *Auto Fit Columns* from the Format menu. The page width can also be adjusted by selecting *Page Setup* from the Option menu.

1.4.8. Change Data Mapping

Once you complete the data mapping, a rough version of the report will be generated. If it is not correct, you can change the data mapping by selecting *Change Data Mapping* from the *Data* menu, or clicking the *Change Data Mapping* button on the toolbar. This will take you back to the Report Wizard (starting with data mapping), allowing you to go back to change the report type and data mapping.



Column Name	Data Type	Visible	Row Break	Column Break	Column Break Value	Aggregation	Order
PRODUCTNAME	String	☑	☑	☐	☐	LAST	not sorted
CATEGORYNAME	String	☑	☑	☐	☐	LAST	not sorted
CATEGORYNAME	String	☑	☐	☑	☐	SUM	not sorted
PRODUCTID	Integer	☑	☐	☐	☐	SUM	not sorted

Keep Data Source Order ☑

☑ Show Column Aggregation Grand Total

Column Aggregation Subtotal ○ Hide ○ Left ● Right

Row Aggregation ○ Hide ○ Header ● Footer

Align Column Break Value ○ Vertical ● Horizontal

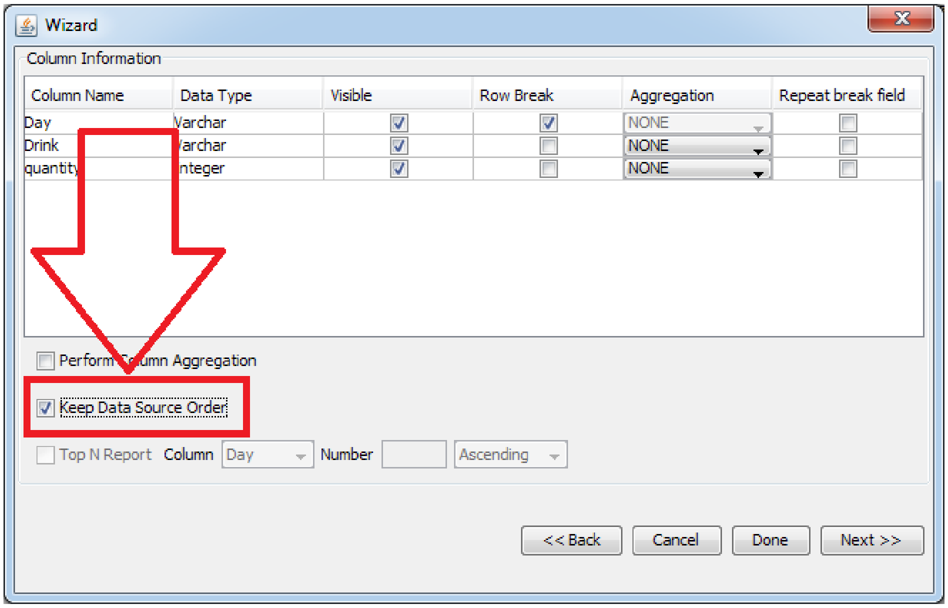
Apply Template ☐ Apply Formula And Script ☐

Change Data Mapping window

When you change the data mapping, there will be two checkboxes at the bottom of the last screen in the Report Wizard. One is marked *Apply Template* and the other *Apply Formula and Script*. Checking the first box will retain the formatting of your report; however, the labels and formulas may not be correct. Un-checking this box will create a blank report with the new data mapping. If you did not format lot of the report objects, it is recommended that you un-check this box. This will ensure that the new data mapping is completely accurate. The second box allows you to include functions and scripts placed in the data table section to be applied (formulas and scripts in other sections are always applied), when you select the *Apply Template* option. Please note that if you have changed the number of columns or the data types of certain columns, the functions and/or scripts may no longer work.

1.4.9. Keep Data Source Order

Summary Break, Cross-tab and Master&Details reports use memory-optimized query processing which can unfortunately change the order of the data. In other words: the data in the report may be displayed in different order than the data in the data source. To prevent this from happening, choose the *Keep Data Source Order* option in the report mapping dialog. This will disable the memory-optimized query processing which means that the data will be passed to the report without any optimalization (i.e. the original data ordering will be preserved).



Wizard

Column Information

Column Name	Data Type	Visible	Row Break	Aggregation	Repeat break field
Day	Varchar	☑	☑	NONE	□
Drink	Varchar	☑	□	NONE	□
quantity	Integer	☑	□	NONE	□

□ Perform Column Aggregation

☑ Keep Data Source Order

□ Top N Report Column Day Number Ascending

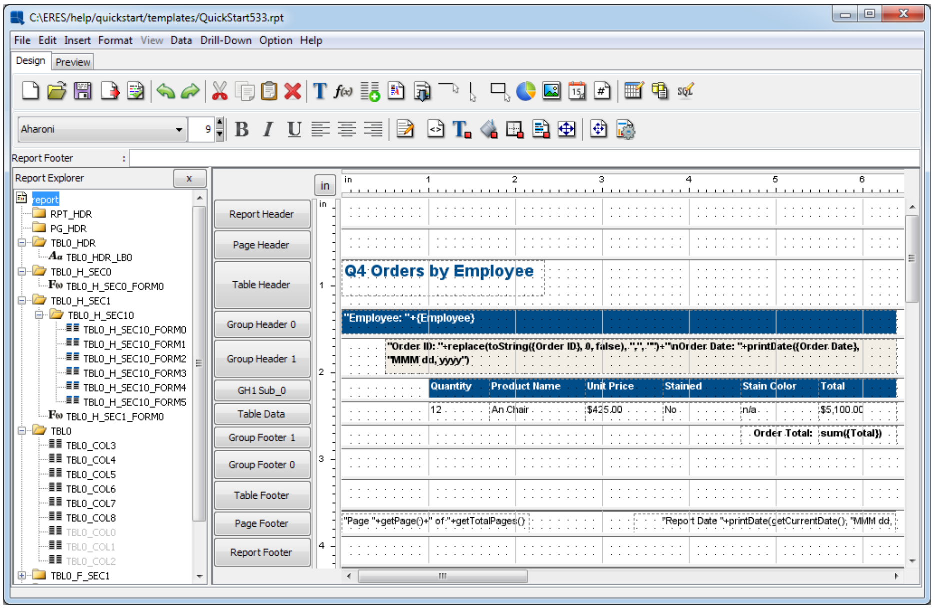
Keep Data Source Order option

Note

If you choose the *Keep Data Source Order* option, the *Top N Report* option for Master&Details and Summary Break reports will be disabled.

1.5. The Designer Interface

Users who have some familiarity with other report writers will recognize the banded-style interface of the Report Designer. Report elements can be added and modified within the Designer window and then the resulting report can be previewed in the preview window.



Report Designer Interface

1.5.1. Report Sections

Each group within the designer interface represents a different report section. The section names are listed on the buttons to the left of each report group.

Report Header: This section is like the title of the report. It appears only once at the top of the report.

Page Header: Any element placed in this section will appear at the top of every page of the report.

Table Header: This section serves as a header to the detail or data section of the report. By default, it only appears once in the report.

Group Header: This section appears in reports with grouped data (i.e. master & details report), or data with row breaks inserted (i.e. summary break report). It repeats at the top of each grouping within the report.

Table Data: This is the main section of the report that contains most of the data. Data columns that have been selected for the report are placed in this section and are repeated for each entry in the column.

Group Footer: This section appears on reports with grouped data (i.e. Master & Details report) or data with row breaks inserted (i.e. summary break report). It repeats at the bottom of each grouping within the report.

Table Footer: This section serves as the footer or data section of the report. By default, it appears only once in the report.

Page Footer: Any element placed in this section will appear at the bottom of every page of the report.

Report Footer: This is the last summary of footer section of the report. It only appears once at the end of the report.

1.5.1.1. Nested Sections

There can also be nested sections in addition to the Report Header, Table Header, Group Header, Group Footer, Table Footer, and Report Footer sections. Nested sections are useful for placing sub-reports and rich text fields with variable lengths. This allows these type of elements to resize without overlapping any elements below. Nested sections inherit most of the section options from their parent sections, including page-breaking and repeating options.

To add a nested section, select *Insert Section* from the section options pop-up menu for the parent section. For more information about section options, options see Section 1.5.3 - Section Options. Note that if you select to insert a section from a nested section, it will inherit the same parent section (i.e. nested sections cannot be a parent).

1.5.2. Resizing Sections

To resize section height, place your mouse over the section divider within the design window. The pointer will change to a resize pointer. Click and drag the mouse to expand or decrease the section height.

1.5.3. Section Options

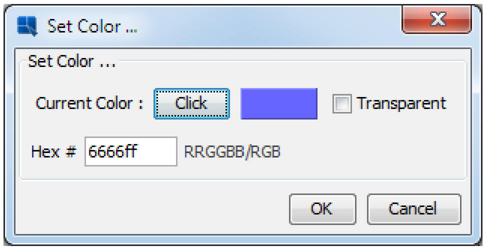
There are several formatting and display options available for each report section. You can access the section options menu by clicking the button with the section name at the left side of the Designer or by right clicking on a blank portion of the section field.

Background Color: This option is available for every section of the report. It allows you to set background color for the entire section. Selecting this option will bring up a dialog box prompting you to specify whether or not to set the background transparent. If you do not want a transparent background, uncheck the checkbox and click the button. This will bring up a dialog prompting you to specify the background color. You can select font color from swatches or enter

HSB/RGB values. You can also directly enter a HEX color value to the *Hex #* field.

Note

If there are no elements placed in a section, that section will not display. Hence, changing a section background color will have no effect on the finished report if the section is blank.



Background Color Dialog

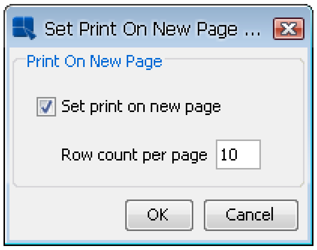
Section Height: This option is available for every section of the report. It allows you to set the section height. It has the same effect as dragging the section's bottom edge, but unlike the drag&drop method, this option allows you to set the section height precisely in inches.

Insert Section: This option is available for Report Header, Table Header, Group Header, Group Footer, Table Footer, and Report Footer sections. Selecting this option will add a nested section to the currently selected report section.

Remove Section: This option is only available for nested sections that you have added to the report (regular sections can only be rendered invisible). This option will remove the nested section from the report and delete its contents.

Repeat On Every Page: This option is available for Table Header and Group Header sections. Selecting this option causes the section to repeat again after every page break (group headers will only repeat if the data for that group exceeds one page).

Print On a New Page: This option is available for Table Header, Group Header, Table Data, Group Footer, Table Footer, and Report Footer sections. For the Table Data section, a new dialog will pop-up allowing you to select the number of rows to display on each page. This number will be set as a new maximum of table data rows for the page. After this amount of rows a page break will be inserted. This count does not include any formulas in the footer sections. For example, if the report contained row breaks and an aggregation calculated for each group, this element will not count towards the number of rows.



Print on New Page Dialog

Selecting this option for any other section inserts a page break each time the section appears.

New Excel Sheet: This option is only available for Group Header section(s). Selecting this option causes each group to be printed on a separate worksheet when the report is exported to Excel (XLS) or Excel 2007 (XLSX) format.

Reset Page Number: This option is only available for Group Header section(s). Selecting this option causes the report pagination to reset each time the section is run. Hence, each time the Group Header section runs, the current page number goes to 1. Note that setting this option will automatically enable the print on a new page option.

Fit Group On Page: This option is only available for Group Header section(s). It allows you to control some of the pagination for grouped data. Selecting this option will cause EspressReport to try to fit the entire group on a page. If the entire group cannot print on a page, it will print on a new page. If the group cannot fit on any page, (the fields are longer than the page height), it will be broken up into multiple pages.

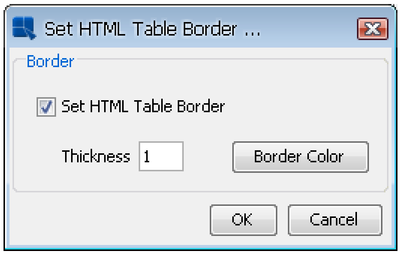
Skip First Value: This option is available for Table Header, Group Header, Group Footer, and Table Footer sections. This option is a subset of the print on a new page option. Selecting this option will ignore the page break for the very first time a section is run. For example, setting this option for the Group Header section will cause the report not to insert a page break for the first time the section is run. The page break will only occur when the report runs the Group Header for a second time.

Skip First Group Value: This option is only available for Group Header section. This option is a subset of the print on a new page option. Selecting this option will ignore the page break for the first value of each group of data. This option is used for nested grouping in a report where more than one group is set to print on a new page. To prevent a blank page for records where more than one grouped column value changes, the 'Skip First Group Value' can be set for the inner Group Header section.

Resize Cells Proportionally: This option only applies if you have set cells within the section to resize to fit content. If this option is selected, all cells in the section will resize with the variable height cells.

Invisible: This option is available for every section of the report. Selecting this option hides the entire section. To make the section visible again, click on the *Format* menu at the top of the designer and select *Make Section Visible* option.

HTML Border: This option is available for every section and only takes effect if the report is exported to HTML. Since tabular HTML (not DHTML CSS formatting) does not allow selective cell borders within a table, users can select to turn on or off a border for all cells in a section when the report is rendered in HTML. If you select this option, the following dialog will appear allowing you to set options for the HTML border:



☑ Set HTML Table Border

Thickness 1

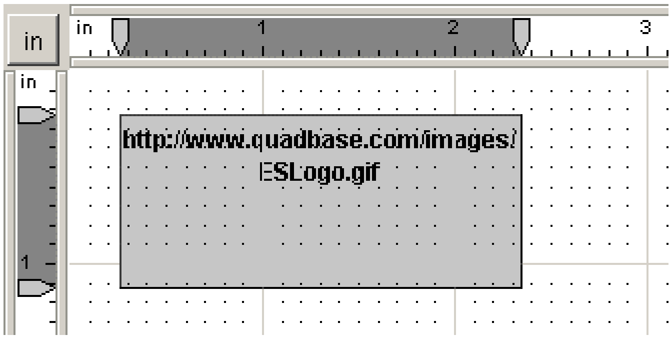
HTML Border Options

In this dialog, you can set thickness and border color by clicking the *Border Color* button. This will open the color palette dialog allowing you to pick a color for the HTML border.

Scripting: This option will bring up scripting interfaces allowing you to write, modify, or apply a script to dynamically modify certain section attributes. For more information about this feature, please see Section 1.9.2.7 - Section Scripts.

1.5.4. Rulers

The Report Designer has two rulers in the top left corner of the Design window. The rulers measure how the report will look in terms of the physical page dimensions. Please note that the dimensions do not include page margins. The rulers helps you to get a good idea of how the printed report will look like. You can also use the rulers to adjust the bounds of elements in your report. Selected element bounds will appear as shaded areas within the rulers and can be adjusted by clicking on and dragging the markers.



Rulers and in/cm button

1.5.4.1. Toggling Between Inches and Centimeters

The button in the upper left corner of the Design window allows you to toggle between inches and centimeters. By default, EspressReport uses inches for measurement. Depressing the button will change the ruler measures to centimeters. If centimeter is selected, all of the manually entered values like element bounds and page margins will also be entered and displayed in centimeters.

You can set the Report Designer to use metric (cm) measurements by default. To do this, specify an additional command line argument `-metricSystem:true/false` when executing `ReportDesigner.bat/.sh` file. For example: `ReportDesigner -metricSystem:true` will launch the Report Designer with metric measurements. You can also add this argument into the batch file.

If you are running Report Designer through an applet, you will need to add the following parameter to the applet HTML page: `<PARAM NAME=metricSystem VALUE="true">`.

1.5.5. Designer Menus

Most of the Report Designer's features can be controlled using the drop-down menus at the top of the designer.

1.5.5.1. File Menu

This menu performs most of the file operations.

New: Creates a new report.

Open: Opens an existing report (`.pak/.rpt/.xml/.stl` format).

Close: Closes the current report.

Apply Template: Imports and applies a template (`.pak/.rpt` or `.xml`) file to the current report. All of the report attributes are changed to those of the template.

Save: Saves the current report.

Save As: Allows you to name and save the current report as a template (`.pak`) file. A checkbox within the *Save As* window allows you to generate an applet page with the Report Viewer embedded, as well as generate an XML template.

Export: This option exports the report to a selected format. Options include DHTML, PDF, CSV (data file), Excel (XLS), Excel 2007 (XLSX), RTF, TXT, and XML.

Print: Prints the current report (This option is only executable from Preview tab.)

Exit: Closes the application.

1.5.5.2. Edit Menu

This menu allows you to edit and cut/paste report elements.

Undo: Cancels the last operation performed in the Designer and reverts back to the previous state. The Designer will remember last 10 actions made.

Redo: Reverses the action of the undo command. For example, if you change the font color from black to red, you can undo this command to change it back to black and then redo this command to have it changed back to red again.

Edit: This option is available for labels, formulas, images, and charts. For labels, it will prompt you to change the label text. For formulas, it will prompt you to change the formula. For images, it will prompt you to change the image file URL. For charts, it will bring up Chart Designer to edit your chart.

Copy: This option makes a copy of the selected element and places it on the clipboard.

Cut: This option removes the selected element and places it on the clipboard.

Paste: This option pastes the current clipboard element into the report.

Delete: This will delete the selected element, except for elements in the Table Data section. It will not delete these, but render them invisible.

Remove Sub-Report: Removes a sub-report from the current report.

1.5.5.3. Insert Menu

This menu allows you to insert elements into the current report.

Insert Label: Inserts a label element into the current report.

Insert Formula: Inserts a formula element into the current report.

Insert Column Field: Inserts a column field into the report from the available fields (Available fields are those that have been selected for the report. See data mapping: Section 1.4 - Report Types and Data Mapping).

Insert Database Field: This option is only available if the template's data source is a database. It allows you to insert a field from a database that is not part of the original query.

Insert Column Header: Inserts a column header for one of the report fields.

Insert Parameter Value: Inserts a formula that returns the supplied value for a report parameter.

Insert Chart: Inserts a new chart into the current report.

Insert Image: Inserts an image from a file into the report. Images can be in JPEG, GIF, or PNG format.

Insert Rich Text Field: Inserts a new rich text into the current report.

Insert Table of Contents: Inserts a table of contents into the report.

Insert Line: Inserts a vertical or horizontal line into the report.

Insert Rectangle: Inserts a grid rectangle.

Insert Guideline: Inserts a guideline into the report. Guidelines can help you move and position elements in the report. For more information about guidelines, please see Section 1.5.7.14 - Guidelines.

Today's Date: Inserts the current date into the report in one of three ways: `Date`, `Date & Time`, or `Time`. The date function is actually a formula and the format can be adjusted after it is inserted. For more information about formula formatting, please see Section 1.5.7.3 - Data Formatting for Formulas and Column Fields.

Page Number: Inserts the page number into the report in one of two ways: `Page Number` or `Page Number of Total Pages (i.e. page X of Y)`. The page number function is actually a formula and it can be formatted after it is inserted. For more information about formula formatting, please see Section 1.5.7.3 - Data Formatting for Formulas and Column Fields.

Insert Sub-Report: Inserts a new or pre-existing sub-report into the current report. For more information about sub-reports, please see Section 1.11 - Sub-Reports.

1.5.5.4. Format Menu

The format menu contains formatting options for the currently selected element within the report. Some menu options will not work depending on the type of element which is currently selected.

Data Format: This option is available for formulas and column fields. It will bring up the data format editor for the data type of the selected element, either `string`, `numeric`, `date/time`, or `Boolean`.

Chart Export Format: This option is only available for charts. When reports are exported to DHTML format, charts are converted to static images. This option allows you to specify the image properties for the chart upon export.

Edit Attributes: This option is available for every report element. It will bring up a dialog box that allows you to adjust attributes depending on the selected element (e.g. bounds, background color, alignment, font style and size, rotation, border/round corners, data format).

Font Style and Size: This option is available for labels, formulas, and column fields. It brings up a dialog box prompting you to change the font appearance and the size of the selected element.

Background Color: This option is available for labels, formulas, and column fields. It allows you to adjust the background color of the selected element.

Dual Colors: This option is only available for elements in the Table Data section of the report. It allows you to specify alternating background colors, font colors, and font styles for the selected column, as well as the number of rows between changes or on which column field to base the change.

Line Color & Thickness: This option is only available for lines. It allows you to set the color and the thickness for the selected line(s).

Border: This option is available for every report element. It allows you to specify thickness (in pixels), and color and shape of corners (sharp or round) of the border drawn around the selected element. The border thickness/corners can be set individually for the each side/corner of the element.

HyperLink: This option is available for all elements except column fields. It allows you to hyperlink an element to a web page or another report.

Scripting: This option is available for all labels, formulas, and column fields. It allows you to create a script and apply conditional formatting to the selected element.

Rotation: This option is available for all labels, formulas, and column fields. It allows you to rotate the element 90 degrees clockwise or counter-clockwise.

Z-Index: This option allows you to set the z index for the selected report element. The z index controls how report elements behave (i.e. which one appears first) when placed on top of each other.

Bounds: This option is available for every report element. This feature allows you to manually enter the boundaries of an element. The measurements will be in inches or centimeters depending on which unit is selected with the toggle button in the upper left corner of the Designer window where the rulers meet.

Alignment: This option is available for labels, formulas, and column fields. It allows you to align the selected element horizontally to the left, center,right, as well as vertically to the top, middle,or bottom.

Wordwrap: This option is available for labels, formulas, and column fields. It allows users to enable or disable word wrapping within report elements.

Resize To Fit Content: This option is available for labels, formulas, and column fields. It causes the height of the selected element to adjust dynamically to fit its content.

Make Column Visible: This option is only available if you deleted an element in the Table Data section, or if you selected an invisible data column from the column options screen of the Report Wizard. It allows you to render any invisible column visible.

Make Section Visible: This option is only available if you rendered a report section invisible (See Section 1.5.3 - Section Options). It will make the section visible again.

Invisible: This option is available for elements in the Table Data section or for report sections. It will make the selected element invisible.

Column Wrap: This option allows column fields to be wrapped, so rather than breaking to the next page at the end of the page, they will continue next to the original column field within the current page.

Auto Fit Columns: This option will shrink all elements within the report to fit the report within the page width.

Swap Columns: This option allows the position of two elements to be swapped in the report. Elements must first be selected using **CTRL+Click**.

1.5.5.5. View Menu

The view menu contains options allowing you to navigate through the report in the Preview window and sort the data. It is only active when you are previewing the report.

First Page: This option will navigate to the first page of the report within the Preview window.

Previous Page: This option will scroll back one page within the Preview window.

Next Page: This option will scroll forward one page within the Preview window.

Last Page: This option will navigate to the last page of the report within the Preview window.

Go To Page...: This option will prompt you to enter a page number and then navigate to that page of the report within the Preview window.

Sort by (ascend): This option opens up a second menu allowing you to select one of the report columns to sort the report in ascending order (either numeric or alphabetical).

Sort by (descend): This option opens up a second menu allowing you to select one of the report columns to sort the report in descending order (either numeric or alphabetical).

Sort by... This opens a dialog which allows you to select nested sorting in a report (by more than one column). The direction of the sorting can be set for each column.

Launch Page Viewer: This will load the entire report in the page viewer window. For more information about this feature, see Section 1.6.2.4 - Using Page Viewer.

1.5.5.6. Data Menu

The data menu contains options that allow you to see and manipulate the data used in the current report and its properties.

Refresh: This option will re-query the data source used for the current report and then update the report.

Change Data Mapping: This will re-open the Report Wizard, allowing you to change data mapping for the current report.

Change Data Source: This will re-launch the Data Source Manager, allowing you to modify or change the report's data source.

Modify Database: If the report uses a database as the data source, this option allows you to modify the connection information that the report uses to connect to the database. For more information about this feature, please see Section 1.3.2.5 - Editing Database Connections.

Modify Query: If the report uses a database as the data source, this option allows you to modify the query used to retrieve the report data. For more information about this feature, please see Section 1.3.2.4 - Editing Queries.

View Column Mapping: This option will bring up a window that displays column mapping for the current report. The window displays the column index, name, data type, options, and whether it is currently visible or invisible.

View Table: This option will bring up a window containing the data table from which the current report is generated. The table will initially display only the first 20 records. Clicking on the *Show All Records* checkbox will display all of the records in the data table.

View Data Source Info: This option will bring up a window containing information about the data source that was used to create the template. The data source type, location, and the data registry location will be displayed.

Sub-Report Parameter Sharing: This allows you to link parameters defined within sub-reports to parameters in the main report or to the parameters defined in other sub-report. With this feature enabled, reports can share a user supplied parameter value automatically without having to pass the value to each sub-report. For more information about sub-report, see Section 1.11 - Sub-Reports.

Sub-Report Parameter Mapping: This allows you to link parameterized sub-reports to column fields in the primary report. For more information about sub-reports, see Section 1.11 - Sub-Reports.

Chart Parameter Mapping: This allows you to link parameterized charts (with an independent data source) to column fields in the report. For more information about this features, see Section 3.3.1.1 - Chart Parameter Linking.

Select Multiple Drill-Down Values: This option is only available when you preview a multi-value drill-down report. It allows you to select values to use when drilling to the next level. For more information about drill-down reports, please see Section 1.10 - Drill-Down.

Security: This will bring up the security settings dialog for the currently selected cell. For more information about the template security, please see Section 1.12 - Template Security.

Cell Properties: This option is available for all elements in the report. It will bring up a window displaying the properties of the selected element including its ID value, as well as column index, name, and data type for column fields.

Set Preview Security Level: This option allows you to specify which defined security level you want to use when previewing the template. For more information about the template security, please see Section 1.12 - Template Security.

Set Preview Data Options: This option allows you to turn on/off live data for the preview window as well as set the maximum number of records that should be returned when previewing the report.

Secured Query Parameters: This will bring up the parameter security dialog, allowing you to assign parameter values to security levels. For more information about the template security, please see Section 1.12 - Template Security.

Preview Parameter Prompt: This option allows you to enable/disable parameter prompting when you preview the report.

1.5.5.7. Drill-Down Menu

The drill-down menu contains options that allow you to add, remove, and navigate layers of drill-down within the report. For more information about drill-downs, please see Section 1.10 - Drill-Down.

Navigate: This option brings up the drill-down navigation window allowing you to edit, add, and remove layers of drill-down.

Drill-Down Link: This option allows you to link the currently selected element to a drill-down layer.

1.5.5.8. Option Menu

This menu contains display and printing options

Report Explorer: This option turns on/off the report explorer display. For more information about the explorer interface, see Section 1.5.8 - The Report Explorer.

Page Setup: This option allows you to customize the page size and margins. The measurements will be in inches or centimeters depending on which unit is selected with the toggle button in upper left corner of the designer window where the rulers meet.

HTML Page Title: This option allows you to specify the page title that is generated when the report is exported to DHTML format.

Background Color: This option allows you to set the page background color for the report. Specifying a color here will color the entire page in Report Viewer and when exporting to DHTML or PDF. Any non-transparent section color or report element background color will draw over the background color.

Background Image: This option allows you to specify a background image for the report.

Snap to grid: This option allows you to control the settings for the snap to grid feature. You can turn the feature on/off, as well as control the step size.

Font Mapping: This allows you to map system (true type) font files for the PDF export. For more information about this feature, please see Section 1.7.2.1 - PDF Font Mapping.

Viewer Font Setup: This option allows the font size for text in the Report Designer and Report Viewer to be relative to the screen resolution. Turned on by default, this feature allows for more precise conversions of reports to various export formats and between installations.

Export Style Sheet: This option allows you to export the style sheet (.css) file with the style definitions for the elements in the report. This style sheet file can be used to apply the DHTML export settings onto other reports. For more information about working with CSS, please see Section 1.7.2.2 - CSS Options.

Show Cell Outline: This option draws the outline for all the elements in the report. It allows you to see the boundaries of all elements, as well as see elements where the text may not be visible.

Shift On/Off: This option allows you to turn the cell shifting mode on or off. If the shift mode is on, resizing and moving report elements will cause the surrounding elements to shift: accommodating the resized or repositioned element.

Show Formula Name: This option allows you to enable/disable displaying the function name, rather than the text of the function in the Design window. For more information about working with formulas, see Section 1.8 - Using Formulas & the Formula Builder.

Enable Table of Contents: This option allows you to show/hide a table of contents (if it is inserted).

Global Format: This option allows you to set formatting for each of the different type of report elements. It will apply changes to elements currently in the report, as well as change the default properties.

Null Data Handler: This option allows you to specify the treatment of null data within the report.

1.5.5.9. Help Menu

This menu allows you to view program version and open the User's Guide.

About: This option shows you information about the program version.

Contents: This option opens the EspressReport User's Guide.

1.5.6. Designer Toolbar

The toolbar at the top of the designer offers easy access to EspressReport's most commonly used features. The first row of buttons perform the following functions:

Start a new report

Open an existing report

Save the current report

Export the current report

Apply a template to the current report

Undo the last change

Redo the last undone change

Cut the selected element and place it on the clipboard

Copy the selected element

Paste the current clipboard object into the report

Delete the selected element

Insert a label

Insert a formula

Insert a column field

Insert a rich text field

Insert a sub-report

Insert a horizontal line

Insert a vertical line

Insert a grid rectangle

Insert a chart

Insert an image

Insert the current date

Insert page number

Change Data Mapping

Change Data Source

Edit Query

The first two boxes in the second row of the toolbar allows you to select font face and size for the currently selected element. Buttons in the second row perform the following functions:

Set the font style to bold

Set the font style to italic

Underline the text

Set the horizontal alignment to left

Set the horizontal alignment to center

Set the horizontal alignment to right

Edit the selected element

Set data format for the current element

Apply a script to the current element

Set font style, size and color for the current element

Set background color for the current element

Set border thickness, color and corner rounding for the current element

Set dual colors for the current element

Set bounds for the current element

Page setup

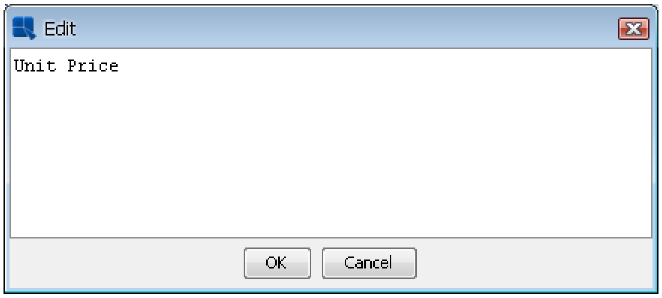
Limit preview display rows

1.5.7. Inserting and Manipulating Report Elements

When you finish with the Report Wizard, a rough version of your report will be generated based on the mapping and it will display the options you selected. To polish the report, you may want to insert new report elements, as well as format the existing ones.

1.5.7.1. Inserting Elements

Labels: You can insert a label in one of two ways: by selecting *Insert Label* from the *Insert* menu or by clicking the T *Insert Label* button on the toolbar. After you select the *Insert Label* option, a small box will follow your mouse pointer around the Design window. Position the box where you want to insert the label and click. A dialog box will appear prompting you to enter the label text. Click the *OK* button and the label will appear in your report.



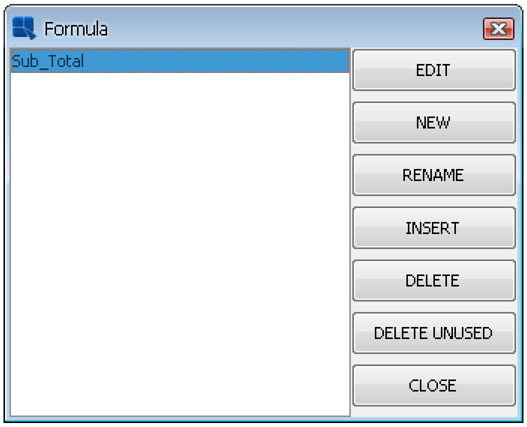
Unit Price

Insert Label Dialog

Formulas: You can insert a formula in one of two ways: by selecting *Insert Formula* from the *Insert menu* or by clicking the *f(x)* *Insert Formula* button on the toolbar. After you select the *Insert Formula* option, a dialog box will open allowing to select a formula to insert. To insert a formula, select a formula from the list and click the *Insert* button. A small box will follow your mouse pointer around the Design window. Position the box where you want to insert the formula and click. The formula will be added. You can use an existing formula or create a new one. To create a new formula, click the *New* button in the formula list dialog. After entering a name for the new formula, the Formula Builder window will open allowing you to construct the formula (See Section 1.8 - Using Formulas & the Formula Builder for more information about using the formula builder and formula syntax). Any formula is automatically anchored to the report section in which it is inserted. This means that the formula will reset each time the section repeats.

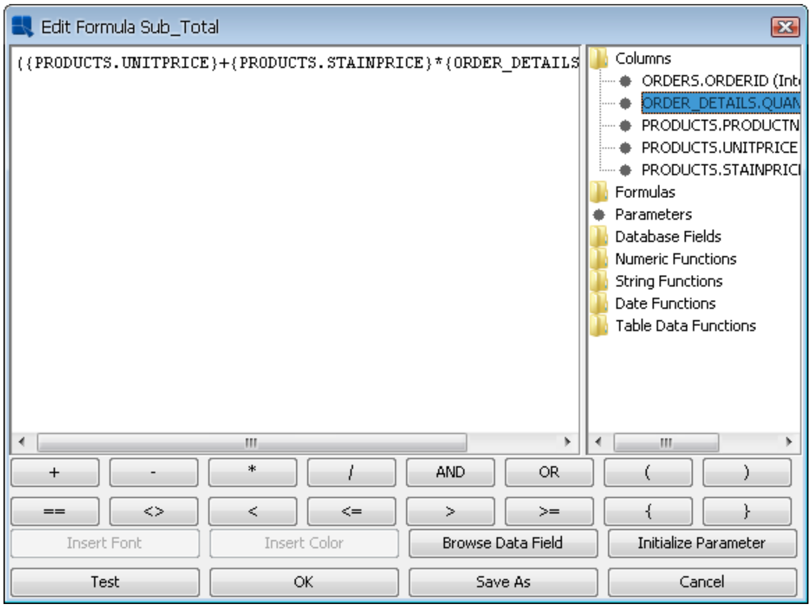
Note

Only the text of the formula will appear in the design window (unless the formula is inserted in the Table Data section). You can choose to display the formula names instead of the text by selecting *Show Formula Name* from the Option menu.



Sub_Total

Formula List

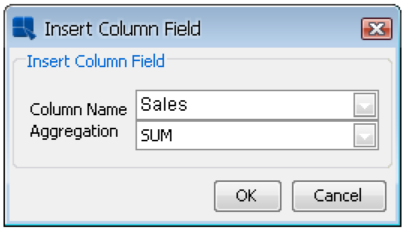


```
({PRODUCTS.UNITPRICE}+{PRODUCTS.STAINPRICE}*{ORDER_DETAILS
```

Formula Editor

Column Fields: You can insert a column field in one of two ways: by selecting *Insert Column Field* from the *Insert* menu or by clicking the *Insert Column Field* button on the toolbar. After you select the *Insert Column Field* option, a dialog box will appear with two drop-down menus. The first one allows you to select the column field you want to insert into the report. The second one allows you to select aggregation (if any) for the column field. After you select the desired column field, click the *OK* button and a small box will follow your mouse pointer around the design window. Position the box where you want to place the column field and click. The column field will appear in the report. The column field is automatically anchored to the report section in which it is inserted. This means that each time the section repeats, the column field will display the next row in the data column.

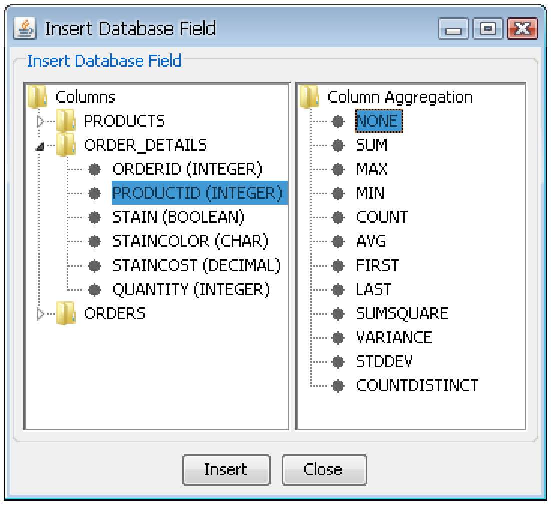
> ### Note
>
> The column value will not appear in the Design window (unless it is placed in the Data Table section). All that will show is the column name `{name}`. You can see the column field value in the preview window.



Column Name: Sales
Aggregation: SUM

Insert Column Field

Database Field: This option is only available if the current report uses a database as the data source. It allows you to add a field from the database into the report that may not have been initially selected by the query or as part of the data mapping. To insert a database field, select *Insert Database Field* from the *Insert* menu. This will open a dialog with all of the tables you have included in your query and their respective fields, allowing you to select the field you want to add. Another option allows you to select the aggregation (if any) that you want to perform on the field.

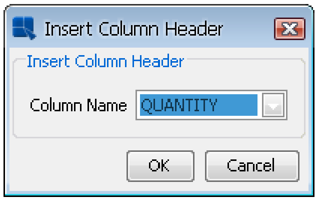


Insert Database Field Dialog

Note

If the template's query contains aggregation or a `Group by` clause, a third option will appear in the dialog prompting you to select `Database Aggregation` for the field. This can either be group by or any aggregation supported by the database.

Column Header: Column headers are a unique report element that will dynamically display column header from the data source. You can add a column header by selecting *Insert Column Header* from the *Insert* menu. This will bring up a dialog prompting you to select the column for which you want to retrieve the header.

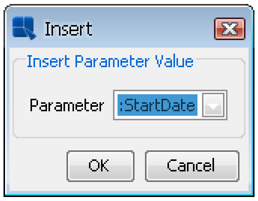


Column Name: QUANTITY

Insert Column Header Dialog

Select a desired column and click the *Ok* button. A box will then allow you to position the header wherever you want in the report. Click to add the column header.

Parameter Values: You can insert user supplied values to query or formula parameters directly into the report. To insert a parameter value, select *Insert Parameter Value* from the *Insert* menu. This will bring up a dialog prompting you to select which parameter you want to display the value for.



Insert Parameter Value Dialog

Parameter values are also accessible through formulas. For more information about formulas, please see Section 1.8 - Using Formulas & the Formula Builder.

Images: You can insert an image in one of two ways: by selecting *Insert Image* from the *Insert* menu or by clicking the 🖼 *Insert Image* button on the toolbar. After you select this option, a small box will follow your mouse pointer around the Design window. Position the box where you would like to place the image and click. A new window will appear prompting you to enter the URL of the image you want to insert.

There are two ways of inserting images: either locate the image on your hard drive or retrieve it from an URL.

To locate an image on your hard drive, click the *Browse* button. If you want to use an image from a different location, insert its URL.

To retrieve image from an URL, enter the URL in the *Image URL* text field. After that, click the *Refresh Preview* button to verify the URL. If the image from the URL appears in the *Preview* section, the URL is correct.

Note

You have to insert complete URL with protocol (e.g. `http://`).

If you save the report as PAK, the image will be stored in the PAK file along with the report.

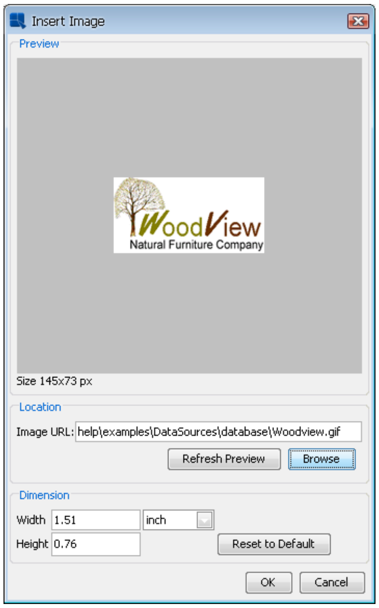
If you want to re-open a RPT file (from an older EspressReport version) with a background image, please note that the image itself is not stored within the report. Only the path or URL is saved. If you move a RPT report, you need to be sure that it can still access the image using the specified path.

You can manually set the image dimensions in pixels, inches, or millimeters in the *Dimension* section.

Note

If the image dimension exceeds the page dimension, the image will not be shown (although space will be allocated for that image).

EspressReport also supports database images. You can add images from a database (BLOB format) by selecting the BLOB field as part of the query for the report. The images will be added as a column in the report. You can also retrieve images in the Table Data section via URL instead of using a BLOB field. For more information about this, please see Section 1.5.7.3 - Data Formatting for Formulas and Column Fields.



Size 145x73 px

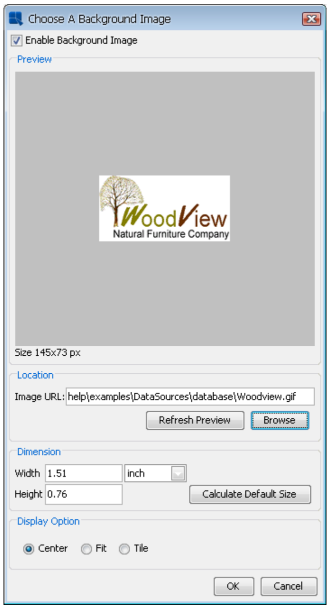
Image URL: help\examples\DataSources\database\Woodview.gif

Width 1.51 inch

Height 0.76

Insert Image Dialog

Background Images: EspressReport allows you to add background images to a report. Background images will underlay all other report elements and will repeat on every page. To add a background image, select *Background Image* from the *Option* menu. This will bring up a dialog prompting you to specify location of the image, as well as several display options.



☑ Enable Background Image

Size 145x73 px

Image URL: help\examples\DataSources\database\Woodview.gif

Width 1.51 inch

Height 0.76

● Center ○ Fit ○ Tile

Background Image Dialog

To insert a new image, select the *Enable Background Image* option. If you want to remove an existing background image and use simple background color instead, unselect this option.

This dialog also allows you to choose one of the *Display options* (*Center*, *Fit* and *Tile*).

The rest of this dialog is the same as the *Insert image* dialog described in the previous section.

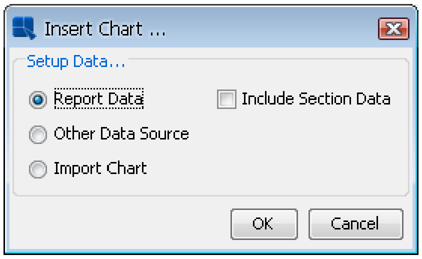
Lines/Rectangles: To insert a line or a grid rectangle, select either *Insert Line* or *Insert Rectangle* from the *Insert* Menu. If you select a line, you can further select whether you want to insert a vertical or horizontal line. You can also select the corresponding icons on the toolbar. After you select a line or rectangle, a cross or a box will follow the mouse pointer around the Design window. Position the cross/box where you would like to insert the line/rectangle and click, then drag the cross to draw the line or rectangle. A line or grid will then be inserted into the report.

Note

The line or grid will appear thicker than it actually is in the Design window.

1.5.7.1.1. Inserting Charts

You can insert a chart in one of two ways: by selecting *Insert Chart* from the *Insert* menu or by clicking the *Insert Chart* button on the toolbar. After you select this option, a small box will follow your mouse pointer around the design window. Position the box where you want to place the chart and click. You will see the following dialog box:



Insert Chart ...

Setup Data...

● Report Data

○ Other Data Source

○ Import Chart

□ Include Section Data

Insert Chart Dialog

You have an option to use data from the current report or use an external data source. If you choose to use the data from the report, you will be shown a preview of your dataset and you will be taken directly to the chart selection window in the Chart Designer. Otherwise, you will be given an option to either select a different data source for your chart or import a chart file created before.

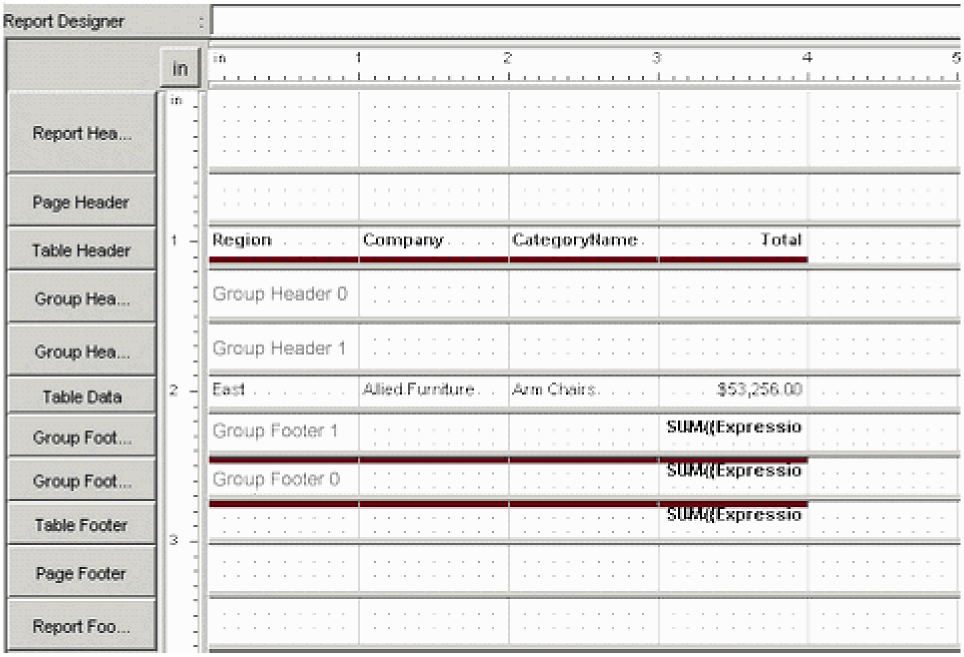
Once you select your options, the Chart Designer will open in a new window, allowing you to design and customize your chart. Please see the Chapter 3 - Charting Guide for more information about using the Chart Designer.

Note

The actual chart will not appear in the Design window. Instead, you will see a gray rectangle with the chart URL. You can see the actual chart in the preview window.

If your report contains grouping (Summary Break, CrossTab, or Master & Detail Reports), you will also have the option to *Include Section Data*. If you select this option, you will be able to use group aggregations that are located within any section that the chart encompasses. This means that you can create charts using data from any aggregation located in the same section as the chart or in any inner sections.

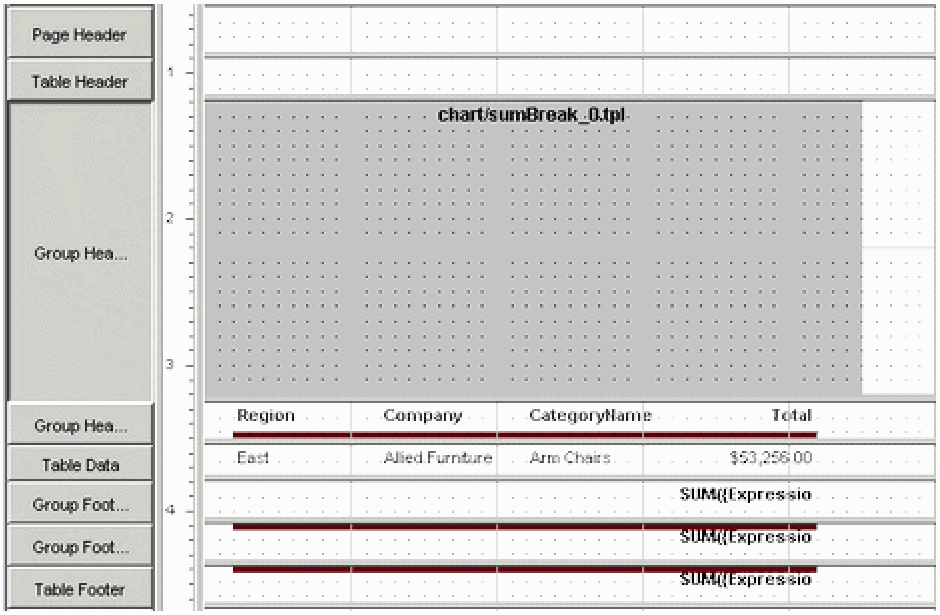
For example, if you were to create a Summary Break Report with two row break columns, this will result in two group header and two group footer sections. Assuming we have a group aggregation for a column in your report, the report might look like this:



Summary Break Report with Two Row Break Columns

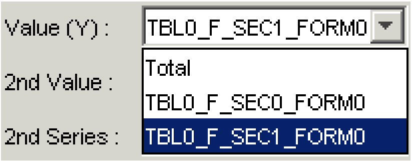
If you position the chart in the inner group header (Group Header 1) or inner group footer (group footer 1), you will only be able to use the group aggregation for the inner section in your chart (the aggregation in Group Footer 1). If you positioned the chart in the outer group header (Group Header 0) or outer group footer (Group Footer 0), you will be able to use the aggregation for the outer section as well as the inner section in your chart because the outer section encompasses the inner section. Positioning the chart in the table header or table footer sections will allow you to use any group aggregation in the report.

Suppose you moved the column headers to Group Header 1 and positioned the chart in Group Header 0 as shown below.



Insert Chart in Outer Group Header

When you see the data mapping window in Chart Designer, you will be able to use the summary data from both Group Footer 0 (TBL0_F_SEC0_FORM0) and Group Footer 1 (TBL0_F_SEC1_FORM0).



Value (Y) : TBL0_F_SEC1_FORM0

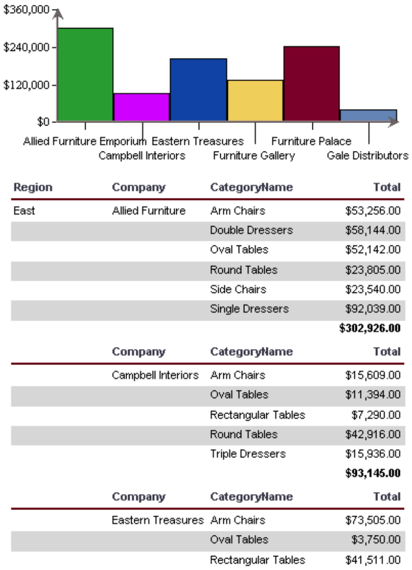
2nd Value :

2nd Series :

Chart Mapping with Section Data

To make the aggregation easier to recognize, you can use Custom IDs. For more information about using Custom IDs, see Section 1.8.2.1 - Using Column Field Data.

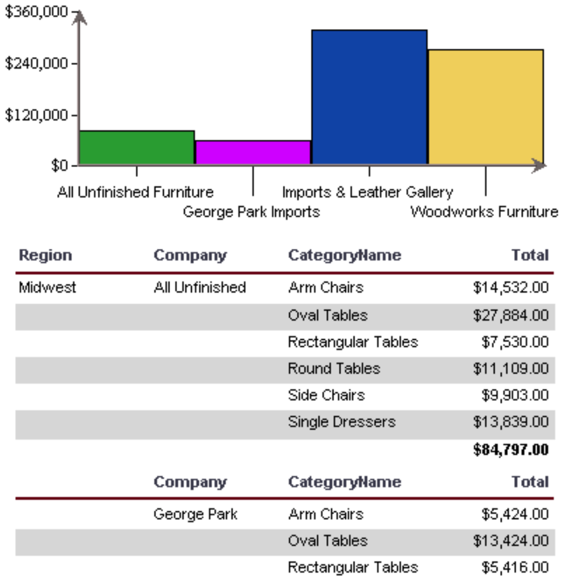
The resulting report will look like the image below. Notice that the chart plots the aggregation for each company. If you used the aggregation from Group Footer 0, the chart would have shown the aggregation for each region instead.



Region	Company	CategoryName	Total
East	Allied Furniture	Arm Chairs	$53,256.00
		Double Dressers	$58,144.00
		Oval Tables	$52,142.00
		Round Tables	$23,805.00
		Side Chairs	$23,540.00
		Single Dressers	$92,039.00
			$302,926.00
	Company	**CategoryName**	**Total**
	Campbell Interiors	Arm Chairs	$15,609.00
		Oval Tables	$11,394.00
		Rectangular Tables	$7,290.00
		Round Tables	$42,916.00
		Triple Dressers	$15,936.00
			$93,145.00
	Company	**CategoryName**	**Total**
	Eastern Treasures	Arm Chairs	$73,505.00
		Oval Tables	$3,750.00
		Rectangular Tables	$41,511.00

Report with Chart Using Section Data (East)

Since the chart is placed in Group Header 0, a new chart will be displayed for every region. The above image shows the chart for the East region only, but the report will contain charts corresponding to each of the other regions as well. Here is the chart for the Midwest region:



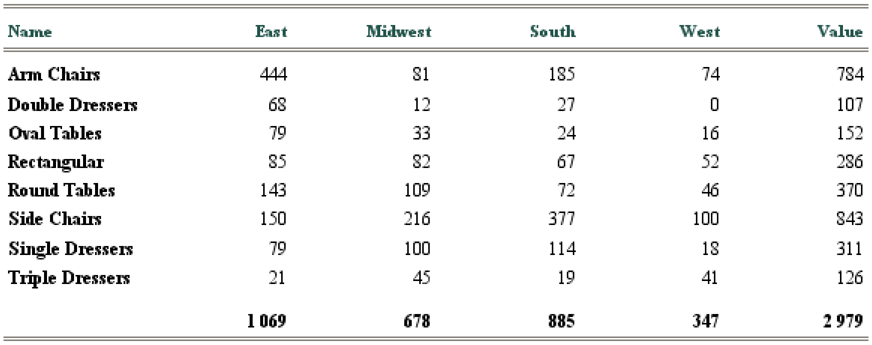
Region	Company	CategoryName	Total
Midwest	All Unfinished	Arm Chairs	$14,532.00
		Oval Tables	$27,884.00
		Rectangular Tables	$7,530.00
		Round Tables	$11,109.00
		Side Chairs	$9,903.00
		Single Dressers	$13,839.00
			$84,797.00

	Company	CategoryName	Total
	George Park	Arm Chairs	$5,424.00
		Oval Tables	$13,424.00
		Rectangular Tables	$5,416.00

Report with Chart Using Section Data (Midwest)

Keep in mind that if you select the *Include Section Data* option, the data available for the chart will vary between sections, so moving the chart between sections is not recommended. If you need to move the chart after it is made, make sure that the data mapped to the chart is also available from the location where you are moving the chart to.

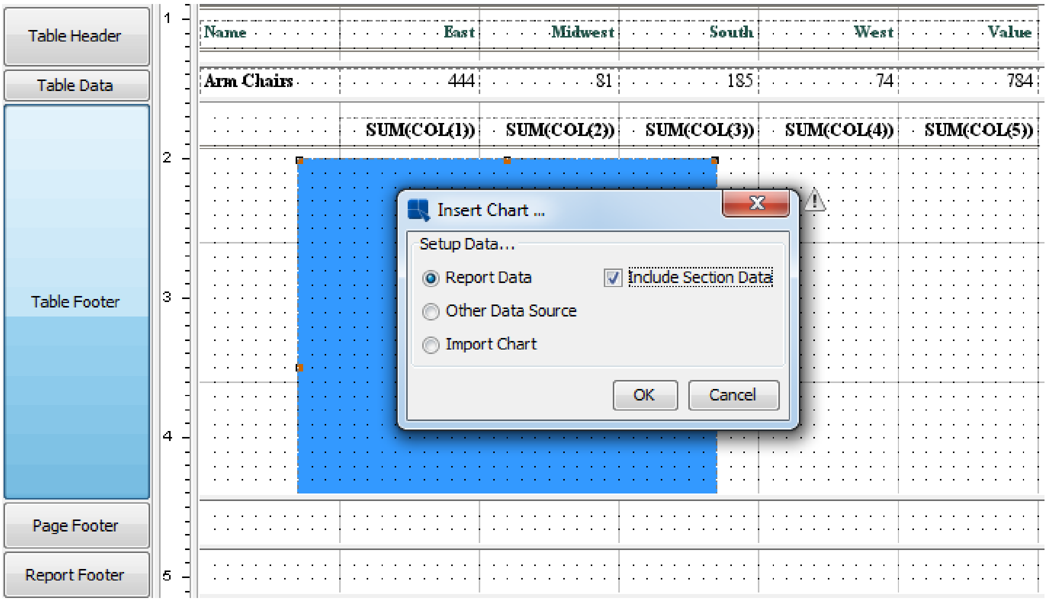
Here is another example using *Include Section Data*. Suppose you had the following CrossTab report and you wanted to use the aggregation for each column as data points in your chart.



Name	East	Midwest	South	West	Value
Arm Chairs	444	81	185	74	784
Double Dressers	68	12	27	0	107
Oval Tables	79	33	24	16	152
Rectangular	85	82	67	52	286
Round Tables	143	109	72	46	370
Side Chairs	150	216	377	100	843
Single Dressers	79	100	114	18	311
Triple Dressers	21	45	19	41	126
	1 069	**678**	**885**	**347**	**2 979**

CrossTab Report with Group Aggregation

To make the aggregations easier for later use, give each group aggregation a Custom ID that matches their column header. For example, give `SUM(COL(1))` the ID **`SumEast`**, give `SUM(COL(2))` **`SumMidwest`** and so on. For more information about Custom IDs, see Section 1.8.2.1 - Using Column Field Data. Then insert the chart in the Table Footer Section using Report Data and Include Section Data.



Name	East	Midwest	South	West	Value
Arm Chairs	444	81	185	74	784
	SUM(COL(1))	SUM(COL(2))	SUM(COL(3))	SUM(COL(4))	SUM(COL(5))

Insert Chart ...

Setup Data...

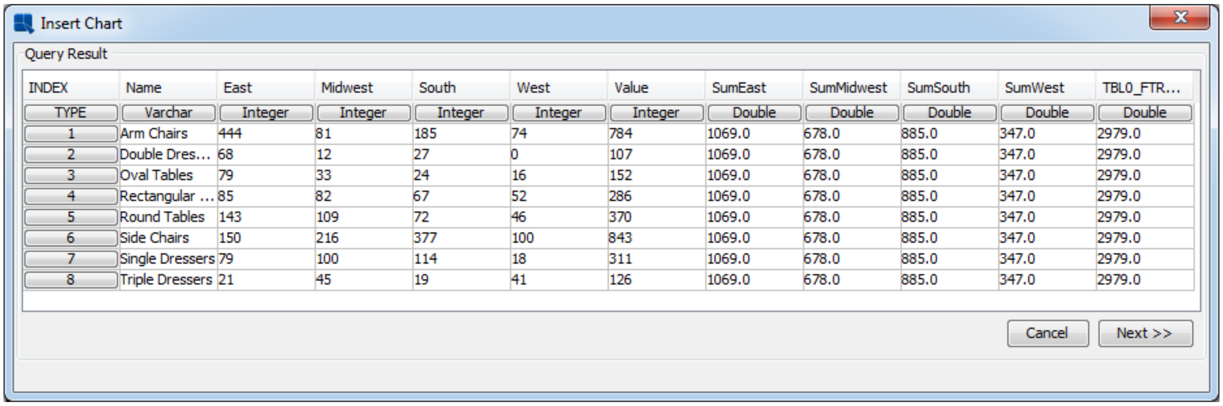
● Report Data ☑ Include Section Data

○ Other Data Source

○ Import Chart

Insert Chart in Table Footer

The formulas with custom IDs will be visible in the data source preview (if you selected the *Include section data* option).


Insert Chart

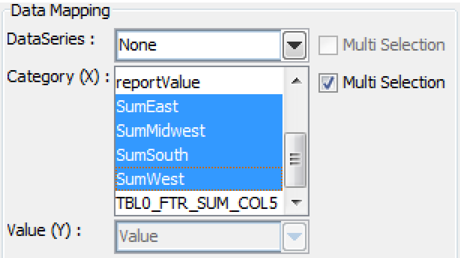
Query Result

INDEX	Name	East	Midwest	South	West	Value	SumEast	SumMidwest	SumSouth	SumWest	TBL0_FTR...
TYPE	Varchar	Integer	Integer	Integer	Integer	Integer	Double	Double	Double	Double	Double
1	Arm Chairs	444	81	185	74	784	1069.0	678.0	885.0	347.0	2979.0
2	Double Dres...	68	12	27	0	107	1069.0	678.0	885.0	347.0	2979.0
3	Oval Tables	79	33	24	16	152	1069.0	678.0	885.0	347.0	2979.0
4	Rectangular ...	85	82	67	52	286	1069.0	678.0	885.0	347.0	2979.0
5	Round Tables	143	109	72	46	370	1069.0	678.0	885.0	347.0	2979.0
6	Side Chairs	150	216	377	100	843	1069.0	678.0	885.0	347.0	2979.0
7	Single Dressers	79	100	114	18	311	1069.0	678.0	885.0	347.0	2979.0
8	Triple Dressers	21	45	19	41	126	1069.0	678.0	885.0	347.0	2979.0

Data source preview with section data

Since the data points are in different aggregations, you will need to use the transpose feature. To learn more about data transposition in charts, see the Section 3.4.1.1 - Data Transposition chapter.

In the data mapping window, select the *Multi Selection* option for the Category (X). This option allows you to select multiple columns for the category. Select all the formulas with custom IDs. The Value (Y) should be automatically set to *Value*.


Data Mapping

DataSeries : None □ Multi Selection

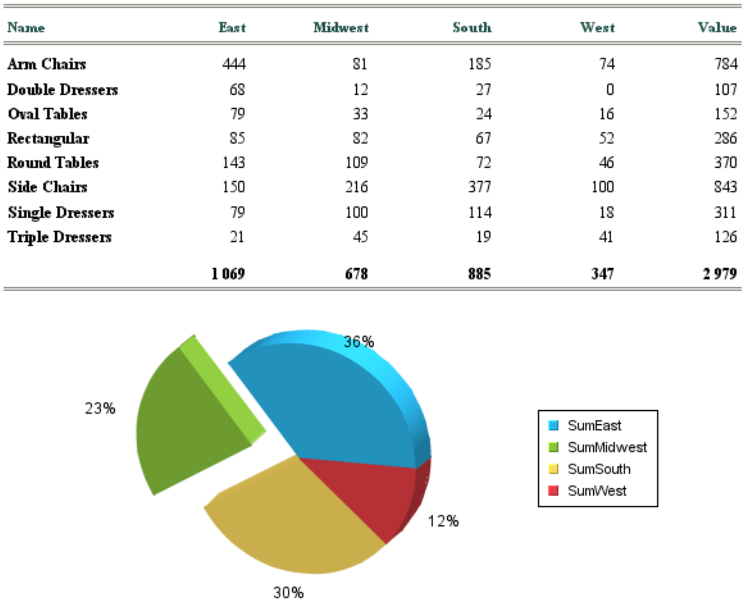
Category (X) : ☑ Multi Selection

reportValue
SumEast
SumMidwest
SumSouth
SumWest
TBL0_FTR_SUM_COL5

Value (Y) : Value

Chart Mapping for Transposed Data

Depending on the type of chart you select, the finished report might look like this:



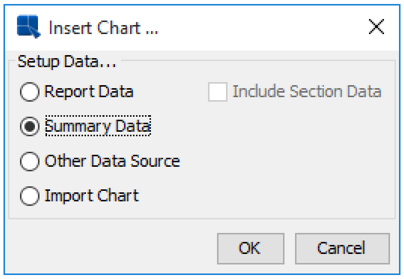
Name	East	Midwest	South	West	Value
Arm Chairs	444	81	185	74	784
Double Dressers	68	12	27	0	107
Oval Tables	79	33	24	16	152
Rectangular	85	82	67	52	286
Round Tables	143	109	72	46	370
Side Chairs	150	216	377	100	843
Single Dressers	79	100	114	18	311
Triple Dressers	21	45	19	41	126
	1 069	678	885	347	2 979

Report with Chart Using Section Data

1.5.7.1.1.1. Inserting Summary Charts into Crosstab Reports

You can also insert summary charts into crosstab reports. A summary chart is a type of chart that is created from summarized data of a fixed-field crosstab report. For more information about crosstab reports, please see Section 1.4.3 - Crosstab Report.

When you are going to insert a chart into a crosstab report, you will still have the option to use data from the current report or use an external data source. Otherwise, you will be given an option to select a summary data source for your chart or import a chart file created before. Note that a summary chart can only be inserted into the Table Header/Footer sections, depending on the Formula Position. For example, if you created a fixed-field crosstab report with the formula position set to be drawn in the Header section, you can only insert a summary chart into the Table Header section. If you try to insert a chart to other report sections, the *Summary Data* option will not be available in the Insert Chart dialog.



Insert Chart ...

Setup Data...

○ Report Data □ Include Section Data

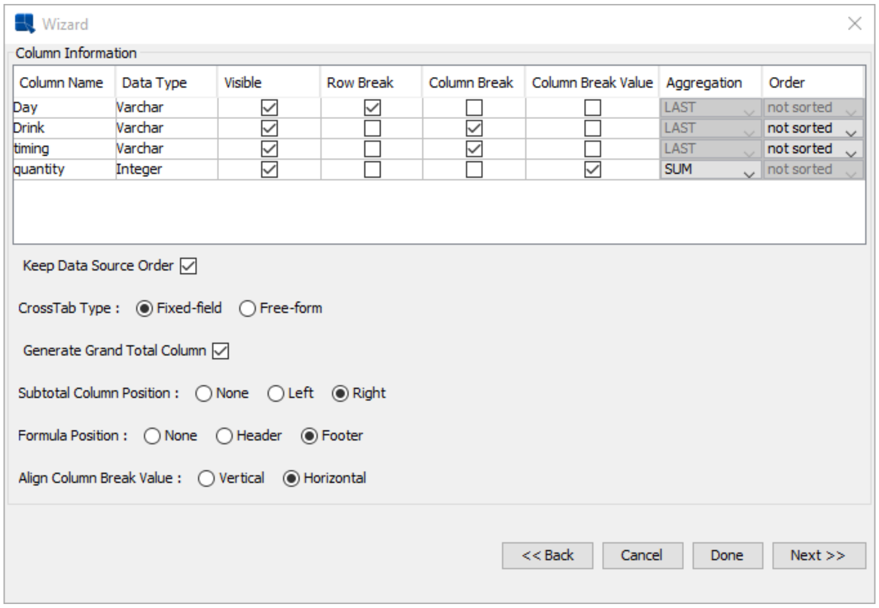
● Summary Data

○ Other Data Source

○ Import Chart

Insert Chart Dialog

For example, say we have the following data mapping for a crosstab report:



Column Information

Column Name	Data Type	Visible	Row Break	Column Break	Column Break Value	Aggregation	Order
Day	Varchar	☑	☑	□	□	LAST	not sorted
Drink	Varchar	☑	□	☑	□	LAST	not sorted
timing	Varchar	☑	□	☑	□	LAST	not sorted
quantity	Integer	☑	□	□	☑	SUM	not sorted

Keep Data Source Order ☑

CrossTab Type : ● Fixed-field ○ Free-form

Generate Grand Total Column ☑

Subtotal Column Position : ○ None ○ Left ● Right

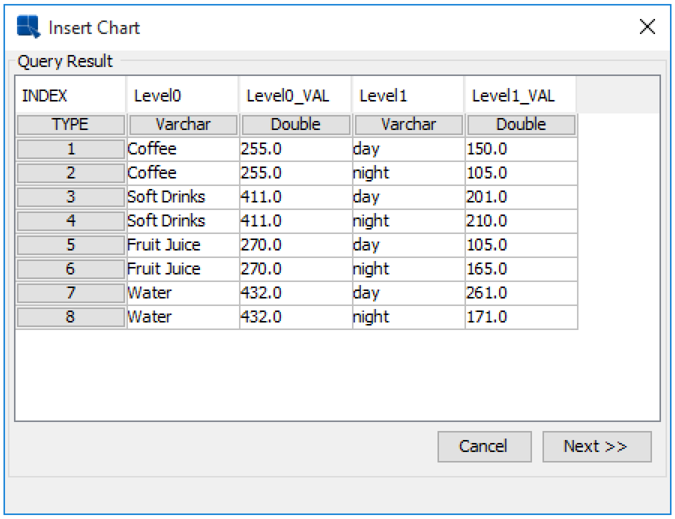
Formula Position : ○ None ○ Header ● Footer

Align Column Break Value : ○ Vertical ● Horizontal

Data Mapping

We selected `Day` as a *Row Break* field, `Drink` and `Timing` as *Column Break* fields, `Quantity` as a *Column Break Value* field with an aggregation of **`Sum`**. We also chose the *Formula Position* to be drawn in the Table Footer section of the report, which means we have to insert the summary chart into the Table Footer section in which data will be summarized.

Inserting a summary chart into the table footer section and selecting the **`Summary Data`** as datasource in the *Insert Chart* dialog will show the following Query Result dialog. The dialog shows data that will be used for a summary chart.

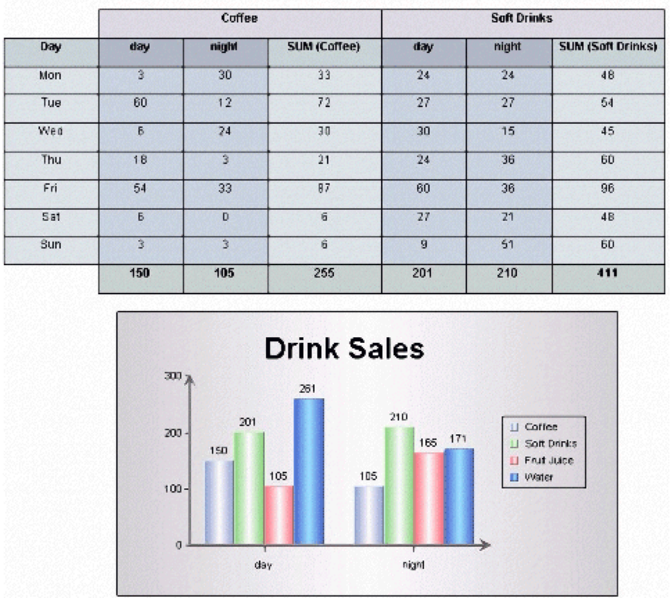


Query Result

INDEX	Level0	Level0_VAL	Level1	Level1_VAL
TYPE	Varchar	Double	Varchar	Double
1	Coffee	255.0	day	150.0
2	Coffee	255.0	night	105.0
3	Soft Drinks	411.0	day	201.0
4	Soft Drinks	411.0	night	210.0
5	Fruit Juice	270.0	day	105.0
6	Fruit Juice	270.0	night	165.0
7	Water	432.0	day	261.0
8	Water	432.0	night	171.0

Query Result

Depending on the type of chart and formatting you select, the finished report might look like this:

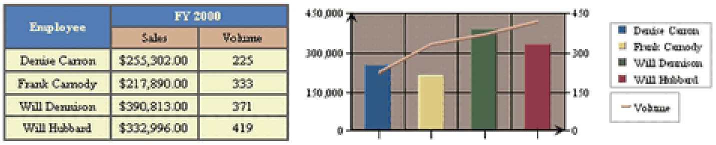


	Coffee			Soft Drinks		
Day	day	night	SUM (Coffee)	day	night	SUM (Soft Drinks)
Mon	3	30	33	24	24	48
Tue	60	12	72	27	27	54
Wed	6	24	30	30	15	45
Thu	18	3	21	24	36	60
Fri	54	33	87	60	36	96
Sat	6	0	6	27	21	48
Sun	3	3	6	9	51	60
	150	105	255	201	210	411

Report Preview

1.5.7.1.2. Inserting Elements into the Table Data Section

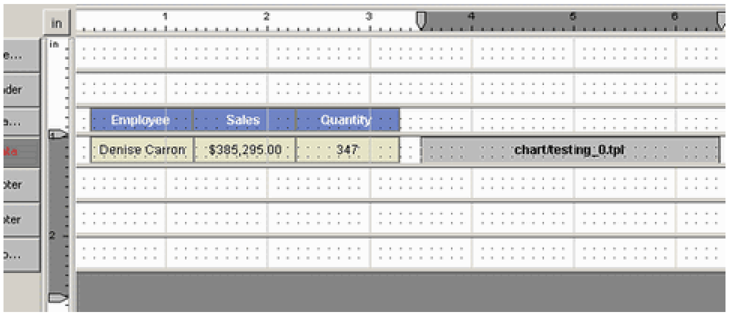
Generally, when elements are inserted into a report section, they will repeat each time that the report section repeats. However, this does not apply when certain elements are inserted into the Table Data section. Charts, images, and sub-reports do not repeat for each row of data (labels, column fields, and formulas do) when inserted into the Table Data section. This allows you to design a report in a side-by-side configuration where you might have a chart or in sub-report placed next to a data table as pictured below.



Employee	FY 2000	
	Sales	Volume
Denise Carron	$255,302.00	225
Frank Carmody	$217,890.00	333
Will Dennison	$390,813.00	371
Will Hubbard	$332,996.00	419

Side-by-Side Data Table & Chart

Normally, when you insert a chart, image, or sub-report into a report section, you will want to resize the section so that the entire element fits within the section (otherwise it is truncated). This is not the case for charts, images, or sub-reports inserted into the Table Data section. In this situation, you will want to keep the section the same height as your column fields regardless of the size of the element you're placing in the section.



Employee	Sales	Quantity	
Denise Carron	$385,295.00	347	chart/testing_0.tpl

Side-By-Side Data Table & Chart (Designer)

You can resize an element in the Table Data section using the rulers or by temporarily resizing the section to adjust the element size.

1.5.7.2. Editing Elements

You can edit elements in one of four ways: by selecting *Edit* from the *Edit* menu, clicking on the *Edit* button in the toolbar, right clicking on the element, and then selecting *Edit* from the pop-up menu, or by double clicking on the element.

Editing Labels: When you select the *Edit* option for a label a dialog box will appear prompting you to change the label text. Click on *OK* and the label will be changed.

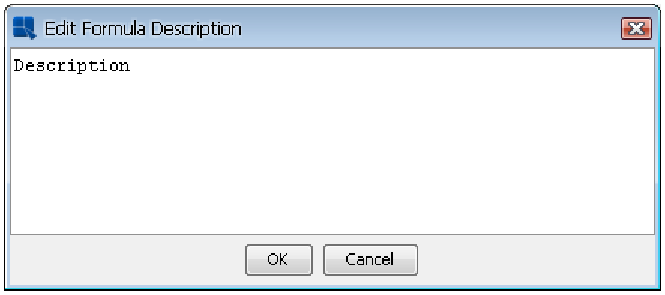
Editing Formulas: When you select the *Edit* option for a formula, the Formula Builder will appear prompting you to change the formula. Make any changes you want, then click the *OK* button and the formula will be changed.

> **Note**
>
> If you use the same formula in more than one place in the report, modifying the formula will change it everywhere. You can avoid this by clicking the *Save As* button in the Formula Builder. This will allow you to specify a new name for the modified formula and modify it only for the particular element.

Editing Charts: When you select the *Edit* option for a chart, the Chart Designer will open, allowing you to edit and format the chart.

Editing Images: When you select the *Edit* option for an image, a new window appears prompting you to select a new image or directory URL. After selecting the new image, click the *OK* button and the new image will replace the old one.

Editing Column Headers: Column header is a unique type of report element that is generated when you first create the report. By default, column headers are dynamic and will display the name of the column even when the column changes (by changing data mapping, source, or applying a template). When you select to edit a column header, a dialog box will appear prompting you to change the text of the header.



Edit Formula Description

Description

Edit Column Header Dialog

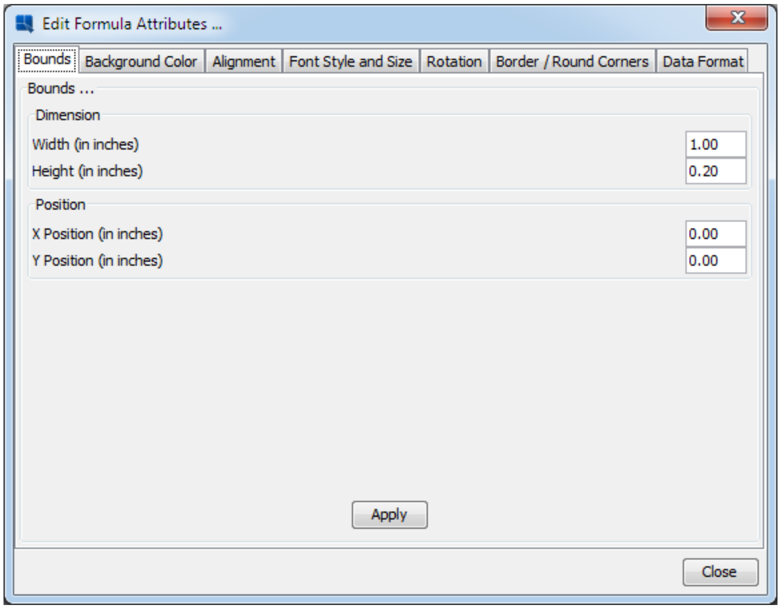
When you edit the header, a new text will be displayed. However, the header will no longer be dynamic. You can return the header to display the (dynamic) column name by right clicking on it and selecting *Original Column Header* from the pop-up menu.

You can also copy and paste cell attributes between report elements. This will apply all attributes including data formatting (assuming the data types are the same), alignment, font, color, and border attributes, as well as the bounds of a cell.

To copy the attributes of an element, right click on the element and select *Element Appearance* from the pop-up menu. This will expand into two additional choices. Select *Copy* from the second menu. To apply the copied attributes to another cell, repeat the same steps as before and select *Paste* from the secondary menu.

1.5.7.2.1. Editing Element Attributes

In addition to directly editing an element, you can directly access all of the element's attributes by selecting *Edit Attributes* option from the *Format* menu, or by right clicking on an element and selecting *Edit Attributes* from the pop-up menu. This will bring up a tabbed dialog that allows you to set a number of properties for the element at once.



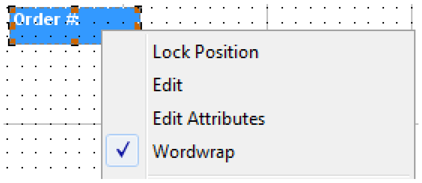
Width (in inches) 1.00
Height (in inches) 0.20
X Position (in inches) 0.00
Y Position (in inches) 0.00

Multi-Attribute Editing Dialog

The options will vary depending on what type of element you select. For each tab in the dialog, you can set the option you want and then click the *Apply* button to set the changes for that attribute.

1.5.7.2.2. Word Wrapping

By default, any text within report elements, including labels, formulas, and columns will wrap to the next line if the cell boundaries are not wide enough to fit the text. However, this behavior can be disabled. To set text wrapping, select an element and select Format → Wordwrap, or right click on an element and select the option from the pop-up menu. This will disable/enable text wrapping.

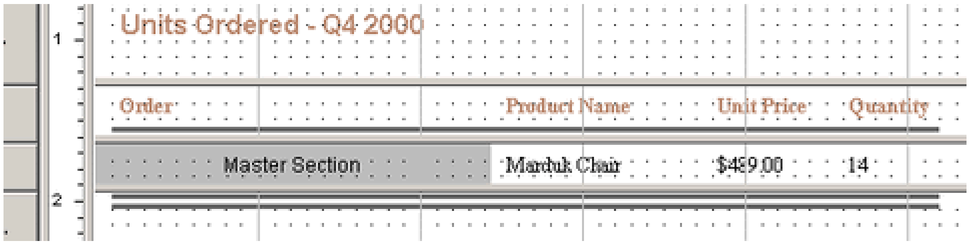


Word Wrapping Dialog

To disable word wrapping, un-check the option. Note that when word wrapping is disabled, the entire contents of the cell will be printed regardless of the cell boundaries. This can overlap and obscure other report elements. Also, word wrapping cannot be disabled for rich text fields as the rich text and Excel-based exports will still display the wrapped text.

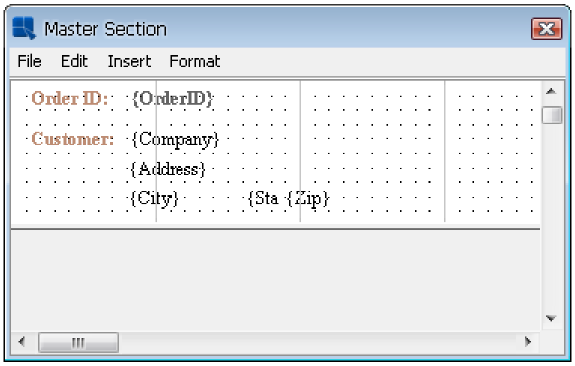
1.5.7.2.3. Editing Side-By-Side Master & Details Reports

As noted in Section 1.4.4 - Master & Details Report of this guide, you can specify a side-by-side layout for master & details reports. This will print the master field (group header) next to the details section (table data), rather than its traditional position above the details section. When you select this report format, the master section will appear as a gray rectangle next to the column fields within the table data section of the report.



Side-by-Side Master & Details

The master section can be moved in free-form like any other report element. You can edit the contents of the master field in one of three ways: selecting it and clicking the *Edit* button on the toolbar, right clicking it and selecting *Edit* from the pop-up menu, or double clicking it. The master section will open in a new window, allowing you to modify its contents.



Edit Master Section Dialog

The elements in this section can be moved, resized, and edited in the same way as other elements in the report. The master section acts the same as a group header section. It repeats for each unique value in the primary key column. After you finish editing the contents of the master section, it can be closed by selecting *Close* from the *File* menu.

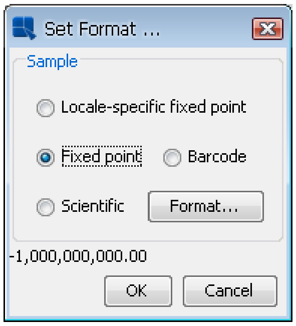
Note

After you finish editing the master section, it will automatically resize to fit its contents.

1.5.7.3. Data Formatting for Formulas and Column Fields

You can change the output of formulas and column fields in one of three ways: clicking the %$ *Data Format* icon in the toolbar, selecting *Data Format* from the *Format* menu, or right clicking on a formula or column field and selecting *Edit Attributes* from the pop-up menu. When you select the format option, a dialog box will appear (or the format tab of the multi-attribute editing dialog). The box that appears depends on what type of data is present in the selected element: numeric, string, date/time, or logical/boolean.

Formatting Numeric Data: The dialog box for numeric data contains four primary options for the data: locale-specific fixed point, fixed point, bar code, and scientific. You can select the option you want and then click on format for additional options.



○ Locale-specific fixed point

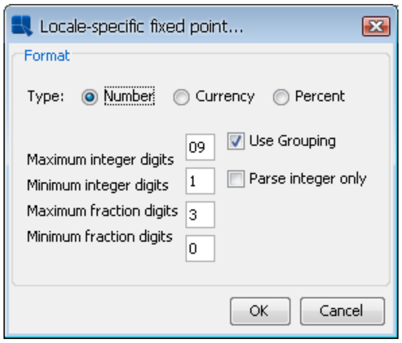
● Fixed point ○ Barcode

○ Scientific

-1,000,000,000.00

Numeric Format Dialog

Locale-Specific Fixed Point: This will change the data format depending on the locale in which it is being viewed. Additional formatting for this option allows you to specify whether the data should be displayed as a number, currency, or percentage. Additionally, you can set the maximum and minimum number of integer digits and fraction digits. Other display attributes will vary depending on locale.



Type: ● Number ○ Currency ○ Percent

Maximum integer digits 09

Minimum integer digits 1

Maximum fraction digits 3

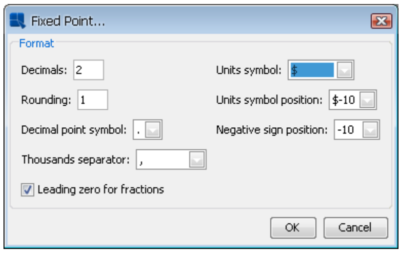
Minimum fraction digits 0

☑ Use Grouping

□ Parse integer only

Locale-Specific Formatting

Fixed Point: This will keep the data format consistent, regardless of locale. Additional formatting for this option allows you to set the number of decimals, rounding for digit number, unit symbols, negative sign positions, decimals and thousands separator, and specify leading zeroes for fractions.



Decimals: 2

Rounding: 1

Decimal point symbol: .

Thousands separator: ,

☑ Leading zero for fractions

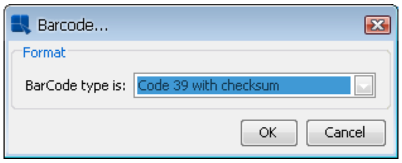
Units symbol: $

Units symbol position: $-10

Negative sign position: -10

Fixed Point Formatting

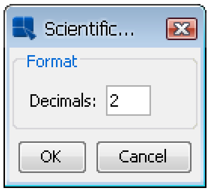
Bar Code: This will convert the data into a bar code. Additional formatting for this option allows you to select the symbology to use for the bar code. Supported symbologies are Code 39 with and without checksum, Interleave 2 of 5, UPC A, EAN 13, EAN 128, Standard 2 of 5 with and without check digit, Code 128, Code 128A, Code 128B, Code 128C, USD 3 with/without checksum, Code 3 of 9 with and without checksum, Global Trade Item Number, Random Weight UPCA, SCC 14 Shipping Code, Shipment Identification Number, SSCC 18, and US Postal Service.



BarCode type is: Code 39 with checksum

Bar Code Formatting

Scientific: This will display the data in scientific notation. Additional formatting for this option allows you to set the number of decimals.

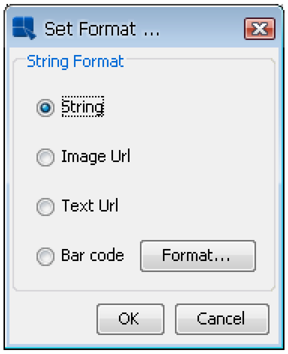


Decimals: 2

Scientific Formatting

After you finish selecting additional options, click the *OK* button to return to the main dialog box. Click the *OK* button and the data format will be changed.

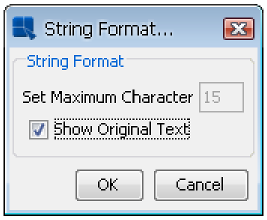
Formatting String Data: The dialog box for string data contains four primary options: string, image URL, text URL, and bar code. You can select the option you want and in the case of string or bar code, click *Format* for additional options.



● String
○ Image Url
○ Text Url
○ Bar code

String Data Format

String: This dialog allows you to format the appearance of the string. The checkbox labeled *Show Original Text* controls whether the complete string for each data entry will be displayed. If you un-check it, you can then specify the maximum number of characters to be displayed. Click the *OK* button to return to the previous dialog.



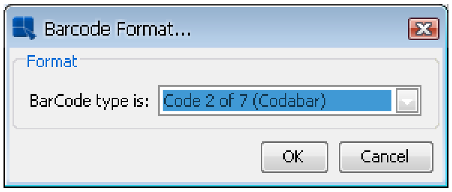
Set Maximum Character 15
☑ Show Original Text

String Formatting

Image URL: This will convert the string into an image. This is used in situations where instead of storing an image as an object in a database (BLOB), you have stored URLs that point to image files on a server. Selecting this option will read the URL and retrieve the images to be placed in the report.

Text URL: This will convert the string into a large text object. This is used in situations where instead of storing a large text file as an object in a database (CLOB), you have stored URLs that point to files on a server. Selecting this option will read the URL and retrieve the text files to be placed in the report.

Bar Code: This will convert the data into a bar code. Additional formatting for this option allows you to select the symbology to use for the bar code. Available symbologies for string data are Code 39 with/without checksum, Code 2 of 7 (Codabar), Codabar, EAN 128, Code 128, Code 128A, Code 128B, Code 128C, Global Trade Item Number, Monarch, NW 7, SCC 14 Shipping Code, Shipment Identification Number, SSCC 18, US Postal Service, and USD 4.



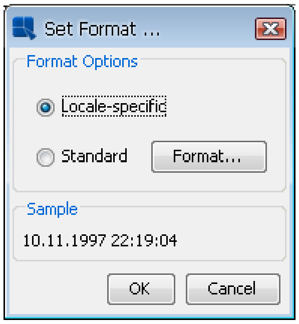
BarCode type is: Code 2 of 7 (Codabar)

Bar Code Formatting

Note

Codabar will not accept start/stop characters in the data. If the input data is incorrect, the bar code may not be readable.

Formatting Date/Time Data: The dialog box for date/time data contains two primary options for the data: locale-specific and standard. You can select the option you want and click the *Format* button for additional options. The available additional options will vary depending on the nature of your data. Date, time, and timestamp data will bring up date, time, and date & time options respectively.



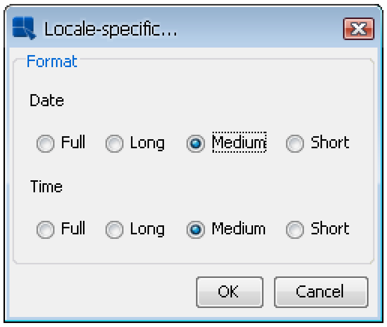
● Locale-specific

○ Standard

10.11.1997 22:19:04

Date/Time Data Format

Locale Specific: This will change the format of the data depending on the locale in which it is being viewed. Additional formatting for this option allows you to select full, long, medium, or short notations for date and time information. Other display attributes will vary depending on locale.



Date

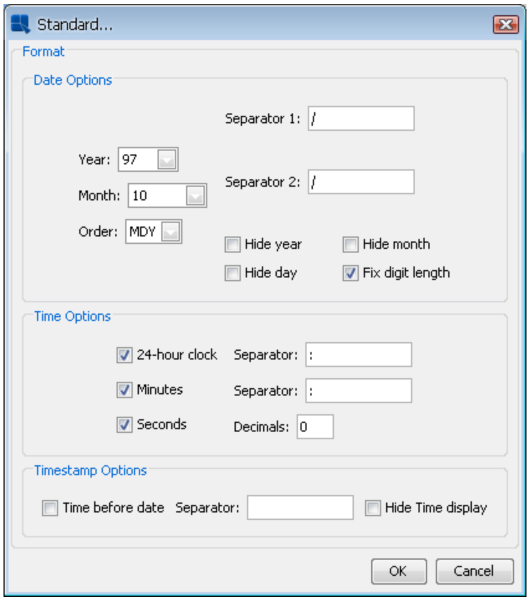
○ Full ○ Long ● Medium ○ Short

Time

○ Full ○ Long ● Medium ○ Short

Locale-Specific Formatting

Standard: This will keep the data format consistent, regardless of locale. Additional formatting for this option allows you to select year and month displays, as well as the order in which month, day, and year information is presented. You can also select the characters to be used as separators. Time options allow you to display hours, minutes, and/or seconds, and select the separators between them. For timestamp data, you can select to display the time before or after the date and the separator to be used between them.



Standard...

Format

Date Options

Separator 1: /

Year: 97

Month: 10

Order: MDY

Separator 2: /

□ Hide year □ Hide month

□ Hide day ☑ Fix digit length

Time Options

☑ 24-hour clock Separator: :

☑ Minutes Separator: :

☑ Seconds Decimals: 0

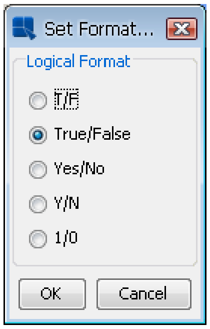
Timestamp Options

□ Time before date Separator: □ Hide Time display

Standard Formatting

After you finish selecting additional options, click the *OK* button to return to the main dialog box. Click the *OK* button and the data format will be changed.

Formatting Logical Data: The dialog box for Logical or Boolean data contains five options for displaying the data. They are: T/F, True/False, Yes/No, Y/N, and 1/0. Select the format you want to use and then click the *OK* button to change the data format.



Set Format...

Logical Format

○ T/F

● True/False

○ Yes/No

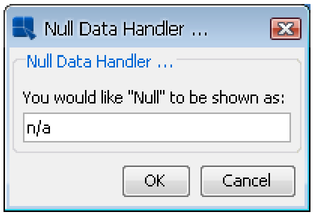
○ Y/N

○ 1/0

Boolean Data Format

1.5.7.3.1. Formatting Null Data

By default, null data will display in a report as `Null`. You can change the appearance of null data by selecting *Null Data Handler* from the *Option* menu. This will bring up a dialog, prompting you to specify a string to be displayed for nulls.



You would like "Null" to be shown as: n/a

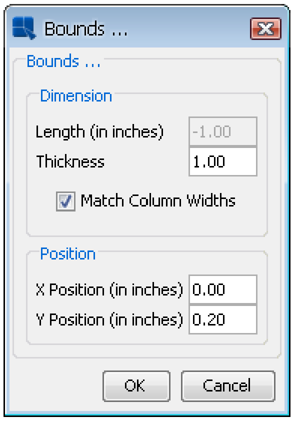
Null Data Handler

Enter a value you want and click the *OK* button. All the null values in the report will be displayed as the value you specified.

1.5.7.4. Line/Rectangle Format

You can format the appearance of lines and rectangles in the report by right clicking on a line or rectangle element. The pop-up menu will contain a list of available options.

Lines: There are three formatting options for lines. *Line Color & Thickness* allows you to set the color and thickness of the line. *Line Style* allows you to select the style of line - either solid, dotted, or double. *Set Bounds* allows you to specify the length and thickness of the line, as well as the X and Y coordinates of its origin. In addition, you can set the width of horizontal lines to match the aggregate width of columns in the report. This feature is useful for crosstab reports where the number of columns can vary and also for user-defined report styles. For vertical lines, you can set the length to match the section height.



Length (in inches) -1.00
Thickness 1.00
☑ Match Column Widths
X Position (in inches) 0.00
Y Position (in inches) 0.20

Bounds Dialog for Horizontal Lines

Rectangles: There are four formatting options for grid rectangles. *Border* allows you to set the border thickness and color for the grid rectangle. *Background* allows you to set a background color for the fill area of the rectangle. *Set Bounds* allows you to specify the width and height of the grid rectangle, the X and Y coordinates of its origin, as well as the degree of rounding for the corners. *Border Style* allows you to select the line style of the border - either solid, dotted, or double.

1.5.7.5. Chart Export Format

This option allows you to specify attributes for charts exported to DHTML. You can format the chart export properties by selecting *Chart Export Format* from the *Format* menu or by right clicking on a chart and selecting *Export Format* from the pop-up menu. When you select this option, a dialog box will appear. You can select GIF, PNG, JPEG, or FLASH for the image type and set some options for each type. For GIF images, you can set the background to be transparent. For PNG images, you can set the compression. For JPEG images, you can set the quality (Image quality is directly proportional to file size). For FLASH images, you can enable animation and set the frame count and frame rate.

You can also specify to create an image map when the report is exported. If you create an image map, the hyperlinks defined in the chart will be active when the report is exported. If no hyperlinks are defined, the map will contain display pop-up labels for the chart containing data point information. Note that FLASH images do not support image maps.

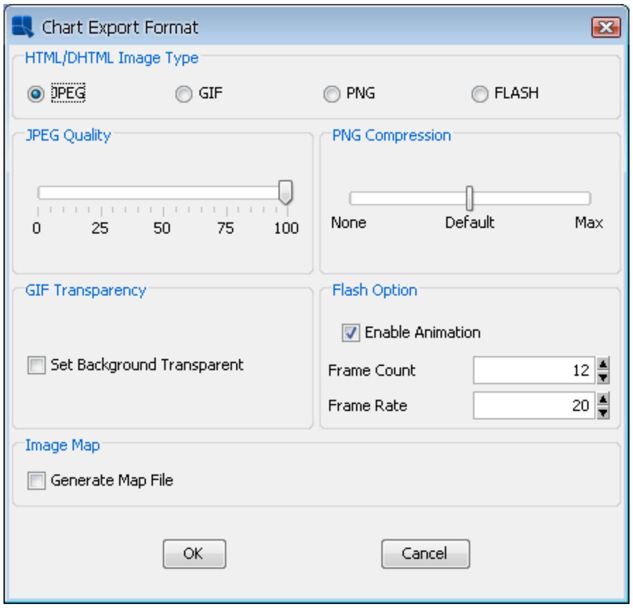


Chart Export Format

HTML/DHTML Image Type

● JPEG ○ GIF ○ PNG ○ FLASH

JPEG Quality

PNG Compression

GIF Transparency

□ Set Background Transparent

Flash Option

☑ Enable Animation

Frame Count 12

Frame Rate 20

Image Map

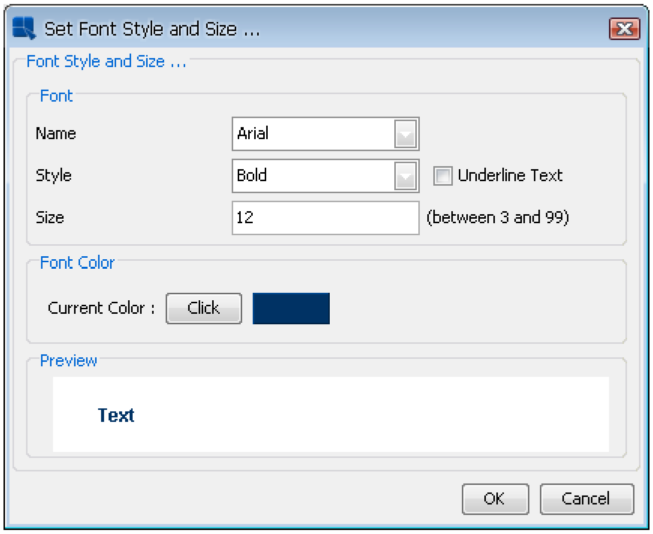
□ Generate Map File

Chart Export Format

1.5.7.6. Font, Color, and Border Options

You can change the appearance of report elements by changing the font, color, and border properties.

Font Style and Size: The font style and size can be adjusted for labels, formulas, and column fields in one of three ways. You can directly modify the font style and size using the options in the Report Designer toolbar. In addition, you can also select *Font Style and Size* from the *Format* menu or right click on a report element and select *Edit Attributes* from the pop-up menu. The latter two options will invoke a separate dialog prompting you to select the font style and font size or in the attribute editing dialog, where font options can be set in the 'Font Style and Size' tab.



Set Font Style and Size ...

Font Style and Size ...

Font

Name Arial

Style Bold □ Underline Text

Size 12 (between 3 and 99)

Font Color

Current Color : Click

Preview

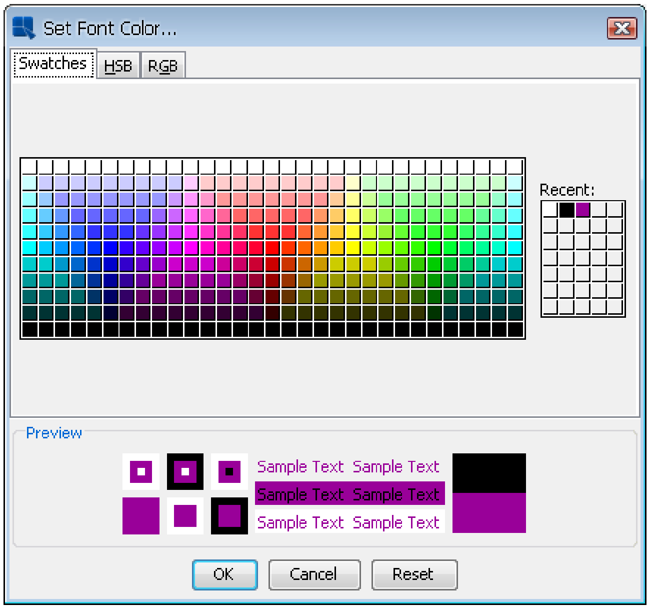
Text

Font Style and Size Dialog

> **Note**
>
> You can use any fonts from the system when designing a report. However, these fonts may no longer be present when you move reports between platforms. Also, you will have to explicitly map font files (.ttf) to font styles when exporting a report to PDF. For more information about this, please see Section 1.7.2.1 - PDF Font Mapping.

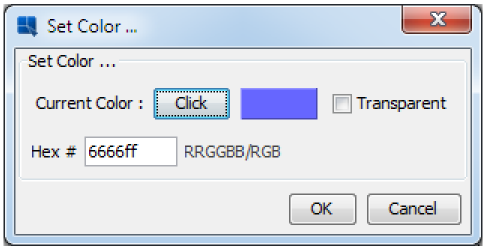
Alignment: The alignment for labels, formulas and column fields can be adjusted in one of three ways. You can directly adjust the horizontal alignment for an element by selecting the option(s) from the toolbar. You can also select *Alignment* from the *Format* menu, or right click on the element and select *Edit Attributes* from the pop-up menu. The latter two options also allow you to select vertical as well as horizontal alignment for the element text.

Font Color: The font color can be adjusted for labels, formulas, and column fields in one of three ways: selecting *Font Style And Size* from the *Format* menu, clicking the *Font Style And Size* button on the toolbar, or right clicking on the element and selecting *Edit Attributes* from the pop-up menu. Selecting this option will bring up a dialog box (or the multi-attribute editing dialog where options can be set in the *Font Style And Size* tab). To change the font color, click the *Click* button next to the *Current Color:* field. The Set Font Color dialog will appear allowing you to select font color from swatches. You can also enter HSB, HEX, or RGB values.



Font Color Dialog

Background Color: The background color can be adjusted for labels, formulas, and column fields in one of three ways: selecting *Background Color* from the *Format* menu, clicking the *Background Color* button on the toolbar, or right clicking on the element and selecting *Edit Attributes* from the pop-up menu. Selecting this option will bring up a dialog box (or the multi-attribute editing dialog where options can be set in the *Background Color* tab) prompting you to specify whether or not to set the background transparent. If you do not want a transparent background, uncheck the checkbox and click the button. This will bring up a dialog prompting you to specify the background color. You can select font color from swatches, or enter HSB or RGB values. You can also directly enter HEX color value to the *Hex #* field.



Set Color ...

Set Color ...

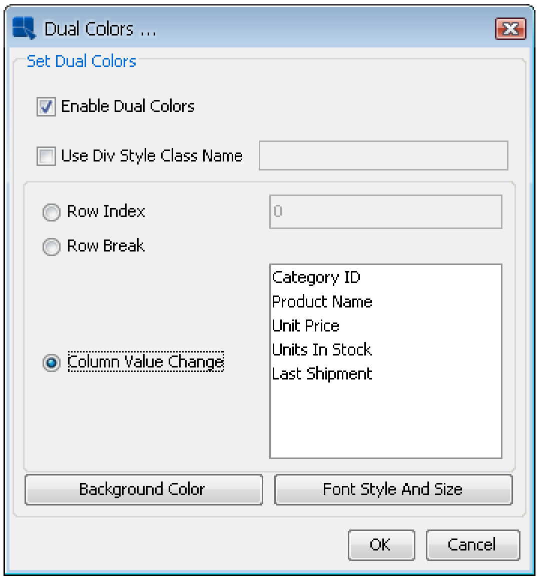
Current Color : Click □ Transparent

Hex # 6666ff RRGGBB/RGB

Background Color Dialog

Dual Colors: You can specify alternating color for labels, formulas, or column fields that are in the data table section of the report in one of three ways: selecting *Dual Colors* from the *Format* menu, clicking the *Dual Color* button on the toolbar, or right clicking on the element and selecting *Edit Attributes* from the pop-up menu. Selecting this option will bring up a dialog box (or the multi-attribute editing dialog where options can be set in the *Dual Colors* tab), allowing you to specify the number of rows between alternating colors. Instead of specifying alternate row numbers, you can also set dual colors to change on the row break (for summary break or crosstab reports) or as a particular column value changes. Using this feature, you can set the cell attributes to change when you reach a new group in a summary break report.

For alternate rows, you can set the background color, the font color, as well as the font style and size. Set the primary attributes as you would for any other report element.



Dual Colors ...

Set Dual Colors

☑ Enable Dual Colors

□ Use Div Style Class Name

○ Row Index 0

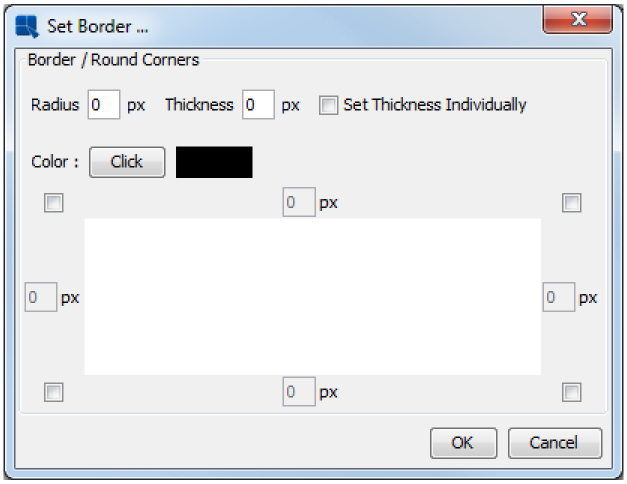
○ Row Break

● Column Value Change

Category ID
Product Name
Unit Price
Units In Stock
Last Shipment

Dual Colors Dialog

Border: You can specify the thickness and color of the border to be drawn around any report element in one of three ways: selecting *Borders* from the *Format* menu, selecting the *Border* button on the toolbar, or right clicking on the element and selecting *Edit Attributes* from the pop-up menu. Selecting this option will bring up a dialog box (or the multi-attribute editing dialog where options can be set in the *Border/Round Corners* tab) prompting you to enter the border thickness in pixels and to choose a color. Entering **0** as the border thickness value will remove the border.



Radius 0 px Thickness 0 px □ Set Thickness Individually

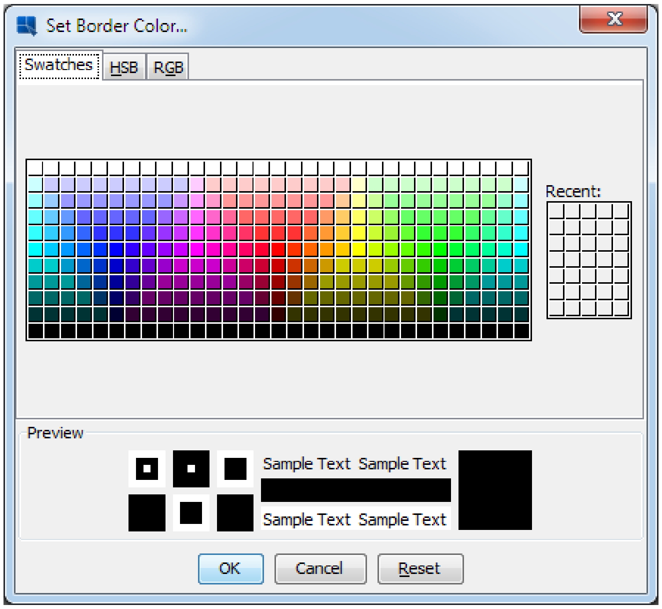
□ 0 px □

0 px 0 px

□ 0 px □

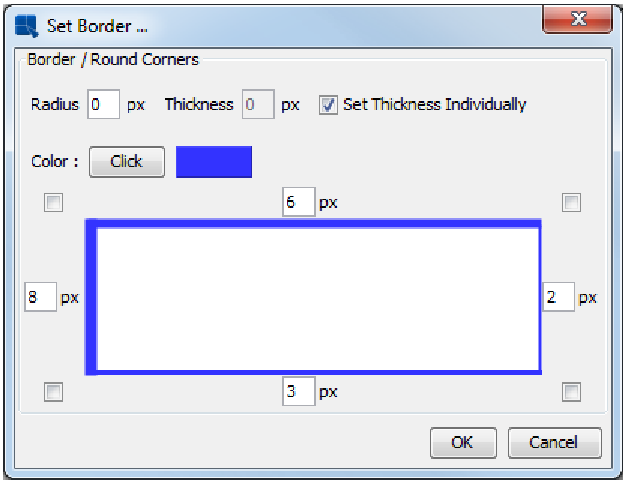
Border Dialog

To change the border color, click the *Click* button next to the *Color:* field. The *Set Border Color* dialog will appear allowing you to select border color from swatches. You can also enter HSB, HEX, or RGB values.



Border Color Dialog

To set the border thickness of the each side of the report element, check the *Set Thickness Individually* option. Then type the thickness (in pixels) in the text fields near the sides of the rectangle representing a selected element.



Radius 0 px Thickness 0 px ☑ Set Thickness Individually

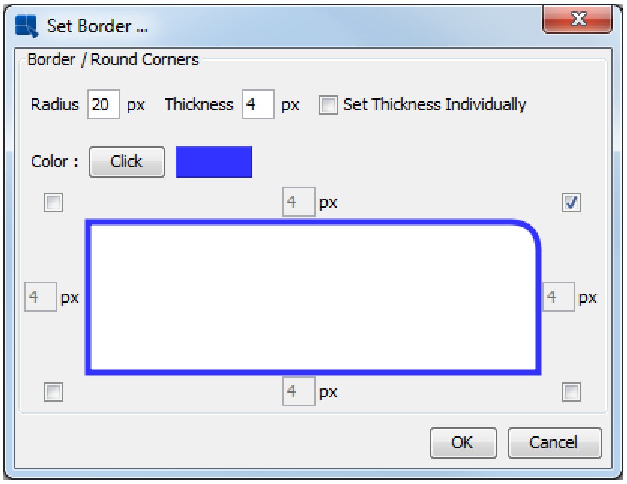
□ 6 px □

8 px 2 px

□ 3 px □

Border / Individual Thickness

To set round corners to an element, the elements need to have either non-transparent background color or a visible border, otherwise you will not see any changes. There are four checkboxes near the corners and each checkbox represents an element corner. To make a corner round, check its checkbox (you can also select several corners at once), then type a radius (in pixels) into the *Radius* text field.



Radius 20 px Thickness 4 px □ Set Thickness Individually

□ 4 px ☑

4 px 4 px

□ 4 px □

Border / Round Corners Dialog

Note

If you plan on exporting reports with round corners, please note that round corners are available for DHTML and PDF export formats only.

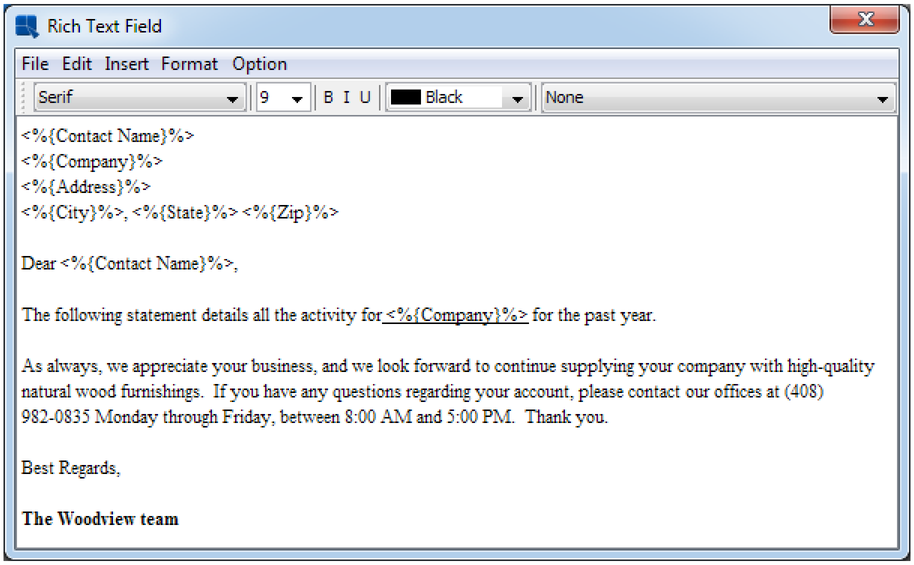
1.5.7.6.1. Show Cell Outline

Sometimes you may want to show the boundaries or borders of report elements in the Design window, without actually turning on the borders. This is especially important if you have blank place-holder cells or columns where the first record is empty. To turn on the cell boundaries, select *Show Cell Outline* from the Option menu. This will draw a gray dotted border around each report element in the Design window.

1.5.7.7. Rich Text Fields

One of the common reporting needs is the ability to create form letters or other blocks of formatted text to a report and merge in data fields and functions. This functionality is available using rich text fields. Rich text fields, unlike labels or string function fields, allow for complex paragraph, font, and color formatting within the cell. They also allow the ability to embed column fields and functions directly in the text flow.

To add a rich text field to a report, select *Insert Rich Text Field* from the Insert menu, or click the *Rich Text Field* button on the toolbar. After you select this option, a small box will follow your mouse pointer around the design window. Position the box where you want to insert the field and click. A new window will open, allowing you to enter the rich text field.



<%{Contact Name}%>
<%{Company}%>
<%{Address}%>
<%{City}%>, <%{State}%> <%{Zip}%>

Dear <%{Contact Name}%>,

The following statement details all the activity for <%{Company}%> for the past year.

As always, we appreciate your business, and we look forward to continue supplying your company with high-quality natural wood furnishings. If you have any questions regarding your account, please contact our offices at (408) 982-0835 Monday through Friday, between 8:00 AM and 5:00 PM. Thank you.

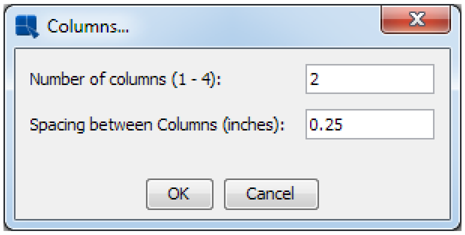
Best Regards,

The Woodview team

Rich Text Editor

The rich text editor works like a small word processor. The first drop-down box allows you to select the font and the font size. The three buttons allows you to specify bold, italic, or underlined text. The next drop-down-box allows you to specify font color. The last one allows you to select function or column field formats.

You can specify a columnar layout for the text by selecting *Columns* from the *Option* menu. This will bring up a dialog allowing you to specify the number of columns for the rich text field, as well as the spacing between the columns.



Number of columns (1 - 4): 2

Spacing between Columns (inches): 0.25

Rich Text Column Options Dialog

You can add in-line images to the text flow in rich text fields as well. To add an image, select *Insert Image* from the *File* menu. This will bring up a dialog prompting you to specify the image file you want to import. After you specify it, a new dialog will open, allowing you to specify the size for the in-line image in pixels. After you set the size, click the *Ok* button and the image will be added to the rich text field.

You can import any rich text file (.rtf format) into a rich text field. To do this, select *Import* from the *File* menu in the editor. This will bring up a dialog prompting you to specify the file you want to import.

Note

Certain paragraph settings will be lost when you import a rich text file.

1.5.7.7.1. Adding Formulas

You can add any formula directly into the text flow in a rich text field. It is also possible to add column fields and parameter values. To insert a formula, column field or a parameter value, click on the *Insert* menu, select a category (column, formula or parameter), and choose an item from the list of all available fields/parameters/formulas. The following syntax will be added into the text: *<% formula name/parameter name/column field name %>*.

The Insert → Formula list contains all formulas from the report. You can also create a custom formula directly in the rich text. To do so, simply type the *<%Formula Syntax%>* syntax (replace the *Formula Syntax* by the actual formula syntax). For more information about formula syntax, please see Section 1.8 - Using Formulas & the Formula Builder. This approach is recommended for simple formulas only. If you want to insert a more complex formula, you can create it in the report using the Formula Builder and then add it to the rich text field by selecting it from the Insert → Formula menu. See Section 1.8 - Using Formulas & the Formula Builder chapter to learn how to create formulas in the Formula Builder.

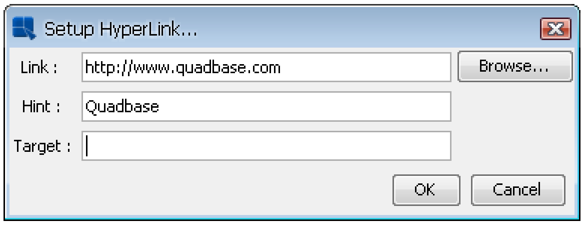
To format formulas and column fields, you must first pre-define a format. To do this, select *New* from the editor *Format* menu and select the data type for which you want to create a format. This will bring up the formatting dialog for that data type, allowing you to define the format. After you close the formatting dialog, you will be prompted to specify a name for the formatting. For example, you could define a format called *Currency* that uses locale-specific currency formatting for numbers.

To apply a pre-defined format of a formula, select the formula and then select the format that you want to apply from the drop-down box on the right side of the rich text editor.

1.5.7.8. Hyperlinks

You can apply a hyperlink to any report element except column fields. To apply a hyperlink to an element, select it and then select *HyperLink* from the *Format* menu or right click on the element and select *HyperLink* from the pop-up menu. A dialog box will then pop-up asking you to enter the link, hint, and target. You can enter any URL as the link or another `.pak` file. The hint will display text in a hint box on mouse over when the report is viewed.

If you are running EspressReport in stand-alone mode, then hyperlinks to URLs will not work from the Preview window. If you export to DHTML or PDF, links to `.pak` files will no longer work.



Link : http://www.quadbase.com

Hint : Quadbase

Target :

Hyperlink Dialog

You can also apply dynamic hyperlinks to report elements, including column fields, by using cell scripting. For more information about cell scripting, please see Section 1.9 - Scripting.

1.5.7.9. Column Wrapping

This feature is useful for reports that have few or narrow columns of data that are longer than one page. Rather than have the data take up only a portion of the page width and extend over multiple pages in length, you can 'wrap' the columns so that they will continue to the right of the original columns on the first page of the report.

For example, say you have the following report listing product names, and the number of units ordered:

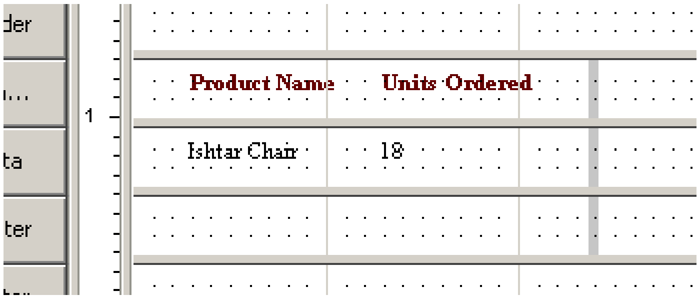
Product Name	Units Ordered
Adad Chair	4
Addad Dresser	5
Amon Table	2
An Chair	5

Product Name	Units Ordered
Anahita Dresser	3
Anubis Table	7
Apep Table	10
Apsu Dresser	3
Asherat Dresser	2
Atun Table	4
Bast Table	4
Bes Table	3
Cula Chair	10
Enki Chair	12
Enlil Chair	7
...	...
...	...

Assuming this report is long enough to encompass multiple pages, it is ideal to use column wrapping because it uses fewer pages and fills the blank space in the page width. The report will look like this with column wrapping:

Product Name	Units Ordered	Product Name	Units Ordered
Adad Chair	4	Bast Table	4
Addad Dresser	5	Bes Table	3
Amon Table	2	Cula Chair	10
An Chair	5	Enki Chair	12
Anahita Dresser	3	Enlil Chair	7
Anubis Table	7	...	...
Apep Table	10	...	...
Apsu Dresser	3		
Asherat Dresser	2		
Atun Table	4		

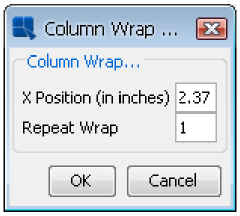
Column wrapping is available for simple columnar and summary break report types. To implement column wrapping, select *Column Wrap* from the *Format* menu. A gray vertical bar will be drawn across the middle of the Design window indicating the column wrapping placement.



Column Wrapping in Design Window

Column wrapping begins immediately to the right of the bar, so be sure to allow adequate space. Also, anything to the right of the column wrap bar will be truncated.

You can edit the column wrap properties by first selecting the column wrap bar and then selecting *Column Wrap* from the Format menu. You can also right click on the column wrap bar and then select *Column Wrap* from the pop-up menu. A dialog box will then appear allowing you to edit the properties of the column wrapping.



X Position (in inches) 2.37
Repeat Wrap 1

Column Wrapping Options

From this dialog, you can specify the X position of the column wrapping, as well as the number of times you want the columns to wrap in the page. The X position is the distance (in inches or centimeters) from the right side of the page where column wrapping is to begin. By default, column wrapping occurs once. Setting *Repeat Wrap* to **`-1`** will cause the wrapping to occur as many times as can fit within the page width.

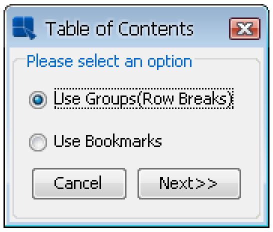
> **Note**
>
> You can not specify column wrapping to occur more times than can fit within the page width. For example, if you specify three times, and the resultant report would be wider than the page, it will only repeat the wrapping twice.

1.5.7.10. Table of Contents

EspressReport has an ability to generate a table of contents for your report. The table of contents can either show the defined groups in the report, or a list of user-defined bookmarks. The TOC will display in Report Viewer, Page Viewer, DHTML, and PDF exports.

1.5.7.10.1. Adding a Table of Contents

To add a table of contents to a report, select *Insert Table of Contents* from the *Insert* menu. Your cursor will turn into a cross. Click to place the table of contents in either the Report Header or Table Header sections. You can not add it to any other section. A dialog will open asking you if you would like to use groups or bookmarks for the table of contents.



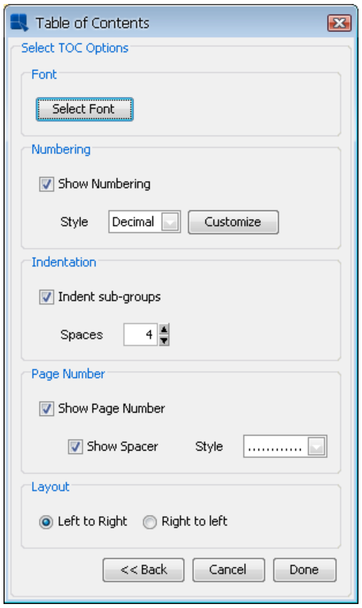
● Use Groups(Row Breaks)

○ Use Bookmarks

TOC Type Dialog

If you select to use groups, the table of contents will generate an entry for each group (and sub-group) in the report. In order to use this option, your report must contain grouped data (i.e. summary break, crosstab, master & details reports) with detail records. A crosstab report with one row break or a summary break report with column aggregation may be grouped, but because there are no detail records, you cannot create a table of contents for this presentation. If you select to use bookmarks, a one-level table of contents will be generated for each bookmark that is defined in the report. Bookmarks are defined using cell scripts. For more information about setting bookmarks, please see Section 1.9.2.1 - Formatting Actions.

The next dialog that appears allows you to specify options for the table of contents.



☑ Show Numbering

Style: Decimal

☑ Indent sub-groups

Spaces: 4

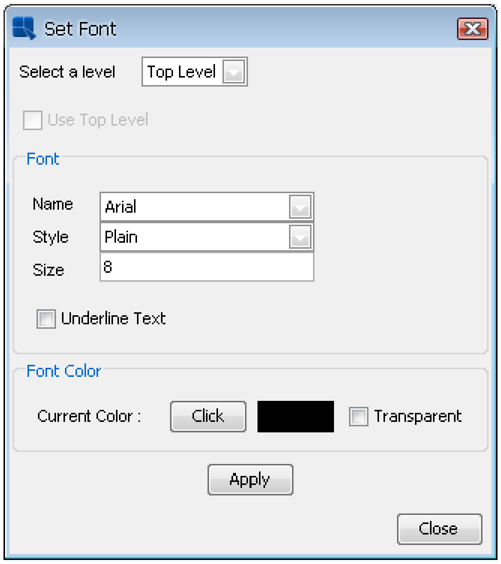
☑ Show Page Number

☑ Show Spacer Style: ....

● Left to Right ○ Right to left

TOC Options Dialog

The first option in this dialog allows you to set the font for each level of the table of contents. When you click the *Select Font* button, a new dialog will open that allows you to specify font options.



Select a level: Top Level

□ Use Top Level

Name: Arial

Style: Plain

Size: 8

□ Underline Text

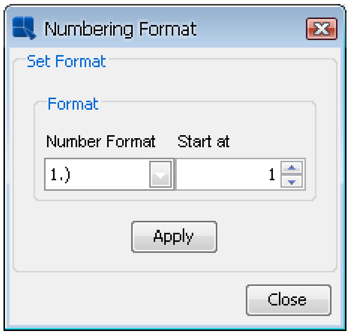
Current Color : ■ □ Transparent

TOC Font Dialog

For each level you can set the font face, style, size, and color. Note that a level corresponds with a level of nested grouping in the report. If you select to create a report with bookmarks, only the top level will be available. Sub-levels can inherit the font from the parent level. After you make changes to the font for a particular level, click the *Apply* button to save the changes.

The next option allows you to select whether to show numbering for the table of contents and which style to use. You can customize each style by clicking the *Customize* button. The following styles are supported:

Decimal: This will use a numbered configuration for the table of contents entries with whole numbers for the outermost group and decimals for each sub-group. If you click the *Customize* button for this option, the following dialog will appear:



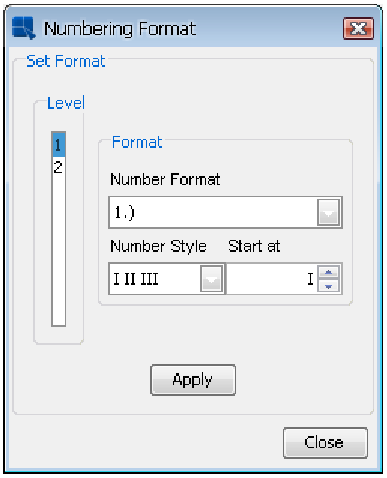
Number Format: 1.)

Start at: 1

Customization Options for TOC Decimal Format

The first drop-down list allows you to select how the number should be formatted. The second option allows you to select the starting number. After you make your choices, click the *Apply* button to apply the changes.

Outline: This option will use an outline format allowing you to select numbers, letters, or Roman numerals for each level of the table of contents. If you click the *Customize* button for this option, the following dialog will appear:



Number Format: 1.)

Number Style: I II III

Start at: I

Customization options for TOC Outline Format

The list on the left side allows you to select the level for which you would like to apply formatting. The first drop-down allows you to select the number format (this will apply whether you select numbers letters or Roman numerals for the level). The second drop-down allows you to pick the format for that level - either numbers, capital letters, lower-case letters, capital Roman numerals, or lower-case Roman numerals. The last option allows you to select the starting point for the selected style. After you make your choices, click the *Apply* button to apply the changes.

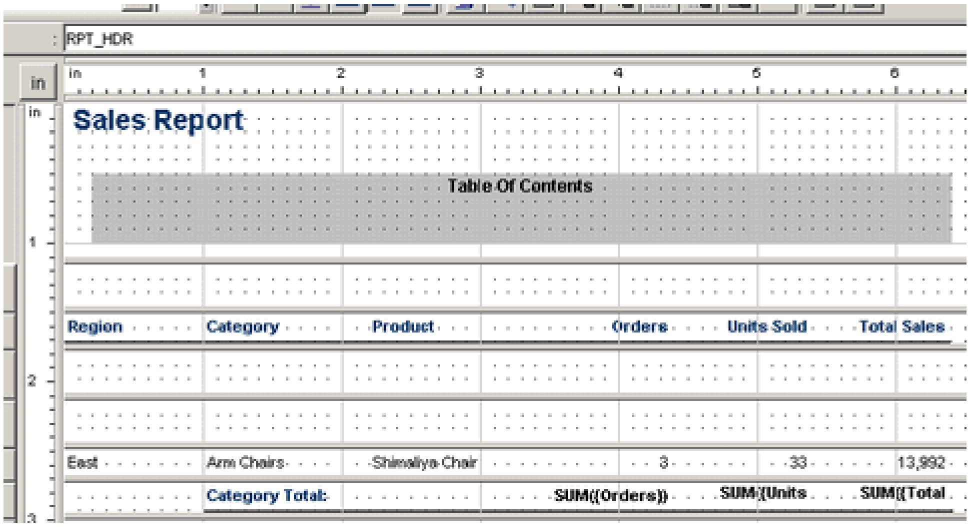
Bulleted: This option will use bullets for each entry. There are now additional customization options for this format.

The next option allows you to specify indentation for sub-groups. You can turn on/off the indentation and specify the number of spaces to use.

The next option allows you to specify whether to show the page number in the table of contents and whether to draw a spacer to the page number. You can draw a dotted line or a solid line for the spacer.

The last option in the dialog allows you to set the layout for the table of contents, either right to left, or left to right. After you finish setting options for the table of contents, click the *Done* button and it will be added to the report.

In the report, the table of contents will appear as a small grey rectangle. By default, the resize to fit content option will be turned on, but the width of the table of contents will be based on whatever size you allot for it in the section. Generally, you want to make this at least as wide as your report.



RPT_HDR

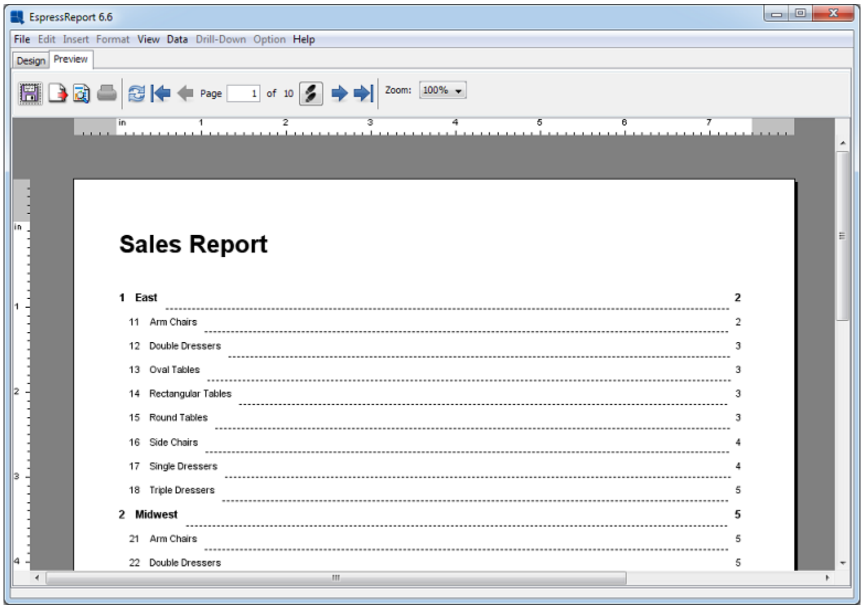
Sales Report

Table Of Contents

Region	Category	Product	Orders	Units Sold	Total Sales
East	Arm Chairs	Shimaliya Chair	3	33	13,992
	Category Total:		SUM({Orders})	SUM({Units	SUM({Total

TOC in Design Window

You can see the complete table of contents when you preview the report. The table of contents is only supported for the DHTML and PDF exports. It will not appear if you export the report to other formats.



Sales Report

1 East 2

11 Arm Chairs 2

12 Double Dressers 3

13 Oval Tables 3

14 Rectangular Tables 3

15 Round Tables 3

16 Side Chairs 4

17 Single Dressers 4

18 Triple Dressers 5

2 Midwest 5

21 Arm Chairs 5

22 Double Dressers 5

TOC in Preview Window

1.5.7.11. Moving and Resizing Report Elements

There are several ways to move and resize report elements. You can use rulers, mouse, or you can manually enter the element bounds.

Rulers: You can resize and move elements using the rulers located in upper left corner of the Design window. When you select a report element by clicking on it with your mouse, a shaded area will appear on each ruler, marking the element's horizontal and vertical bounds.

On the ruler at the top of the Design window, clicking and dragging the left-hand marker will move the element horizontally. Clicking and dragging the right-hand marker will stretch the element horizontally.

On the ruler on the left side of the Design window, clicking and dragging the top marker will move the element vertically. Clicking and dragging the bottom marker will stretch the element vertically.

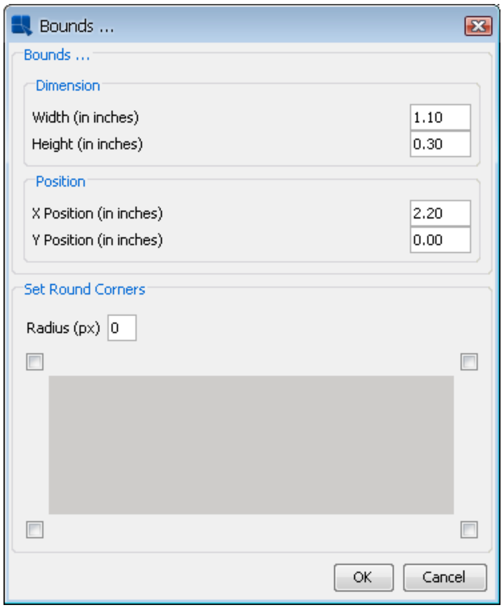
Mouse:

You can resize and move elements using your mouse. To move an element, simply click and drag it. To resize an element, click and drag on any of the sizing handles that appear around the edge of an object when it is selected. You can also right click and drag within the cell to resize it.

Manually Set Bounds:

You can manually set the bounds for any report element by clicking the *Bounds* icon on the toolbar, or by selecting *Bounds* from the *Format* menu (You can also set bounds via the *Edit Attributes* dialog that you can open from the *Format* menu or by right-clicking an object and selecting it from the pop-up menu). A dialog box will appear prompting you to enter the new element bounds. The measurements will be in inches or centimeters, depending on which unit is selected with the toggle button in the upper left corner of the designer window where the rulers meet. The X and Y coordinates are for each specific report section. The point that marks an element's X and Y position is in the upper left corner of the element.

From the same dialog, you can also set round corners for the selected element. To learn more about round corners, please see Section 1.5.7.6 - Font, Color, and Border Options.



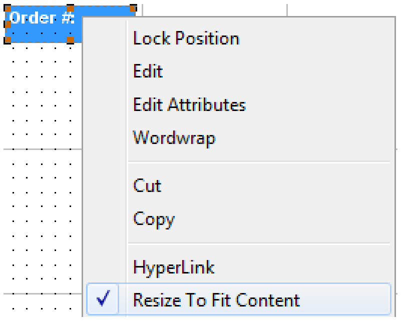
Width (in inches) 1.10
Height (in inches) 0.30
X Position (in inches) 2.20
Y Position (in inches) 0.00
Radius (px) 0
☐ ☐ ☐ ☐

Set Bounds Dialog

Resizing To Fit Contents:

Often, you may have a column field that contains data of varying length. In this case the data may be truncated, or there may be a great deal of blank space within the cell when the field is short. To alleviate this problem, you can set the element to resize its height dynamically to encompass its contents.

To do this, first select the cell you want to resize and then select Format → Resize To Fit Content. The element will dynamically resize to fit its content. Note that only the height of the cell will resize. The width will remain the same.



Dynamic Resize Dialog

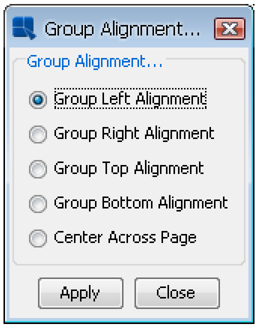
You can also invoke the section option called *Resize Cells Proportionally* to cause all of the other cells in the section to adjust their height with the resized cell. For more information about section options, please see Section 1.5.3 - Section Options.

Group Move/Resize:

You can move or resize a group of elements by selecting the elements first. Multiple elements in a report can be selected by drawing a selection box around them or by selecting them using **CTRL+Click**. You can draw a selection box by clicking and dragging on empty space in a report section. Once the elements are selected, click and drag on any element in the group to move that group. Click and drag on one of the re-sizing handles to adjust the size of elements in the group.

Group Alignment:

You can align a group of elements to the left side, right side, top, or bottom of a group. Select a group of elements and then right click on it. The *Group Edit* dialog box will appear giving you the option to align the group. Select *Group Left Alignment*, *Group Right Alignment*, *Group Top Alignment*, or *Group Bottom Alignment* and click the *Apply* button. All of the elements in the group will move to the left, top, or bottom edge of the group. You can also select the *Center Across Page* option. This will take the selected group of cells and center them in the page.



Group Alignment...

● Group Left Alignment
○ Group Right Alignment
○ Group Top Alignment
○ Group Bottom Alignment
○ Center Across Page

Group Options/Alignment Pop-up Dialog

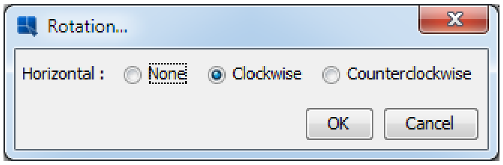
Moving Elements Between Sections:

You can move a single element or group of elements between report sections by dragging them over the section boundaries. Note that elements cannot be dragged into the Table Data section.

Column Swap:

You can swap the position of two elements within a report section. This is primarily used to rearrange the position of column fields in the report. To swap two elements, select the elements using **CTRL+Click** or the selection box and then select *Swap Columns* from the Format menu. The positions of the two elements will be switched and the surrounding elements will be moved to accommodate the swap.

Rotate: Text elements like labels, formulas, column headers, and column fields can be rotated 90 degrees clockwise or counter-clockwise. There are two ways to rotate elements. The first way to rotate an element is to right click on it and then move the mouse pointer over the *Rotate* option. A sub-menu will pop-up containing three options - *None*, *Clockwise* and *CounterClockwise*. The second way to rotate an element is to select it and then choose *Rotation* option from *Format* menu.



Horizontal : ○ None ● Clockwise ○ Counterclockwise

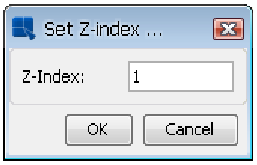
Rotation dialog

Note

Text rotation is available for DHTML, PDF, XSLX, and XLS export formats only.

1.5.7.11.1. Controlling Element Overlapping

In EspressReport, each element in a report has an associated Z index. This Z index number determines how elements behave when placed in the same space (on top of each other). When elements are placed in the same space, the element with the highest Z index will appear on top and all subsequent elements with lower Z indexes will be drawn successively below each other. To set the Z index for a report element, right click on it and select *Set Z-Index* from the pop-up menu. This will bring up a dialog allowing you to specify a number for the index of that element.



Z-Index: 1

Z Index Dialog

Note

Certain export formats that require a tabular layout, like HTML and Excel, may not appear correctly if the report has elements that overlap each other.

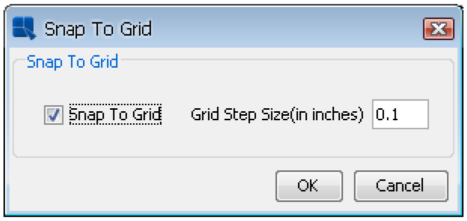
1.5.7.11.2. Locking Element Position

In Report Designer, you can lock a cell position by right clicking on the element and selecting *Lock Position* from the pop-up menu. When this feature is enabled, the report element cannot be moved with mouse, rulers, or any of the group formatting or alignment features. It can only be moved when the lock position option is turned off.

1.5.7.12. Snap to Grid

By default, Report Designer operates in *Snap to Grid* mode. This forces report elements to move only in set increments. The grid layout is represented by small dots drawn in the Design window. By default, the grid step size is 0.1 in or 0.25 cm (depending on which measurement is selected).

To modify snap to grid options, select *Snap To Grid* from the Option menu. A dialog will open allowing you to set grid properties.



☑ Snap To Grid Grid Step Size(in inches) 0.1

Snap To Grid Options

In this dialog, you can change the grid step size or disable the snap to grid feature. If you disable it, the report elements can be moved in free-form around the Design window.

1.5.7.13. Shift Mode

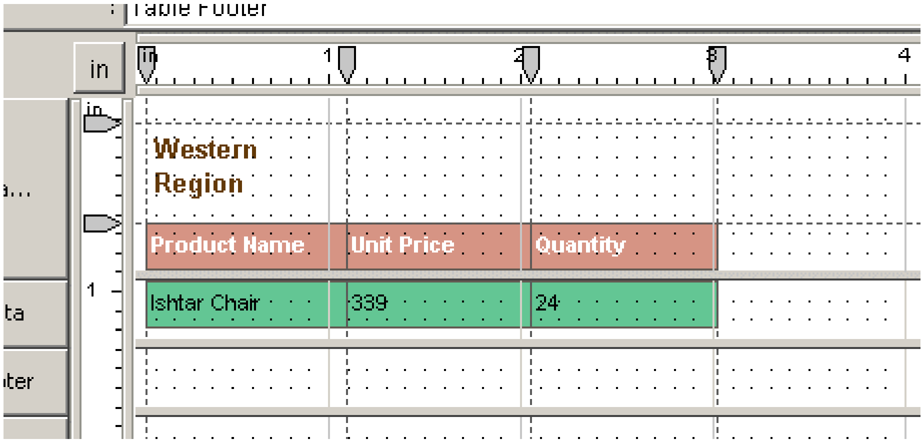
Shift mode is toggled on and off by selecting *Shift On/Off* from the *Option* menu. When shift mode is enabled, report elements will move to accommodate changes in other report elements. Hence, if you resize the width of one report element from 1" to 2", the elements next to the current element will shift to the right. If shift mode is not selected, other report elements will not move, even if the resized element ends up overlapping another.

Note

Shift mode works best if you only resize elements or move them by a small amount. If you move elements a pronounced distance with shift mode enabled, it will displace other report elements by a pronounced distance as well, giving the report an odd appearance.

1.5.7.14. Guidelines

Another way to position and move groups of elements is to use guidelines. Guidelines allow you to place arbitrary position lines within the Design window and snap report elements to those lines. Guidelines allow you to precisely line up and position report elements without worrying about performing precise movements.



Elements Positioned with Guidelines

1.5.7.14.1. Inserting Guidelines

You can insert both horizontal and vertical guidelines by selecting *Insert Guideline* from the *Insert* menu. This option will open a sub-menu, prompting you to select whether you want a vertical or horizontal guideline. After you select your desired option, click in the design panel where you would like the guideline to be drawn. The guideline will appear as a dotted line with a marker in the upper or left ruler. You can move the guideline by clicking and dragging the marker in the ruler.

You can remove any guideline from a report by right clicking on the guideline (or the guideline marker in the ruler) and selecting *Delete* from the pop-up menu.

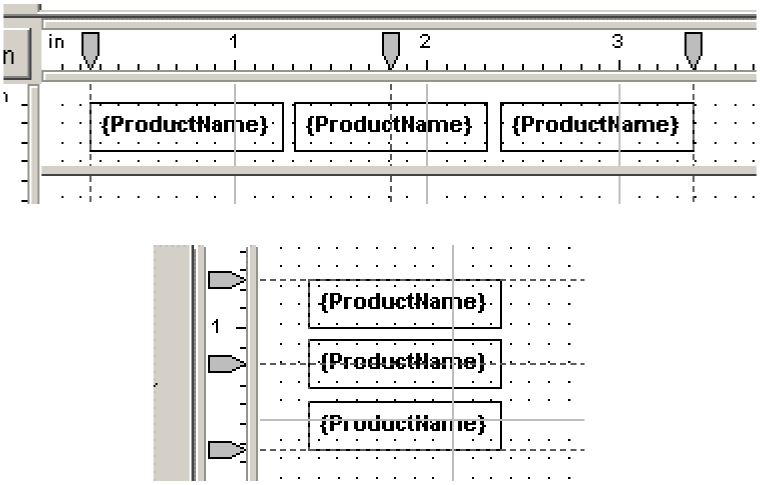
Note

Horizontal guidelines cannot be moved between report sections. If you wish to move your guideline to a new report section, you will need to insert a new guideline.

1.5.7.14.2. Positioning Elements with Guidelines

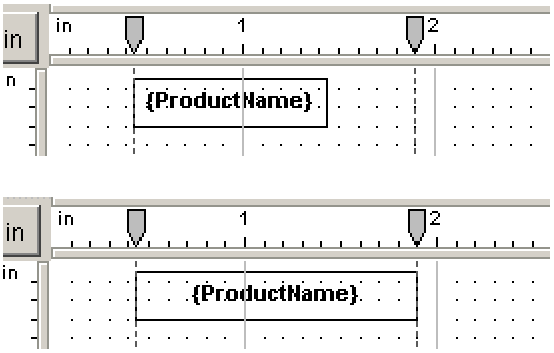
There are two ways in which objects can be snapped to guidelines. The first option is to snap all the elements that are near the guideline. To do this, right click on the guideline (or the guideline marker in the ruler) and select *Snap Elements in Range*. You can then further select to snap the cells left edge, right edge, or center. This will snap all the cells that are up to 0.1 inches away from the guideline.

The other option is to selectively snap elements to guidelines. To do this, first select the elements you want to snap to the guideline using **CTRL**+**Click** (you can select only one element, but you must use **CTRL**+**Click** to select it), then right click on the guideline (or the guideline marker in the ruler) to which you want to snap it. You can specify to snap the cell's left edge, right edge, or center to the guideline for both horizontal and vertical guidelines. Select the option you want from the pop-up menu and the cell(s) will snap to the guideline.



Guideline Positioning Options

Once an element is attached to a guideline, moving the guideline will cause the element to move as well. You can also snap an element to multiple guidelines. Snapping an element to two guidelines will cause it to stretch to fit.



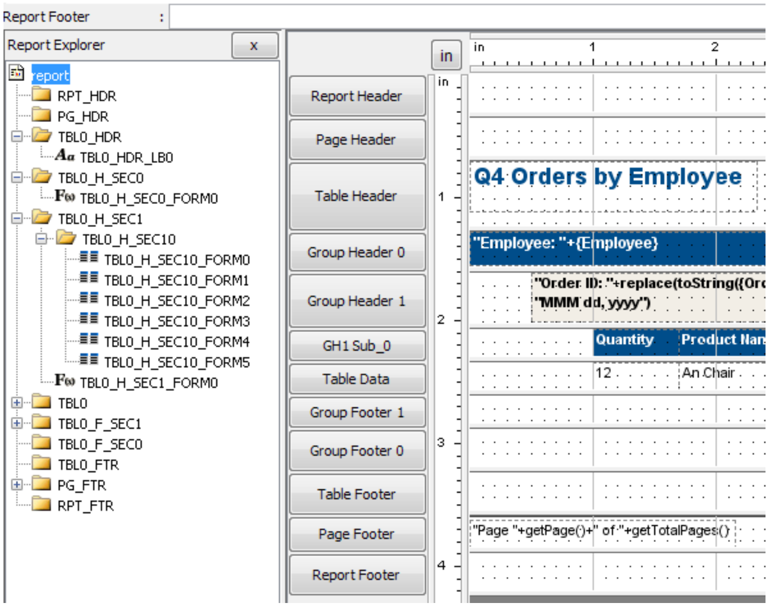
Element Snapped to Two Guidelines

When you have an element attached to two guidelines, moving either guideline will cause the element to resize (either stretch or shrink).

To release an element from a guideline, right click on it and select *Separate From Guidelines* from the pop-up menu. This will release the element from any guidelines to which it is attached. You can also release all of the elements attached to a guideline by right clicking on the guideline (or the guideline marker in the ruler) and selecting *Separate All* from the pop-up menu.

1.5.8. The Report Explorer

In addition to the main Design window, EspressReport provides another interface to select and edit the elements in a report - the Report Explorer. The Report Explorer displays all elements in the report as a tree structure on the left side of the Designer. It can be turned on and off by selecting *Report Explorer* from the *Option* menu.



Report Explorer Window

Each parent node in the tree represents a section in the report. Inside each section, an icon indicates element type and element ID. If you specified a custom ID for the element, it will display instead. For more information about custom IDs, see Section 1.8.2.1 - Using Column Field Data.

You can also edit report elements from within the Explorer. When you select an element in the tree, the element will also be selected in the Design window and the Design window will scroll in order to display that element on your screen. You can then use toolbars, menus, or pop-up menu to set any of the element properties. The pop-up menu can also be activated by right clicking on any element in the tree.

1.5.9. Global Formatting, Group Formatting, and Templates

To this point, you have only seen how to edit and format the properties of individual report elements. However, sometimes you may want to format multiple elements at once. To do this, you can use global formatting, group formatting, and templates.

1.5.9.1. Global Formatting

The global formatting feature allows you to give all the elements in your report a consistent look and feel. To set the global formats, select *Global Format* from *Option* menu. This will bring up a second menu with each of the different report element types listed: chart, column, formula, image, label, line, rectangle, column header, and title. Selecting one of these will bring up a tabbed dialog box prompting you to set the formats for those elements. Each tab contains formatting options for an element attribute. The formatting options are as follows:

Chart: border color, border thickness, set bounds.

Column: data format, font style and size, font color, background color, border color, border thickness, dual colors, set bounds, alignment, rotation, script.

Formula: data format, font color, font style and size, background color, border color, border thickness, set bounds, alignment, rotation, script.

Image: border thickness, border color, set bounds, script.

Label: font color, font style and size, background color, border color, border thickness, set bounds, alignment, rotation, script.

Line: line color, set bounds, script.

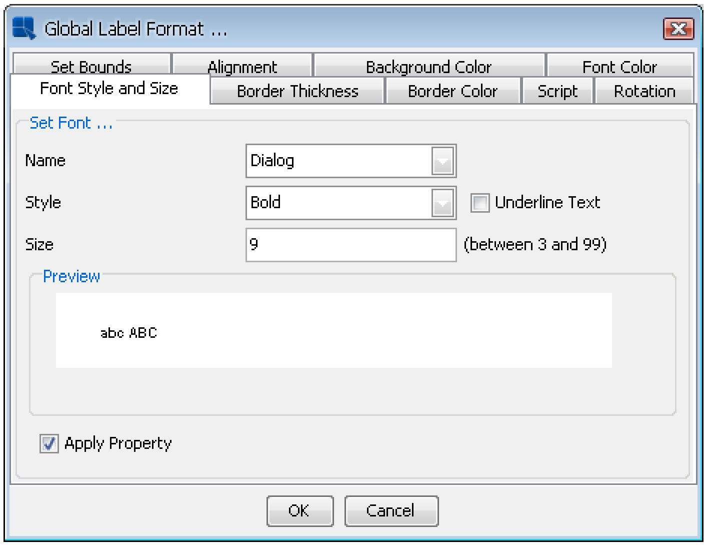
Rectangle: background color, border color, border thickness, set bounds, script.

Column Header: font color, font style and size, background color, border color, border thickness, set bounds, alignment, rotation, script.

Title: font color, font style and size, background color, border color, border thickness, set bounds, alignment.

At the bottom of each tab is a checkbox marked *Apply Property*. Checking this box will apply the property to the global formats when you click *OK*. The box automatically becomes checked when you change a property. After you make all the property changes, clicking the *OK* button will change the properties of all elements of that type in the entire report. It will also become the default attribute for any additional elements placed in the report.

> **Note**
>
> You have an option to insert images in the original size (default attribute). This can be done using the checkbox labeled *Default Size* in the *Set Bounds* tab of the image global format dialog.



Set Font ...

Name: Dialog

Style: Bold □ Underline Text

Size: 9 (between 3 and 99)

Preview: abc ABC

☑ Apply Property

Global Format Dialog

1.5.9.1.1. Global Format Import/Export

You can pass global formats from one report to another using the Import/Export feature. You can export global formats by selecting *Export* from the *Global Format* sub-menu. This will bring up a dialog box prompting you to specify a filename. The global formats will then be saved as an XML file. You can load a global format XML file by selecting the *Import* option from the *Global Format* sub-menu. This will bring up a dialog box prompting you to specify the XML file you want to import. Click the *OK* button and the formats stored in the XML file will be applied to the current report.

1.5.9.1.2. Global Formatting & Crosstab Reports

Global formatting is very important when dealing with crosstab reports. Due to the nature of the report, the number of columns can increase or decrease depending on changes in the data, data source, or filtering criteria. When new columns are generated in a crosstab report, they will inherit the default formatting. In order to control the appearance of new columns in a crosstab report, you will need to set the default formatting using global formats.

Global formatting is also important because it controls the intervals in which new columns appear in the report. New columns will appear in intervals defined by the column width in the global formats. Therefore, if you set the default width to be inch, new columns in the crosstab report will appear at one inch intervals. Because of this, it is recommended that you maintain consistent widths for the data columns in a crosstab report. Setting different widths can result in unexpected behavior when new columns are added to the report.

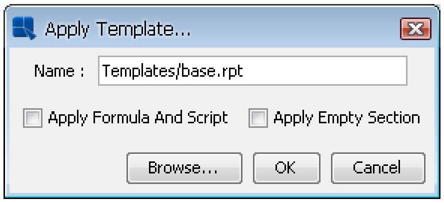
1.5.9.2. Group Formatting

To apply formatting to a group of elements, first select a group of elements either by drawing a selection box around them or using **CTRL+Click**. You can also select a row or column of cells by right clicking on a cell and selecting the *Select Column* or *Select Row* option from the pop-up menu.

Once you select the group, format the properties as you would for a single element. Formatting will only take effect on elements where it is applicable. For example, if you select four elements, two labels, a formula, and a chart, and then change the font size, it will have no effect on the chart.

1.5.9.3. Applying a Template

The features and elements of any report (`.rpt` or `.xml`) file can be applied to another report. This is accomplished using the apply template feature. To apply a template to the current report, select *Apply Template* from the *File* menu or click the *Apply Template Button* on the toolbar. A dialog box will appear, prompting you to select the file you want to apply. When you apply a template, all of the elements and their respective formatting will be applied to the current report. Only the data source information is not carried over (this includes functions and scripts in the table data section of the report).



Name : Templates/base.rpt

☐ Apply Formula And Script ☐ Apply Empty Section

Apply Template Dialog

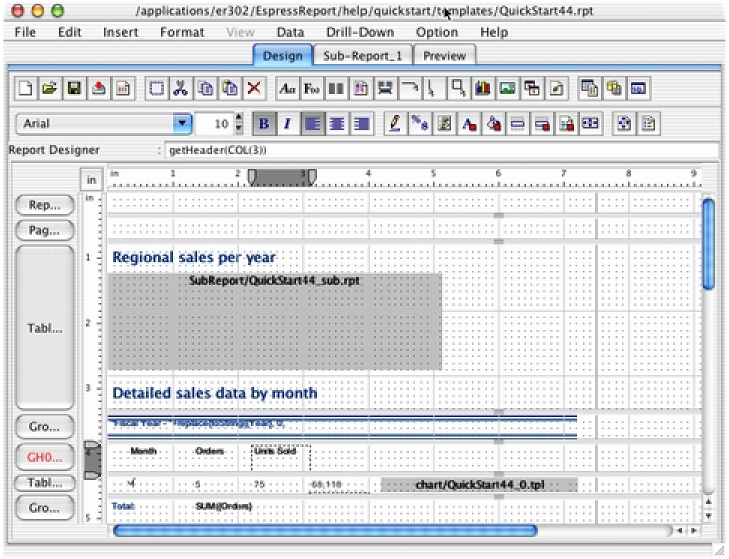
There are two additional options available in the dialog.

Apply Formula and Scripts: This option will apply the formulaic columns and cell scripts from the template onto the report. By default, they do not apply. Note that formulas and scripts from the template may not work correctly if the data types of the columns in the new report are different from those in the template.

Apply Empty Section: This option indicates whether or not to apply the blank sections from the template to the report. If you apply blank sections, they will overwrite sections in your report, essentially removing any elements you may have in the section. This option is generally used if you want to completely replicate the template that you are applying with the new report. If you do not apply blank sections, those sections in your report will not be overwritten. Only sections in the template with elements will be applied to your new report. This feature can be used if you have a certain default layout (headers/footers) that you want to pass among a group of reports. You can define a template where only the page headers/footers have defined elements and apply those headers/footers to other reports by selecting not to apply empty sections.

If you would like to have a report that has the same look and feel as another one without replacing the labels, formulas, etc, you can accomplish this by using the global format import/export feature rather than applying an entire template.

1.5.10. Report Designer in Mac OS X



Report Designer in OS X

When running Report Designer in OS X, most of the controls are the same as for Windows or Unix/Linux, with two exceptions:

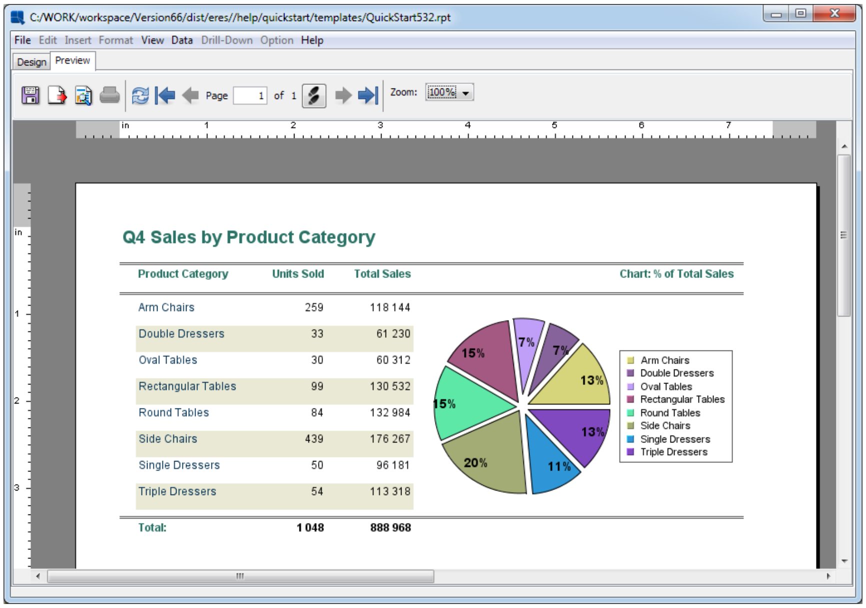
Multiple Select: Instead of **CTRL**+**Click** to select multiple report elements or list box options, use **Command**+**Click**.

Right click: To invoke pop-up menus and resize report elements, use **CTRL**+**Click** instead of right click.

EspressReport also works well with Apple's WebObjects®. For more details about developing and deploying using EspressReport in a WebObjects application, see Section 2.6 - Working with WebObjects.

1.6. The Preview Window

After inserting or manipulating report objects, you can view the results in the Preview window. You can switch back and forth between the Design and Preview windows by using the tabs in the upper left corner of the Report Designer. The Preview window gives you an accurate picture of what the report will look like if it is printed or exported.



Q4 Sales by Product Category

Product Category	Units Sold	Total Sales
Arm Chairs	259	118 144
Double Dressers	33	61 230
Oval Tables	30	60 312
Rectangular Tables	99	130 532
Round Tables	84	132 984
Side Chairs	439	176 267
Single Dressers	50	96 181
Triple Dressers	54	113 318
Total:	**1 048**	**888 968**

Preview Window

The Preview shows the page dimensions in inches or centimeters (depending on which unit is selected with the toggle button at the upper left corner of the designer window where the rulers meet). The drop-down menu on the right side of the toolbar allows you to set zooming for the preview.

1.6.1. Navigating the Report

You can navigate around the report using the toolbar or the *View* menu. The toolbar buttons perform the following functions:

Go to the first page of the report

Go to the previous page of the report

Go to a specific page (that you enter in the *Page* text box)

Go to the next page of the report

Go to the last page of the report

These navigation functions can also be performed by selecting *First Page*, *Previous Page*, *Next Page*, *Last Page*, or *Go To Page...* from the *View* menu.

1.6.2. Other Preview Window Options

Other options in the Preview window toolbar perform the following functions:

Save the file

Export the file

Load the entire report in the page viewer window. For more information about this, please see Section 1.6.2.4 - Using Page Viewer.

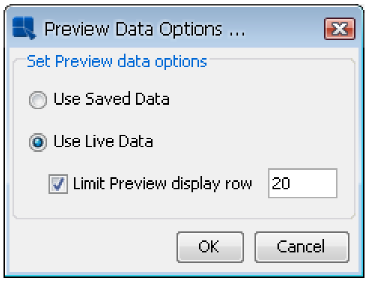
Print the report (this option is not available if the Report Designer is run as applet)

Refresh Data

In addition, the *File*, and *Data* menus remain active, allowing you to manipulate the file and data.

1.6.2.1. Preview Data Options

You will be presented with the data options dialog the first time you preview a report. This dialog allows you to set several options related to how the Preview window displays report data.



Preview Data Options ...

Set Preview data options

○ Use Saved Data

● Use Live Data

☑ Limit Preview display row 20

Preview Data Options Dialog

The first option allows you to select whether or not to use live data when previewing the report. If you select live data, the report will connect to the data source and retrieve data every time the report is previewed. Note that if the data cannot be obtained (for example, if the database connection is down), the preview will show an error. You will then need to reload the report (once the connection has been reestablished) or go back to the *Preview Data Options* and select *Use Saved Data*. If you select to use saved data, the report will show whatever data it has when you preview the report. This could be either two records of back-up data that are stored in the template, the first twenty records that are retrieved when a report is first created, or the full dataset from the previous preview (if the option had been set to use live data).

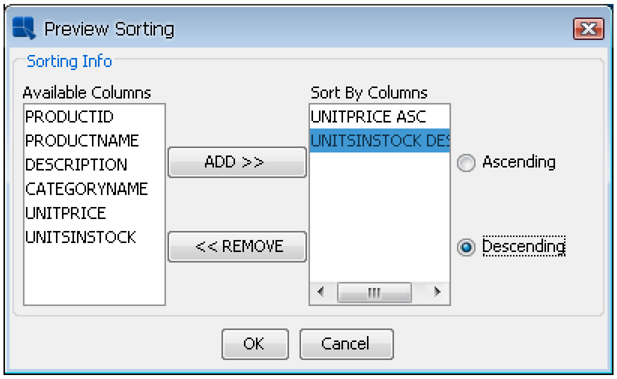
If you select to use live data to preview the report, a second option can be set that allows you to limit the number of records that should be retrieved from the data source. If your report uses a large dataset, this option allows you to see the report with data while limiting the processing time and preserving client memory.

Once you set this option the first time a report is previewed, you will not be prompted again when you preview the report. If you want to change these settings, select *Set Preview Data Options* from *Data* menu (this option is active only in *Design* tab). This will bring up the dialog again, allowing you to change the preview settings.

1.6.2.2. Sorting Data

You can sort the data within the Preview window based on any column in the report in ascending or descending order. To do this, select *‘Sort by (ascend)’* or *‘Sort by (descend)’* from the *View* menu. Each option will bring up a secondary menu containing all of the report column fields. Selecting a column field will cause the report to sort by that column in either ascending or descending order. This feature is also available in the Report Viewer applet.

To sort by multiple columns, select *Sort By...* from the *View* menu. This will bring up a dialog allowing you to select which columns to sort by and the direction.


Preview Sorting

Sorting Info

Available Columns: PRODUCTID, PRODUCTNAME, DESCRIPTION, CATEGORYNAME, UNITPRICE, UNITSINSTOCK

Sort By Columns: UNITPRICE ASC, UNITSINSTOCK DES

○ Ascending
● Descending

Preview Sorting Dialog

To sort by a column, select it in the left side dialog and click the *Add* button. You can set the sorting direction by selecting the column on the right side and clicking either the *Ascending* or *Descending* button.

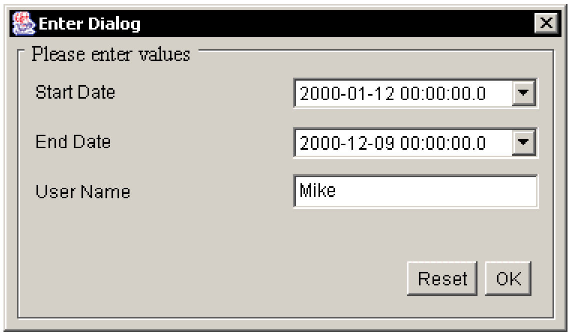
Note

Any sorting performed will not be saved with the template. When the template is re-opened (run), the data will use the original order. However, you can export the report after sorting and the exported file will reflect the re-ordered data.

It is not recommended that you use this feature with reports containing a large amount of data, as this will use a large amount of memory and can compromise performance.

1.6.2.3. Parameter Prompts

If your report uses a parameterized query or if you have defined any formula parameters in the report, you will be prompted to select parameter values when you preview the report.


Enter Dialog

Please enter values

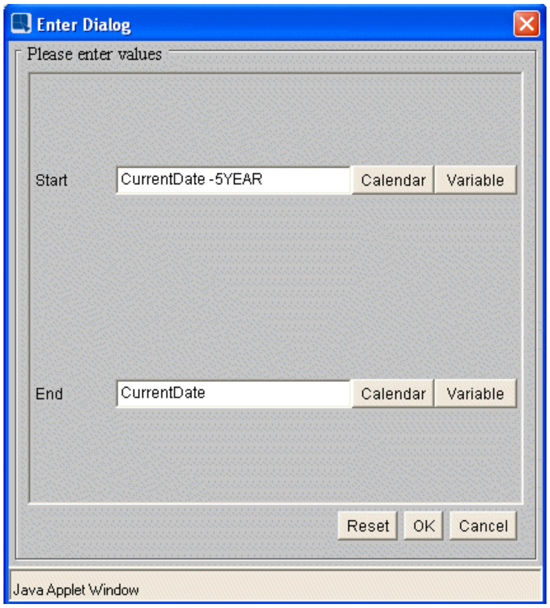
Start Date: 2000-01-12 00:00:00.0

End Date: 2000-12-09 00:00:00.0

User Name: Mike

Parameter Prompt Dialog

You are prompted to type in or select parameter values depending on how the parameters are mapped. Clicking the *Ok* button will generate the report using the specified values. You can disable parameter prompting by toggling the *Preview Parameter Prompt* option in the Data menu. If date variables are not mapped to a database column, the parameter prompt will appear as below, with the options of entering a date from the calendar or a date variable.



Start: CurrentDate -5YEAR

End: CurrentDate

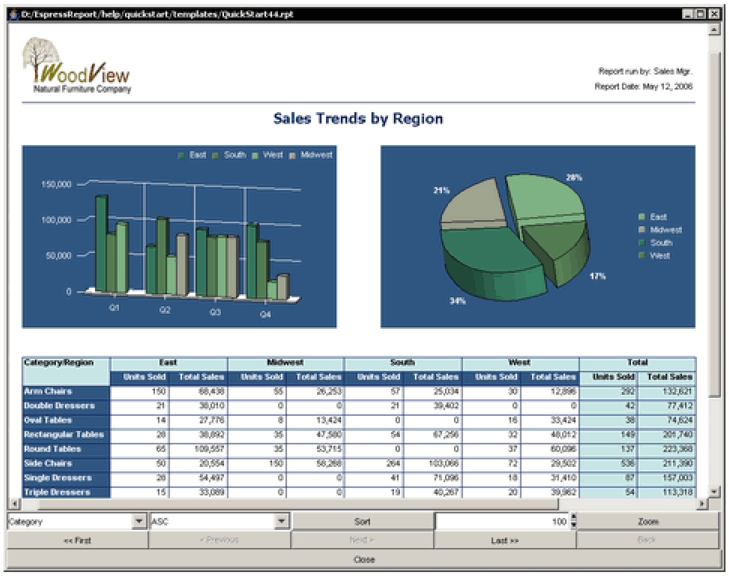
Date Parameter Prompt Dialog

For more information about query parameters, see Section 1.3.2.2.2 - Parameterized Queries. For more information about formula parameters, see Section 1.8.2.6 - Formula Parameters.

1.6.2.4. Using Page Viewer

Generally when you preview the report, the entire report with its data is kept in memory. This allows you to quickly preview and navigate through the generated report. However, for large reports, previewing the entire report could significantly affect system performance as the entire report is loaded into memory. To prevent this problem, you can limit the number of records that are fetched during preview as detailed in Section 1.6.2.1 - Preview Data Options.

On the other hand, you may want to preview the entire report without loading it into memory when creating an ad-hoc report. To do this, EspressReport allows you to load the report in the Page Viewer. To do this, select *Launch Page Viewer* from the *View* menu or click the *Launch Page Viewer* icon on the toolbar. This will open a Page Viewer window containing the report.



D:/EspressReport/help/quickstart/templates/QuickStart44.rpt

Report run by: Sales Mgr.
Report Date: May 12, 2006

Sales Trends by Region

Category/Region	East		Midwest		South		West		Total	
	Units Sold	Total Sales	Units Sold	Total Sales	Units Sold	Total Sales	Units Sold	Total Sales	Units Sold	Total Sales
Arm Chairs	150	68,438	55	26,253	57	25,034	30	12,896	292	132,621
Double Dressers	21	38,010	0	0	21	39,402	0	0	42	77,412
Oval Tables	14	27,776	8	13,424	0	0	16	33,424	38	74,624
Rectangular Tables	28	38,892	35	47,580	54	67,256	32	48,012	149	201,740
Round Tables	65	109,557	35	53,715	0	0	37	60,096	137	223,368
Side Chairs	50	20,554	150	58,268	264	103,066	72	29,502	536	211,390
Single Dressers	28	54,497	0	0	41	71,096	18	31,410	87	157,003
Triple Dressers	15	33,089	0	0	19	40,267	20	39,962	54	113,318

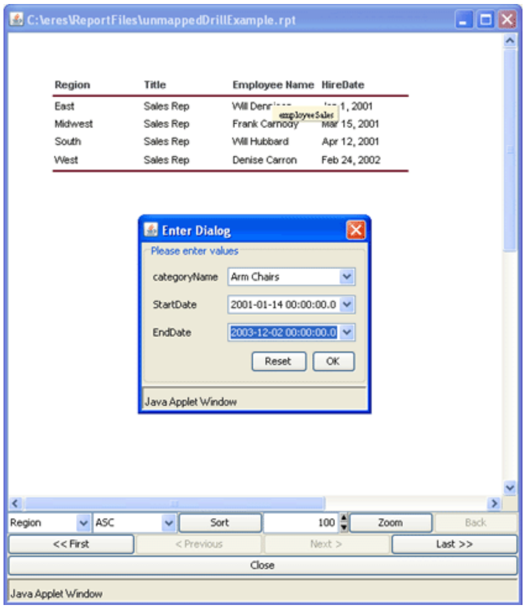
Category, ASC, 100

Report Shown in Page Viewer Window

The Page Viewer window will show you the entire report. You can navigate around the different pages by clicking the buttons on the toolbar at the bottom of the window or by right clicking and selecting to change pages from the pop-up menu. Because the Page Viewer is a static format, the window will not automatically reflect any subsequent changes made to the report. You will need to close the window and re-launch the Page Viewer. For more information about the Page Viewer, please see Section 2.2 - Introduction to Page Viewer.

1.6.2.4.1. Unmapped Drill-down Parameter

When viewing a report with unmapped drill-down parameters, clicking on the links in the main report will cause a parameter selection dialog to pop up, allowing you to fill in the unmapped parameters.



C:\eres\ReportFiles\unmappedDrillExample.rpt

Region	Title	Employee Name	HireDate
East	Sales Rep	Will Dennison	Jan 1, 2001
Midwest	Sales Rep	Frank Carnody	Mar 15, 2001
South	Sales Rep	Will Hubbard	Apr 12, 2001
West	Sales Rep	Denise Carron	Feb 24, 2002

employeeSales

Enter Dialog

Please enter values

categoryName: Arm Chairs

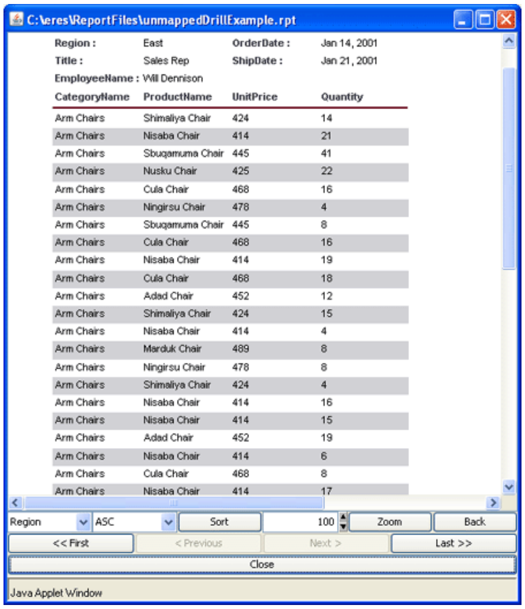
StartDate: 2001-01-14 00:00:00.0

EndDate: 2003-12-02 00:00:00.0

Java Applet Window

Parameter Selection Dialog

Once the values are entered, click the *OK* button and the drill report will be shown. To return to the previous level, click the *Back* button in the Page Viewer toolbar.



C:\eres\ReportFiles\unmappedDrillExample.rpt

Region : East OrderDate : Jan 14, 2001

Title : Sales Rep ShipDate : Jan 21, 2001

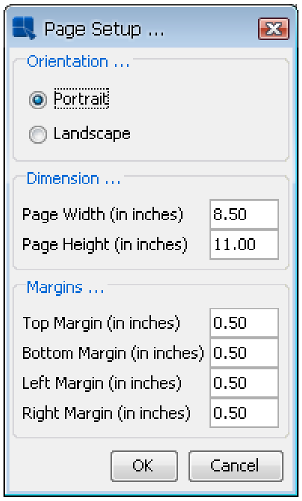
EmployeeName : Will Dennison

CategoryName	ProductName	UnitPrice	Quantity
Arm Chairs	Shimaliya Chair	424	14
Arm Chairs	Nisaba Chair	414	21
Arm Chairs	Sbuqamuma Chair	445	41
Arm Chairs	Nusku Chair	425	22
Arm Chairs	Cula Chair	468	16
Arm Chairs	Ningirsu Chair	478	4
Arm Chairs	Sbuqamuma Chair	445	8
Arm Chairs	Cula Chair	468	16
Arm Chairs	Nisaba Chair	414	19
Arm Chairs	Cula Chair	468	18
Arm Chairs	Adad Chair	452	12
Arm Chairs	Shimaliya Chair	424	15
Arm Chairs	Nisaba Chair	414	4
Arm Chairs	Marduk Chair	489	8
Arm Chairs	Ningirsu Chair	478	8
Arm Chairs	Shimaliya Chair	424	4
Arm Chairs	Nisaba Chair	414	16
Arm Chairs	Nisaba Chair	414	15
Arm Chairs	Adad Chair	452	19
Arm Chairs	Nisaba Chair	414	6
Arm Chairs	Cula Chair	468	8
Arm Chairs	Nisaba Chair	414	17

Drill-Down Level

1.6.3. Setting Page Properties

The output you see in the Preview window is based on the page dimensions and margins that are set in the Design window. The page dimensions are marked by the rulers, with the shaded areas indicating the page margins. To adjust these properties, select *Page Setup* from the *Option* menu. This will bring up a dialog box prompting you to set the page orientation (landscape or portrait), height, width, and margins. Measurements are in inches or centimeters, depending on which metric system is currently selected. Changing the orientation will also change the default print properties.



Orientation ...

● Portrait
○ Landscape

Dimension ...

Page Width (in inches) 8.50
Page Height (in inches) 11.00

Margins ...

Top Margin (in inches) 0.50
Bottom Margin (in inches) 0.50
Left Margin (in inches) 0.50
Right Margin (in inches) 0.50

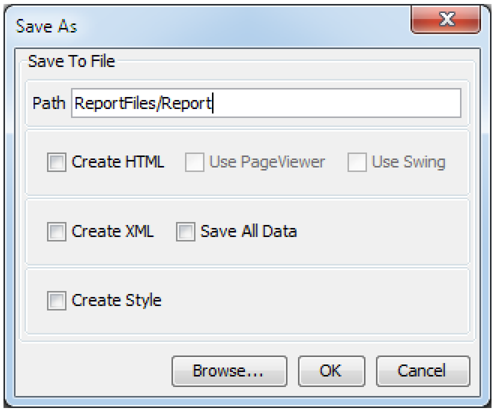
Page Properties Dialog

1.7. Saving and Exporting Reports

After you finish editing the report, you can save it as a template or export it to DHTML, PDF, CSV, Excel (XLS), Excel 2007 (XLSX), Text, XML, or Rich Text format.

1.7.1. Saving Reports

You can save the current report by selecting *Save* or *Save As* from the *File* menu, or by clicking the *Save* button on the toolbar. All reports are saved as templates either in `.pak` (binary) or XML format. Selecting the save option will simply overwrite the existing file, unless you are working on a new report that has yet to be saved. Selecting *Save As* option will bring up a dialog box prompting you to create a file name. The *Save As* dialog also has several checkboxes.



Save To File

Path ReportFiles/Report

□ Create HTML □ Use PageViewer □ Use Swing

□ Create XML □ Save All Data

□ Create Style

Save As Dialog

Create HTML: This option will create a HTML page with the Report Viewer or Page Viewer applet embedded. The file will have the same name as your `.pak` file and it will be placed in

HTML directory. By default, the Report Viewer applet is used. For more information about the Report Viewer, please see Section 2.1 - Introduction to Report Viewer.

Use Page Viewer: Checking this option indicates that you would like to generate the applet page using the Page Viewer applet instead of the Report Viewer. For more information about the Page Viewer, please see Section 2.2 - Introduction to Page Viewer.

Use Swing: This checkbox specifies whether or not to use the swing version of the Report Viewer or the Page Viewer depending on which viewer is selected for the applet page.

Create XML: This checkbox specifies whether or not to save the report in XML format. By default, reports are saved in binary (`.pak`) format. When this checkbox is checked, EspressReport will generate the template in XML format. The XML file can be saved and re-opened by the Designer. It can also be applied as a template for other reports.

Create Style: This checkbox specifies whether or not to save the report as a custom style. Reports saved in this format can be applied as a default look and feel to new reports. However, they cannot be used as a report anymore. For more information about this featue, see Section 1.4.6.1 - Custom Styles.

Save All Data: This option will save all of the report data in the template. By default, only two records are saved. This feature is useful if you want to preserve the structure of a crosstab when changing data sources or editing the report without the datasource. It also allows users to share a complete report when the data source is not available. Reports are always opened with backup data in the Report Designer. Users can get the latest data by selecting the "Live Data" option when previewing the report.

1.7.1.1. Changing the Save Location

By default, all reports are saved/opened from `/templates/` directory. You can change the default directory by modifying `reportdesigner.bat/.sh` file or the HTML source of the applet page that is used to launch Report Designer.

To change the default save location, add the following argument to `reportdesigner.bat` or .sh file: `-templatesDirectory:<Path>` where the path is a directory relative to the root directory of the installation. For example:

```
-templatesDirectory:templates/mytemplates
```

will set the default directory to `<EspressReportInstallDir>/templates/mytemplates`.

If you are running Report Designer using an applet, you will need to add the following parameter to the applet HTML page:

```
<PARAM NAME=templates_directory VALUE="templates/mytemplates">
```

1.7.1.2. XML Encoding

Both data registry files and report templates in EspressReport can be saved in XML format. By default, both types of files use the Western European encoding. In order to save data registries (including queries) and report templates correctly in XML, you will need to set the XML encoding to use a different character set. For more information about using this feature, please see Section 4.1.2.4 - XML Encoding.

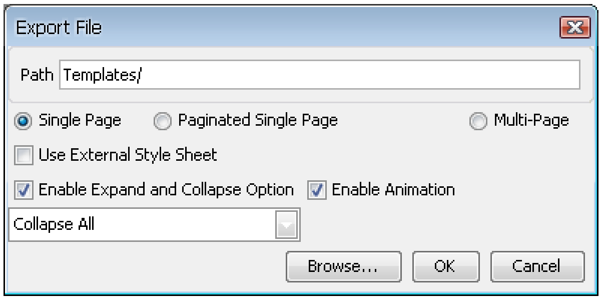
1.7.2. Exporting Reports

You can export the current report to multiple formats directly from the Report Designer. To export a file, select *Export* from the *File* menu or click the *Export* button on the toolbar. This will bring up a second menu allowing you to select the type of file you want to export to: DHTML, PDF, CSV, Excel (XLS), Excel 2007 (XLSX), text, XML, or rich text. You can also export it by clicking the *Export File* button on the toolbar.

DHTML: If you select DHTML as the desired export format, you will be prompted to enter the file name and location of the new DHTML file. Because DHTML is a text only format, charts will be saved as separate image files in the same directory as the DHTML file. If your report includes a chart, be sure to set the chart export format before exporting to DHTML, to specify the desired image format and attributes. If your report has images, the image files will be referenced in the DHTML document. DHTML will produce a much more accurate output than HTML; however, not all of the report features will translate accurately in older browsers. Internet Explorer 11, Chrome, Firefox, or Edge are recommended for viewing.

You can specify to export the report to one page, multiple pages, or as a paginated single page. If you specify one page, the entire report will be exported into one DHTML file. If you specify multiple pages, each page of the report will be generated in a new DHTML file. The output pages are linked by a navigation bar on top of each page. If you specify to export as a paginated single page, only one DHTML file will be generated. However, the page size, spacing, and margins will be retained, allowing the report to be printed out of the browser and appearing the same as if printed from a PDF or Report Designer.

You can also specify to use internal or external style sheet (CSS) definitions to apply formatting to the exported DHTML instead of generating style definitions for each report element. For more information about this option, see Section 1.7.2.1.1 - PDF Font Mapping Import/Export.



Path Templates/

● Single Page ○ Paginated Single Page ○ Multi-Page

□ Use External Style Sheet

☑ Enable Expand and Collapse Option ☑ Enable Animation

Collapse All

Export to DHTML Dialog

For reports with grouped data (except for Side-By-Side Master & Details reports), you can select to add expand and collapse controls for the grouping. This allows for a more compact presentation of the report. Each grouping can be expanded and/or collapsed depending on the viewing requirement. When selecting this option, you can also choose the initial presentation, i.e. show the report completely collapsed, completely expanded or expanded to a particular group level. You can also choose to enable animation wherein the grouping that is being expanded or collapsed fades in or out of view. Note that the expand and collapse controls can only be selected for single page exports and the controls will not appear if the Section Header or Footer is empty.

You can specify a title for the exported DHTML file. The title will appear as `<title>` meta elements in the export. To add a title, select *Html Page Title* from the *Option* menu. This will bring up a dialog allowing you to enter the title.

PDF: If you select PDF as the desired export format, you will be prompted to enter the file name and location of the new PDF file. Because PDF format supports images, there will be no separate files created by the export.

You can also set encryption for PDF files. When you check the *Encrypt PDF Export* box in the export dialog, two new options will appear, allowing you to specify a user and an owner password for the generated document. More encryption options are available when exporting files through the API.

For more information about exporting to PDF, please see Section 2.3.5.7.2 - PDF Exporting Options.

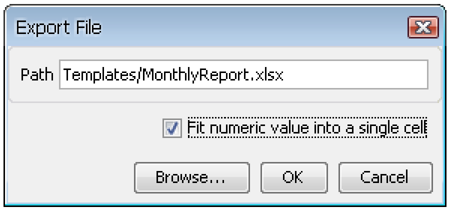
CSV: CSV export creates a text file of the data used in the current report with a comma field delimiter. If you select CSV as the desired export format, you will be prompted to enter the file name and location of the new CSV file, the *Delimiter* for the file (either Tab, Space, Double Space, Comma, or Semi-Colon), and the *Newline Delimiter* (either Windows(\r\n), Mac(\r), Others(\n), System).

> **Note**
>
> CSV export will only export the data associated with the report and none of the objects and attributes of the report itself.

Excel (XLS): If you select an Excel spreadsheet as the desired export format, you will be prompted to enter the file name and the location of the new spreadsheet. EspressReport generates a formatted Excel output where spacing, fonts, colors, lines, and grids will be applied. Excel exports also include charts and images. Note that charts will be exported as images and thus are not editable in Excel. Any gif, jpeg, and/or png image will be shown in the Excel export. You can use the CSV export to get a plain data dump to Excel.

Excel 2007 (XLSX): Excel 2007 (XLSX) export creates an XLSX file used in Microsoft Excel 2007 or newer. If you select an Excel 2007 as the desired export format, you will be prompted to enter the file name and the location of the new spreadsheet. EspressReport generates a formatted Excel 2007 output where spacing, fonts, colors, lines, and grids get applied. Excel 2007 exports also include charts and images. Again, the charts will be exported as images and thus are not editable in Excel.

If you plan to use Excel functions in the report, you will want to select the option *Fit numeric value into a single cell*. This option guarantees that each numeric data is exported to its own cell allowing you to use these values in Excel functions. Note that this option will sometimes affect the cell alignment of the exported report.



Path: Templates/MonthlyReport.xlsx

☑ Fit numeric value into a single cell

Export to Excel Dialog

For reports with grouped data, you can select to print each group on a different sheet in the exported file. This option is set in the Section menu for the Group Header section. For more information, please see Section 1.5.3 - Section Options.

> **Note**
>
> Since Excel spreadsheets are a fixed field format, not all report elements may translate correctly.

TXT: Unlike CSV, which creates a comma delimited data file, the text export feature generates the entire report as a text file. Because the report is exported as pure text, no formatting information is retained. If you select text as the desired export format, you will be prompted to enter the file name and the location of the new text file, the *Delimiter* for the file (either Tab, Space, Double Space, Comma, or Semi-Colon), and the *Newline Delimiter* (either Windows(\r\n), Mac(\r), Others(\n), System).

XML: There are two XML export versions. One is a data file and the other exports report data with formatting information.

Pure Data: This XML export option exports the data used by the report in XML format. The format of the XML file is the same as the formatting requirements for XML input data.

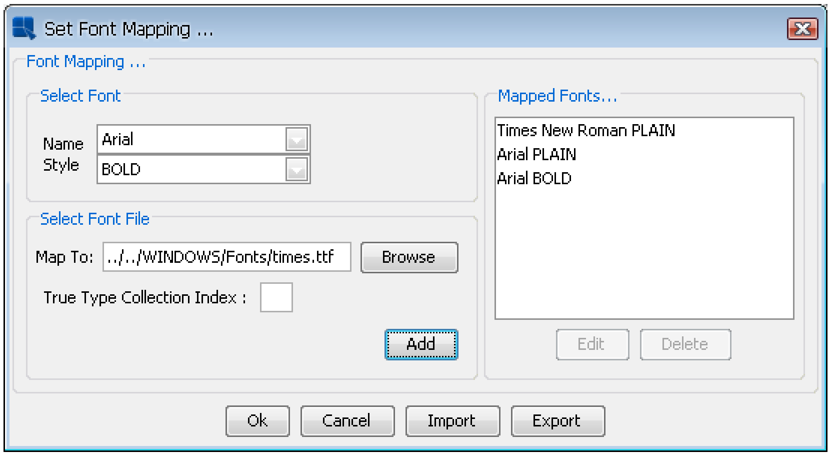
Data + Format: This XML export creates an XML version of the whole report. All of the report data as well as formatting information are included. This file is similar to the XML version of the DHTML export and can be used in conjunction with style sheets to display HTML content. This export allows you to export the entire report as one XML file or as a separate file for each page.

Rich Text: If you select rich text as the desired format, you will be prompted to enter the file name and the location for the new rich text file. EspressReport uses the full capability of the RTF 1.5 specifications to generate a rich text output very similar to the actual report. The rich text export may not work in all viewers. By default, EspressReport will export all the colors in the RTF export directly into the output. However, older versions of Microsoft Word only support 16 colors. To support these viewers, you can export the RTF using 16 colors. This option will convert the colors used in the report into the basic color palette for Microsoft Word 97.

1.7.2.1. PDF Font Mapping

EspressReport allows you to use any font on the system for reports. For most formatted exports (DHTML/Rich Text/Excel/Excel 2007) system fonts will translate automatically in the generated output (For Rich Text and Excel/Excel 2007, if the fonts cannot be found, they will default to Arial). However, for PDF you will have to manually specify a .ttf (true type font), .ttc (true type collection), .pfb, or .afm file for any system font that you would like to use in the report.

To set font mapping for the PDF export, select *Font Mapping* from the *Option* menu. This will bring up a dialog allowing you to specify font files.



Name: Arial
Style: BOLD

Map To: ../../WINDOWS/Fonts/times.ttf

Times New Roman PLAIN
Arial PLAIN
Arial BOLD

Font Mapping Dialog

You can select a specific .ttf, .ttc, .pfb, or .afm file for each font and style combination. You can either type the full path or browse to the font file. If you are using a .ttc file, you will need to specify the font index in the box provided (.ttc files contain more than one font). Once you specify the correct file, click the *Add* button to save the mapping to the list. You can edit or delete existing mappings by selecting them in the list and clicking the appropriate button.

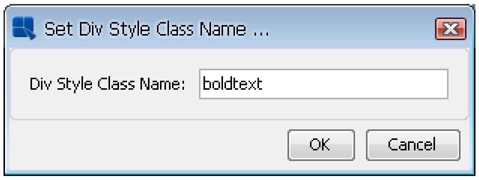
1.7.2.1.1. PDF Font Mapping Import/Export

You can pass the font mapping from one report to another using the Import/Export feature. You can export the font mapping by clicking the *Export* button on the font mapping dialog. This will bring up a dialog box prompting you to specify a file name. The font mapping will then be saved as XML file. You can load a font mapping XML file by selecting the *Import* option from the dialog. This will bring up a dialog box prompting you to specify the XML file that you want to import. Click the *OK* button and the mapping stored in the XML file will be applied to the current report.

1.7.2.2. CSS Options

When exporting a report to DHTML, the style definitions are created for each element in the report by default. This makes the export faster. However, it can result in a very large file size for large reports. To limit the size of the generated DHTML file, users can select to generate an internal list of global style definitions. This option is slightly slower, but it produces a significantly smaller file. In addition, EspressReport provides an option that allows users to apply their own style definitions to report elements by selecting a .css file when exporting the report.

If you are going to use external style sheet definitions, you will need to first select the class name from the external style definitions that should be associated with elements in the report. To set the class for a report element, right click on the cell and select *Set Style Class Name* from the pop-up menu. This will bring up a dialog allowing you to specify a style name for that report element.



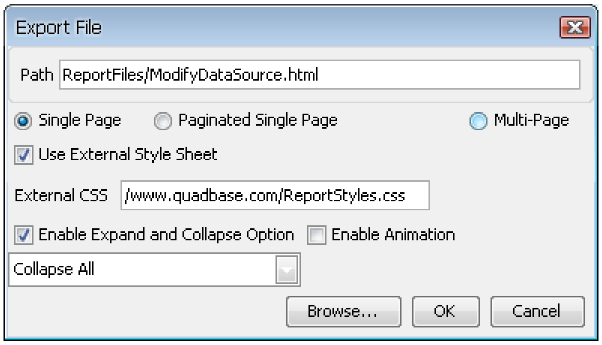
Div Style Class Name: boldtext

Set Style Class Dialog

Elements for which you do not select a style class will generate their own style based on the settings in the report when the report is exported to DHTML.

Users can also export a .css file for the formatting defined in the report. These style definitions are the same as generated by the internal style sheet option. This feature can be used to apply formatting created in the Report Designer to other reports or other web pages. To export a .css file from a report, select *Export Style Sheet* from the *Option* menu. A dialog will open, prompting you for the name and location for the generated file.

CSS options can be set in the export dialog for the DHTML export.



Path ReportFiles/ModifyDataSource.html

● Single Page ○ Paginated Single Page ○ Multi-Page

☑ Use External Style Sheet

External CSS /www.quadbase.com/ReportStyles.css

☑ Enable Expand and Collapse Option □ Enable Animation

Collapse All

DHTML Export Dialog

The *Use External Style Sheet* option allows you to specify the location (either relative path or full HTTP path) of a .css file which contains the definitions for the style classes that you have assigned to report elements.

Note

The formatting from an external style sheet file will only be applied to the DHTML export. Other export formats will use the formatting defined in the report template.

1.8. Using Formulas & the Formula Builder

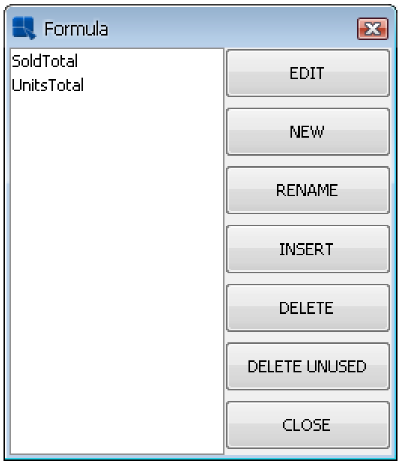
EspressReport supports a wide variety of formulas, giving you an important mechanism for manipulating and displaying report data. Using formulas, users can add summaries and aggregations, perform basic calculations, and build complex expressions using one of over 70 built-in functions.

Note

Some of the syntax for EspressReport formulas has changed greatly between v2.51 and v3.0. In most cases, the deprecated syntax will continue to work correctly. However, it is recommended that you check older templates to make sure the functions still return correctly if you have upgraded to v3.0 or higher from v2.51 or lower.

1.8.1. Creating a Formula

When you select to insert a new formula into the report (either by selecting *Insert Formula* from the *Insert* menu or clicking the *Insert Formula* button on the toolbar), a list of all the defined formulas in the report will appear.



Formula List Dialog

To insert a formula into the report, select a formula from the list and click the *Insert* button. The formula list dialog will then close and you will be allowed to place the formula.

The report section in which you place the formula is extremely important, since the formula will reset each time that section repeats in the report. For example, if you place a formula in the Group Footer of a summary break report, the formula will recalculate for each group in the report. If the formula references a column value, it will only use data within each break group. If you place the same formula in the Report Footer section, it would only calculate once and it would use all of the data in the column field, regardless of breaks and grouping.

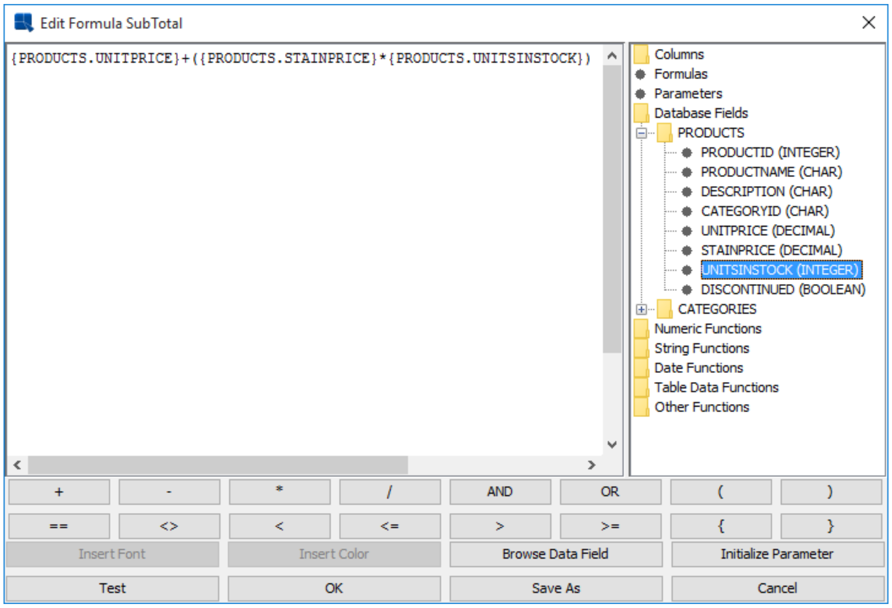
Formulas and labels that are placed in the Table Data section of the report become computed columns and can be treated as column fields as well as formulas.

Note

Labels are automatically treated as Strings and can be edited by double clicking on them (i.e. to use a number as integer in a formula, open the formula builder and remove double quotes from the number).

1.8.1.1. The Formula Builder

Formulas are constructed using the Formula Builder interface. To launch the Formula Builder, you can select to create a new formula by selecting *New* from the formula list dialog. This will prompt you to specify a name for the new formula. Once you specify a name, the Formula Builder will open, allowing you to construct a formula. You can also use the Formula Builder to edit an existing formula. To edit a formula, select it from the list and click the *Edit* button or double-click on a formula cell in the Design window.



```
{PRODUCTS.UNITPRICE}+({PRODUCTS.STAINPRICE}*{PRODUCTS.UNITSINSTOCK})
```

Formula Builder Window

The main window of the Formula Builder contains the text of the formula. The folders on the right side contain various elements that can be added to the formula, including column fields, other formulas, parameter values, database fields, and built-in functions. The first two rows of buttons contain the most common arithmetic and Boolean operators (operators and functions are discussed in the next section). The last two rows of buttons serve as command buttons and perform the following functions:

Insert Font: This option is not available for formulas, it is only available for scripting.

Insert Color: This option is not available for formulas, it is only available for scripting.

Browse Data Field: This option is for database fields. It will query the database and return the first 20 values for the column.

Initialize Parameter: This option will bring up the parameter initialization dialog for any parameters defined in the formula. Parameters are discussed in Section 1.8.2.6 - Formula Parameters.

Test: This option will check the current formula and see if it is correct.

Ok: This opton will close the formula builder and return to the formula list dialog, allowing you to insert the formula into the report.

Save As: This option allows you to save the current formula under a different name. Because you can re-use a formula in different places in the report, using the save as option allows you to modify a formula in one place, without changing it for every place where the formula is referenced.

Cancel: This option will close the Formula Builder without saving the formula/changes.

1.8.2. Formula Syntax

The following section details various operators and functions available in EspressReport and how to use them.

1.8.2.1. Using Column Field Data

You will often want to use data from column fields for your formulas. This can be easily accomplished in formulas using the following syntax: `{<Field Name>}`. The braces delimit the field and the field name is the name of the column field you're referencing. So in a basic example: `{UnitPrice} * {Quantity}` would multiply two column fields together.

In most cases, the column name is the name specified in the data source (i.e. the database column name or alias). If you are uncertain about the correct name, you can simply select the column field you want to use from the *Columns* node in the right panel in the formula builder.

There are two other functions you can use to retrieve column data:

id() This option will allow you to retrieve the value of any cell in the report, including column fields. The syntax is `id( <Object ID>)`. You can retrieve an object's ID by right clicking on the cell and selecting *Properties* from the pop-up menu. You can also retrieve the ID for column fields by selecting *View Column Mapping* from the Data menu.

You can also assign a custom ID to elements in the report, instead of using the default ID. To assign a custom ID to an element, right click on it in the Report Designer and select Custom ID from the pop-up menu. This will bring up a dialog allowing you to enter an ID for that element. The argument for the `id()` function can either be the original ID for the element or the defined custom ID.

Element IDs can also be used to get a handle to an element in the Report API. You can pass in either the original ID or the custom ID into the `getData()` method to get a handle to an element.

The `id()` function also serves another important purpose. It can be used to delay the calculation of computed columns. Normally, formulas in the Table Data section of the report will compute prior to any aggregations in the group or table footers. However, sometimes you may want to calculate the aggregations prior to the column fields. For example, you want to create a column that calculates each row's percentage of a total. In order to get the correct result, the total has to be calculated before the column. To do this, simply refer to the aggregation in the Table/Group Footer using the `id()` function. The finished function in the Table Data section would look like this: `{Quantity} / id(TBL0_FTR_FORM0)`.

col() This option is a deprecated function that is used in earlier version of EspressReport to retrieve data from column fields. Although this function is still valid, it is recommended that you use the new field notation. The syntax for this function is `col(Column Index)`. To view the column index, select *View Column Mapping* from the Data menu. This will display the assigned index number for each column field. Index values are assigned based on the order in which the columns are selected for the report. The first column has an index of 0.

1.8.2.1.1. Column Aggregation

You can aggregate any column field (including computed columns) using one of the aggregation functions provided with EspressReport. The following aggregation options are available:

average() This option returns the mean value of the specified column. The syntax is `average(<Column Field>)`. To return the average value of a column field named `UnitPrice`, you can use the following formula: `average({UnitPrice})`.

median() This option returns the median value for the specified column. The syntax is `median(<Column Field>)`. If the median of the column is two values (for an even number of rows), the aggregation will return the average of the two values.

count() This option returns a count of the values in the specified column. The syntax is `count(<Column Field>)`.

countdistinct() This option returns a count of all the distinct values in the specified column. The syntax is `countdistinct(<Column Field>)`.

max() This option returns the maximum value for the specified column. The syntax is `max(<Column Field>)`.

min() This option returns the minimum value for the specified column. The syntax is `min(<ColumnField>)`.

stddev() This option returns the standard deviation for the specified column. The syntax is `stddev(<Column Field>)`.

sum() This option returns the sum of the specified column. The syntax is `sum(<Column Field>)`.

sumsquare() This option returns the sum of squares for the specified column. The syntax is `sumsquare(<Column Field>)`.

variance() This option returns the variance for the specified column. The syntax is `variance(<Column Field>)`.

It is important to note that aggregation functions can only take column fields as an argument. You cannot use an expression or another function as an argument. Therefore, `sum({UnitPrice}*{Quantity})` is not valid. To make this calculation, you would first need to add `{UnitPrice}*{Quantity}` as a formula in the Table Data section of the report. Then you can sum the computed column created by the first formula using `sum(@formulaname)`.

1.8.2.2. Using Formulas

EspressReport allows you to easily re-use report formulas by plugging them directly into new formulas. You can refer to a formula using the following syntax: `@<FormulaName>`. The formula name is the name that was assigned to the formula the first time it was created. So for example if you created a formula `{UnitPrice}*{Quantity}` called `Total`, you could retrieve the result of this formula by typing `@Total`.

If you are uncertain about the correct name, you can simply select the formula you want to use from the *Formulas* node in the right panel of the Formula Builder.

You can also use this notation to aggregate computed columns if the formula has been placed in the Data Table section of the report.

1.8.2.3. Using Parameter Values

In addition to column field and formula values, you can also access parameter values as part of a formula. Parameter values are the user supplied values to either query or formula parameters. For more information about query parameters, see Section 1.3.2.2.2 - Parameterized Queries. For more information about formula parameters, see Section 1.8.2.6 - Formula Parameters.

To refer a parameter value, use the following syntax `:<ParameterName>` for query parameters and `?<ParameterName>` for formula parameters. For example, if the report has a query parameter named `StartDate` that prompts the user to supply a date, `:StartDate` will return the date that the user has supplied when the report is run.

If you are uncertain about the correct name, you can simply select the parameter you want to use from the *Parameters* node in the right panel of the Formula Builder. Also, if you type a formula parameter name wrong, the Formula Builder will assume that you are trying to define a new parameter.

1.8.2.4. Using Database Fields

If your report uses a database as the data source, you can also use database fields that were not selected for the report in a formula. Database fields are referenced in a similar manner to column fields and use the syntax `{<Table Name>.<Field Name>}`. If your database requires three part names, the fields should be referenced accordingly.

You can select database fields to add from the *Database Fields* node in the right panel of the Formula Builder. In addition, the Formula Builder allows you to preview database fields. To do this, first select a field in the right panel and then click the *Browse Data Field* button. This will bring up a new dialog showing the first few rows from the selected column.

1.8.2.5. Constants

When using constants in formulas, there are certain formats that are required for each data type.

Numeric Data: To represent a numeric constant, you simply have to type the number. Both integers and decimals can be accepted; however, you cannot type in thousands separators. So to specify the number 12.28 as a constant in a formula, type `12.28`.

String Data: To represent a string constant, type the string delimited with double quotes. For example, to specify a string Hello, you would type `"Hello"`.

Boolean Data: To represent a Boolean constant, type either true or false without any delimiters. To specify a false Boolean value, you would type `false`. Note that these are not case sensitive.

Date Data: To represent a date constant, type the date in the following format without delimiters: MM/dd/yyyy. To specify the date August 5th, 2002, you would type `08/05/2002`.

Time Data: To represent a time constant, type the time in the following format without delimiters: hh:mm:ss. To specify the time 2:30 PM, you would type `14:30:00`. Note that time must be in 24-hour format.

Timestamp Data: To represent a timestamp constant, type the date and time in the following format without delimiters: MM/dd/yyyy hh:mm:ss. To specify the time stamp August 5th, 2002 at 2:30 PM, you would type `08/05/2002 14:30:00`.

1.8.2.6. Formula Parameters

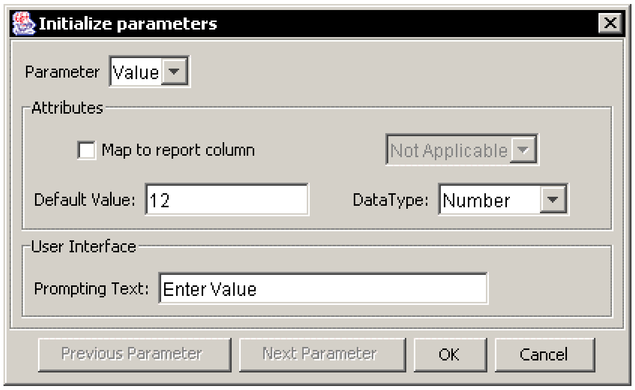
Rather than adding constants into a formula, you can specify a parameter. Using a parameter, you can collect constant values from the user (or elsewhere) at run-time and dynamically compute the formula based on those values. To specify a parameter in the formula, use a question mark “?” followed by the parameter name.

For example, the following formula `{Quantity}+?Value` would add a user supplied value to the `Quantity` field in a report.

> **Note**
>
> If you want to define a new parameter within a formula, you have to use a unique name. If you specify a parameter name of a parameter that is already defined in another formula within the report, the new formula will return the value from the previous parameter.

1.8.2.6.1. Initializing Formula Parameters

Like query parameters (detailed in Section 1.3.2.2.2.2 - Initializing Query Parameters), you must initialize a formula parameter before a formula can be used in a report. To initialize any parameters defined in the current formula, click the *Initialize Parameter* button in the Formula Builder. This will bring up a dialog prompting you to specify options for the parameter.



Initialize parameters

Parameter: Value

Attributes

☐ Map to report column — Not Applicable

Default Value: 12 DataType: Number

User Interface

Prompting Text: Enter Value

Initialize Formula Parameter Dialog

From this dialog, you can specify a default value for the parameter, map the parameter to a report column (this will give the user a drop-down list of distinct values, rather than having them type in the parameter), specify the data type for the supplied parameter, as well as specify the prompt for the end user. Clicking the *Previous Parameter* and *Next Parameter* buttons allows you to initialize each of the parameters that have been defined in the formula. Click *OK* when you finish setting up the options for all parameters.

> **Note**
>
> For formula parameters, you must specify a default value for each parameter. Also, if the parameter is mapped to a report column, the first time the report is previewed, the dialog will only contain the first twenty values of that column. This is because the full dataset is not retrieved when the report is first created. At run-time, the dialog will contain all the available values in the report column.

1.8.2.7. Operators

EspressReport provides several arithmetic and Boolean operators that allow you to create expressions in formulas. Operators use in-fix notation that places the operator between arguments in the expression, i.e. `<argument1> <operator> <argument2>`.

You can automatically insert operators into a formula by clicking one of the operator buttons in the formula builder.

1.8.2.7.1. Arithmetic Operators

Four basic arithmetic operators are supported: `+`, `-`, `*`, and, `/`. These will add, subtract, multiply, and divide two objects respectively. For example `1 + 2` will return `3`.

Generally, arithmetic operators will take numbers as arguments and will return numbers with two exceptions. `+` can be used to concatenate strings, for example `Hello + "World"` would return `Hello World`. Also, `-` can be used to subtract two date objects and return the difference in days, for example `8/2/2002 - 4/7/2002` would return `117`.

1.8.2.7.2. Boolean Operators

Eight Boolean operators are supported: `AND`, `OR`, `==`, `<`, `<=`, `>`, `>=`, and `<>`. `AND` and `OR` take Boolean arguments and return Boolean values. For example `true AND true` would return `true` or `1 < 2 OR 4 < 3` would return `true`.

`==`, `<`, `<=`, `>`, `>=`, and `<>` are comparison operators that signify equal to, less than, less than equal to, greater than, greater than equal to, and not equal respectively. Comparison operators compare two objects of the same data type and will return a Boolean value, for example `3 < 4` would return `true` and `Yes == "No"` would return `false`.

Boolean operators are more commonly used in cell scripting rather than formulas. Cell scripting is covered in Section 1.9 - Scripting.

1.8.2.8. Functions

In addition to the basic operators, EspressReport provides a number of built-in functions. Function syntax generally takes the form of a function name outside of a set of parenthesis enclosing the comma-separated arguments `function(<argument>, <argument>)`.

You can insert functions into the formula by selecting the function you want under the *Numeric Functions*, *String Functions*, or *Date Functions* nodes in the Formula Builder. Functions are inserted with hints indicating the type and number of arguments it should take. In order for the function to return properly, these hints have to be removed and replaced with valid arguments.

1.8.2.8.1. Numeric Functions

The following functions take numeric arguments.

abs()

This option will return the absolute value of a number. The syntax is `abs(<Number>)`. For example `abs(-12)` would return `12`.

acos()

This option will return the arc cosine of an angle, in the range of 0 through pi. Syntax for this is `acos(<radians>)`. For example, `acos(-sqrt(2)/2)` would return `3*pi/4`.

asin()

This option will return the arc sine of an angle in the range of -pi/2 through pi/2. The syntax is `asin(<radians>)`. For example, `asin(sqrt(2)/2)` would return `pi/4`.

atan()

This option will return the arc tangent of an angle in the range of -pi/2 through pi/2. The syntax is `atan(<radians>)`. For example, `atan(1)` would return `pi/4`.

atan2()

This option will convert rectangular coordinates (b, a) into polar coordinates (r, theta). This function returns theta by taking the arc tangent of b/a in the range of -pi to pi. The syntax is `atan2(<y_coordinate>, <x_coordinate>)`. For example `atan2(1,0)` would return `pi/2`.

ceil()

This option will return the lowest integer value that is greater than the specified number. The syntax is `ceil(<Number>)`. For example, `ceil(15.3)` would return `16`.

cos()

This option will return the cosine of an angle in radians. The syntax is `cos(<radians>)`. For example `cos(3*pi()/4)` would return `-sqrt(2)/2`.

e()

This option will return the value e, the natural logarithm base. The syntax is `e()` without any arguments.

exp()

This option will return e raised to the nth power. The syntax is `exp(<Number>)`. For example `exp(3)` would return `e^3` or `20.086`.

factorial()

This option will return the factorial of the specified object. The syntax is `factorial(<Number>)`. For example `factorial(3)` would return `6`.

floor()

This option will return the highest integer value that is less than the specified argument. The syntax is `floor(<Number>)`. For example `floor(5.52)` would return `5`.

getColumnCount()

This option will return an integer indicating the total number of columns in the report. The syntax is `getColumnCount()` without any arguments. This function returns the total number of columns whether they are visible or not.

getRowIndex()

This option will return an integer indicating the index value of the current row (of Table Data) of the report. The syntax is `getRowIndex()` without any arguments.

getTotalRowIndex()

This option will return a count of all the rows (in Table Data) in the report. The syntax is `getTotalRowIndex()` without any arguments.

getRowIndexOfCurrentTable()

This option will return the current row in the current group (table) in a report. This function is generally only relevant in reports that have grouped data like the summary break and master & details layout. For example, if the current row is the third row in a group of data, `getRowIndexOfCurrentTable()` will always return 2 regardless of what the row index is for the total report (which you could retrieve using the `getRowIndex()` function). The syntax is `getRowIndexOfCurrentTable()` without any arguments.

getTotalRowIndexOfCurrentTable()

This option will return a count of all the rows in the current group (table) in a report. This function is generally only relevant in reports that have grouped data like the summary break and master & details layout. The syntax is `getTotalRowIndexOfCurrentTable()` without any arguments.

getSiblingCount()

This option will return a count of the number of sibling groups within a nesting level. This function is generally only relevant in reports that have grouped data like the summary break and master & details layout. For example, you have a summary break report with one level of grouping. There are three distinct values in your row break column creating three groups at the level. Within the group (i.e. Group Header, Table Data, or Group Footer sections) `getSiblingCount()` would return 3. The syntax is `getSiblingCount()` without any arguments.

getChildCount()

This option will return a count of the number of child groups inside a nesting level. This function is generally only relevant in reports that have grouped data like the summary break and master & details layout. The function will return the number of child group in outer most group when running in a table header or a table footer; It will return the number of child in the related group when running in a group footer/header. For example, you have the same example as described for the previous function. Within the group, the `getChildCount()` function would return 0 as there is only one level of grouping. However, outside the group (i.e. Table Header or Table Footer sections) `getChildCount()` would return 3. The syntax is `getChildCount()` without any arguments.

getGroupIndex() This option will return an integer indication the index value of the current group. This function is only relevant in reports that have grouped data like the summary break and master & details layout. This function will only return the index of the inner most group for reports with nested groups.

log() This option will return the natural logarithm of the specified number. The syntax is `log(<Number>)`. For example `log(10)` would return `2.303`.

mod() This option will return the remainder after dividing two numbers. The first argument is the numerator and the second is the denominator. The syntax is `mod(<Number>, <Number>)`. For example `mod(12, 7)` would return `5`.

pi() This option will return the value pi. The syntax is `pi()` without any arguments.

pow() This option will return the value of the first argument raised to the power of a second argument. The syntax is `pow(<Number>, <Number>)`. For example `pow(2, 3)` would return `8`.

random() This option will return a random value greater than or equal to 0 and less than 1. The syntax is `random()` without any arguments.

rint() This option will round the specified argument to the nearest integer. The syntax is `rint(<Number>)`. For example `rint(5.6)` would return `6`.

sin() This option will return the sine of an angle in radians. The syntax is `sin(<radians>)`. For example `sin(pi()/2)` would return `1`.

sqrt() This option will return the square root of the specified number. The syntax is `sqrt(<Number>)`. For example `sqrt(64)` would return `8`.

tan() This option will return the tangent of an angle in radians. The syntax is `tan(<radians>)`. For example, `tan(pi()/4)` would return `1`.

toDegrees() This option will convert an angle measured in radians to degrees. The syntax is `toDegrees(<Number>)`. For example, `toDegrees(pi())` would return `180`.

toRadians() This option will convert an angle measured in degrees to radians. The syntax is `toRadians(<Number>)`. For example, `toRadians(180)` would return `3.142`.

toString() This option will convert a number into a string. The syntax is `toString(<Number>, <Number of Decimals>, <Round Up(True/False)>)`. For example `toString(12.216, 2, True)` would return "`12.22`" as a string.

1.8.2.8.2. String Functions

The following functions take string arguments:

getAllRowData() This option returns whole row of data in a format like `(COLUMN1_NAME) VALUE1 (COLUMN2_NAME) VALUE2 ...` Does not require any arguments.

getHeader() This option will return the column name for a column field. The syntax is `getHeader(<Column Field>)`. For example, if you have a column named `UnitPrice`, `getHeader({UnitPrice})` would return `UnitPrice` as a string.

getPage() This option will return the current page number. The syntax is `getPage()` without any arguments.

getTotalPages()

This option will return the total number of pages for the current report. The syntax is `getTotalPages()` without any arguments.

getTotalSections()

This option will return the total number of sections (i.e. the number of pages needed horizontally) for the current report. The syntax is `getTotalSections()` without any objects.

indexOf()

This option will return the first index value where a specified pattern within a string occurs. There are two syntaxes for this function.

1. `indexOf(<String>, <Pattern>)`. For example `indexOf("Banana", "an")` would return `1`.
2. `indexOf(<String>, <Pattern>, <Starting Index>)`. For example, `indexOf("Banana", "an", 2)` would return `3`.

insert()

This option allows you to insert new characters into a string. The syntax is `insert(<String>, < Character Number>, <New Characters>)`. For example `insert("Wood Natural Furniture", 4, "View")` would return `WoodView Natural Furniture`.

lastIndexOf()

This option will return the last index value where a specified pattern within a string occurs. There are two syntaxes for this function:

1. `lastIndexOf(<String>, <Pattern>)`. For example `lastIndexOf("Banana", "an")` would return `3`.
2. `lastIndexOf(<String>, <Pattern>, <Starting Index>)`. For example, `lastIndexOf("Banana", "an", 2)` would return `1`.

replace()

This option allows you to replace one set of characters in a string with another. There are two syntaxes for this function.

1. `replace(String, Character Start Number, Character End Number, New Characters)`. For example, `replace("Today is a rainy day", 11, 16, "sunny")` would return `Today is a sunny day`.
2. `replace(<String>, <Old Characters>, <New Characters>)`. For example, `replace("Today is a rainy day", "rainy", "sunny")` would return `Today is a sunny day`.

setMaxLength()

This option allows you to set the maximum length of a string. The syntax is `setMaxLength(<String>, <Maximum Number of Characters>)`. For example, `setMaxLength("You're a firefighter",13)` would return `You're a fire`.

strcmp()

This option compares two strings and returns the lexicographical difference between them. If the first argument is larger than the second one, the result is positive. If the first argument is smaller than the second one, the result is negative. If the strings are same, the result is zero. The syntax is `strcmp(<String>, <String>)`. For example, `strcmp("a", "c")` would return `-2`.

strcmpIgnoreCase()

This option compares two strings and returns the lexicographical difference between them while ignoring case. The syntax is `strcmpIgnoreCase(<String>, <String>)`. For example, `strcmpIgnoreCase("a", "A")` would return `0`.

strcat()

This option concatenates multiple strings together. The syntax is `strcat(<String>, <String>, <String>...)`. For example `strcat("Wood", "View")` would return `WoodView`. You can also use the `"+"` operator to concatenate strings.

strlen()

This option will return an integer indicating the number of characters in a specified string. The syntax is `strlen(<String>)`. For example `strlen("ReportDesigner")` would return `14`.

substring()

This option will return a portion of a string as specified by an argument. There are two syntaxes for this function:

1. `substring(<String>, <Character Start Number>)`. For example, `substring("unhappy", 2)` would return `happy`.
2. `substring(<String>, <Character Start Number>, <Character End Number>)`. For example `substring("smiles", 1, 5)` would return `mile`.

trim()

This option will remove any leading or trailing spaces from the specified string. The syntax is `trim(<String>)`. For example, `trim(" Hello ")` would return `Hello`.

toLowerCase()

This option will render any uppercase letters within the specified string to lowercase. The syntax is `toLowerCase(<String>)`. For example `toLowerCase("ABCdef")` would return `abcdef`.

toNumeric()

This option will turn a string into a double. The syntax is `toNumeric(<String>)`. For example `toNumeric("425.52")` would return `425.52`. For this formula to work correctly, the string argument must contain numeric characters.

toUpperCase()

This option will render any lowercase letters within the specified string to uppercase. The syntax is `toUpperCase(<String>)`. For example `toUpperCase("ABCdef")` would return `ABCDEF`.

1.8.2.8.3. Date/Time Functions

The following functions use date/time arguments. Some of the date/time functions use a special argument that indicates a calendar field. To specify a calendar field argument, type one of the following without any delimiters.

ERA
YEAR
MONTH
WEEK_OF_YEAR
WEEK_OF_MONTH
DAY_OF_MONTH
DAY_OF_YEAR
DAY_OF_WEEK
DAY_OF_WEEK_IN_MONTH
AM_PM
HOUR
HOUR_OF_DAY
MINUTE
SECOND
MILLISECOND

addTime()

This option adds a specified amount of time to a given date/time. The syntax is `addTime(<Date/Time>, <Calendar Field>, <Number>)`. The number specifies the amount that the calendar field should be added to the date/time object. For example, `addTime(12/5/1998, MONTH, 5)` would return `May 5, 1999`, and `addTime(12/5/1998, DAY_OF_MONTH, -25)` would return `Nov, 10 1998`.

getAmPm()

This option will return AM or PM as a string for a given time. The syntax is `getAmPm(<Time>)`. For example, `getAmPm(13:24:00)` would return `PM`.

getCurrentDate()

This option will return the current date from the system. The syntax is `getCurrentDate()` without any arguments.

getCurrentDateTime()

This option will return the current date and time from the system. The syntax is `getCurrentDateTime()` without any arguments.

getCurrentTime()

This option will return the current time from the system. The syntax is `getCurrentTime()` without any arguments.

getDateTime()

This option will return the value of a specified calendar field for a given date/time. The syntax is `getDateTime(<Date/Time>, <Calendar Field>)`. For example, `getDateTime(12:24:00, MINUTE)` would return `24`, and `getDateTime(10/10/2001, DAY_OF_WEEK)` would return `4` (meaning Wednesday).

getDayDifference()

This option will return the difference in days as an integer between two dates. The syntax is `getDayDifference(<Date>, <Date>)`. For example, `getDayDifference(5/1/2001, 3/1/2001)` would return `61`. You can also use the "-" operator to return the difference between dates.

getDayOfWeek()

This option will return the day of the week for the specified date. The syntax is `getDayOfWeek(<Date>)`. For example, `getDayOfWeek(10/10/2001)` will return `Wednesday` as a string.

getWeekOfYear()

This option will return the week of the year for the specified date. The syntax is `getWeekOfYear(<Date>)`. For example, `getWeekOfYear(10/10/2001)` will return `41`.

getEra()

This option will return the era for the specified date. The syntax is `getEra(<Date>)`. For example, `getEra(10/10/2000)` would return `AD` as a string.

getMonth()

This option will return the month for the specified date. The syntax is `getMonth(<Date>)`. For example, `getMonth(10/10/2000)` would return `October` as a string.

rollTime()

This option is a time rolling function. The syntax is `rollTime(<Date/Time>, <Calendar Field>, <Number>)`. Unlike the `addTime()` function, this function will only adjust the specified calendar field by the specified amount and will have no effect on the other fields. For example `rollTime(12/5/1998, MONTH, 5)` would return `May 5, 1998`, and `rollTime(12/5/1998, DAY_OF_MONTH, -25)` would return `Dec 11, 1998`. For the latter example, when the count reaches the beginning of the month, it starts over at the end without changing the month field.

printDate(), printDateTime(), printTime()

These three functions allow you to set the way in which date/time information is displayed beyond what is capable using the data formatting options. These functions will return date and/or time information as a string. Their syntax is as follows:

```
printDate(Date, Date Format)
printDateTime(Date, Date & Time Format)
printTime(Time, Time Format)
```

The date and/or time format is entered as a series of characters and delimiters. Letters are used to represent different elements of date/time data. The characters and what each of them represent are listed below:

Character	Represents	Output (text/number)	Example
G	era	text	AD
y	year	number	1996, 96
M	month in year	text or number (dependson length)	July, Jul, 07
d	day in month	number	10

Character	Represents	Output (text/number)	Example
h	hour am/pm (1-12)	number	1
H	hour 24 hr. (0-23)	number	18
m	minute in hour	number	30
s	second in minute	number	55
S	millisecond	number	978
E	day in week	text	Tuesday, Tue
D	day in year	number	189
F	day of week in month	number	2 (as in 2 nd Wed. in July)
w	week in year	number	27
W	week in month	number	2
a	am/pm marker	text	AM, PM
k	hour 24 hr (1-24)	number	24
K	hour am/pm (0-11)	number	0
z	time zone	text	Pacific Standard Time, PST

You can piece together almost any combination of these characters to produce a date expression in any format you want. The count of groups of characters determines the form that the element will take. For text elements with four or more characters in a group, the full form of the element will be used. If less than four characters are used, the short form will be used, if it exists. For example `EEEE` would return `Monday` and `EE` would return `Mon`. For month M which can be displayed as either text or a number, four or more in a group will display the full version, three will display the abbreviation, and two or less will display the number form.

For numeric elements, the count of characters is the minimum number of digits that the element will take. Shorter numbers will implement leading zeros. For example if the day of the date is 2, `dd` would return `02` and `d` would return `2`.

Any character that is not a-z or A-Z like “;”, “:”, “@”, etc can be inserted anywhere within the string expression and will display as they are entered. You can also insert words and expressions by enclosing them within single quotes (type two single quotes to insert an apostrophe as text). Several examples are listed below:

`printDateTime(10/10/2001 21:15:12, "MMMM dd, yyyy 'at' hh:mm a z")` would return `October 10, 2001 at 9:15 PM PDT`

`printDate(10/10/2001, "EEE MMM dd'th', yyyy")` would return `Wed Oct 10th, 2001`

`printTime(13:22:12, "h 'o''clock' m 'minutes and' s 'seconds'")`would return `1 o'clock 22 minutes and 12 seconds`

`printDateTime(10/10/2001 13:22:12, "MM/dd/yy HH:mm:ss")` would return `10/10/01 13:22:12`

If you do not enter any information for the date/time format, the date will display as yyyy-mm-dd and the time will default to HH:mm:ss.S. So `printDateTime(10/10/2001 13:22:12)` would return `2001-10-10 13:22:12.0`.

toDate()

This option will convert a long integer into a timestamp. The syntax is `toDate(<Number>)`. For example, `toDate(1025039526306)` would return `Jun 25, 2002 2:12:06 PM`.

1.8.2.8.4. Table Data Functions

The following functions take numeric arguments:

.row()

This option will return a value of a column field according to the row index. To do this, you can append a column field with the following syntax: `.row(Row Index)`. The row index is the index value associated with each row in a report starting with 0. For example `{ProductName}.row(3)` would return the fourth value of the ProductName field.

.getScriptValue()

This option will return value of a column field according to the row index after applying script. If there is no script applied on the column, it returns the same value as `.row()`. See Section 1.9.2.9 - Accessing Values Computed by Other Scripts for more details.

1.8.2.8.5. Other Functions

The following functions take string arguments:

first()

This option will return the first value of a column field. The syntax is `first(<Column Field>)`. For example `first({UnitPrice})` would return the first value of the UnitPrice field.

last()

This option will return the last value of a column field. The syntax is `last(<Column Field>)`. For example `last({UnitPrice})` would return the last value of the UnitPrice field.

1.8.2.8.6. Custom Functions

In addition to the functions provided with EspressReport, it is also possible to include your own custom functions in the Formula Builder when launching the Report Designer via API. Every Java function is supported. For more information about this feature, please see Section 2.3.5.7.11.6 - Open Report Designer with Custom Functions.

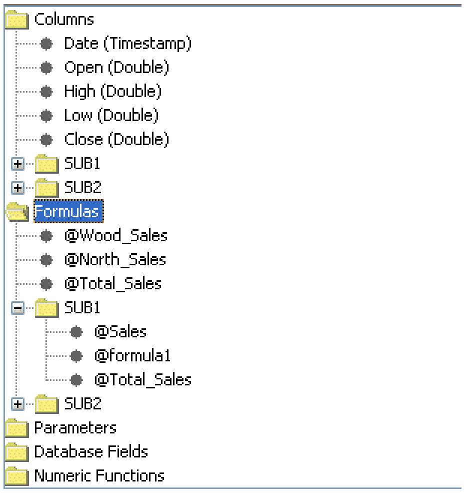
1.8.3. Subreport Formula Access

A main report can access its sub-reports' columns and formulas and use them as its own by using the prefix `SUB#` where "#" is the number of the subreport, beginning with 1. For example, the syntax to access the column `UnitPrice` from sub-report 1 is: `SUB1.{UnitPrice}` or `SUB1.COL(i)` where "i" is the column number of `UnitPrice`. Syntax to access the formula `Sales` from sub-report 1 is: `SUB1.@Sales`.

Note

Accessing a subreport column (i.e. `SUB1.{UnitPrice}`) returns a single value - the value from the first row of the column. You can get the value at a particular row by appending `.row(index)` at the end where "index" is the row number beginning with 0, but it will always only return a single value.

In the formula builder's help panel, under both Columns and Formulas, there is a folder for each subreport:



Sub-report columns and formulas can be used to build more complex expressions or scripts (i.e. `SUB1.@Total + SUB2.@Total`).

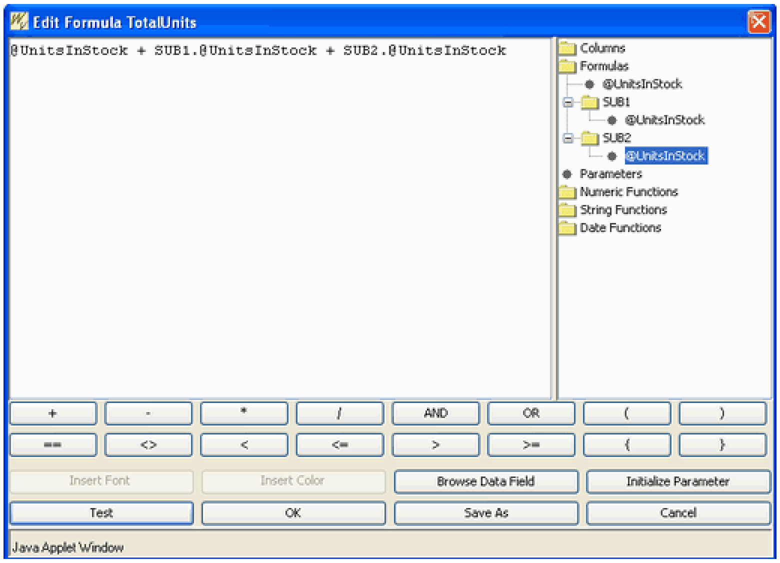
Although sub-report columns and formulas can be used in complex expressions, sub-report columns and formulas cannot be used in aggregation funtions (SUM, COUNT, MAX, AVERAGE, etc). For example, `SUM({UnitPrice})` is a valid formula, but `SUM(SUB1.{UnitPrice})` is not. To accomplish this, instead create a formula `SUB1.{UnitPrice}` and add it as a column into the main report. Then create a second formula which applies aggregation to the first formula (i.e. `SUM(@sub_total)`).

Although the main report can access any of the sub-report's columns or folders, sub-reports cannot access the main report's or any other sub-report's columns or formulas.

If a sub-report formula or column that is used by the main report is deleted, the formula in the main report will display a NULL value when running the report and script will not be applied.

For example, if product information is spread across several databases, but you would like to sum up all the units in stock across the three databases, you could create one main report, add several sub-reports and then create a formula to insert to the report footer adding up all the units in stock. To demonstrate this feature, open a report created from the Inventory XML file with the fields Category Name, Product ID, Product Name, Unit Price, and Units In Stock. Proceed to add subreports to it, one from the Woodview Access database, and one from the Woodview HSQL database, containing the same fields.

Within each subreport, as well as within the main report itself, create a formula called `UnitsInStock`, `SUM({UnitsInStock})`. Then you can create an ultimate formula within the main report using these three functions:



```
@UnitsInStock + SUB1.@UnitsInStock + SUB2.@UnitsInStock
```

Creating a Formula Using Sub-report Formulas

This formula will take the formula values from each of the three reports and add them together to calculate the final value.

This feature is also applicable to linked sub-reports, although the application of it is different. For example, if you create a summary break report containing customer addresses with a sub-report in the group footer detailing all purchases made by that particular customer, you can sum up all the sales info within the main report.

First, create a formula within the sub-report entitled `Sum_Sales` which sums up all the sales in the sub-report. Return to the main report and create a formula `sub_total`, which is simply `SUB1.@Sum_Sales`. Insert it into the Table Data Section. It can be made invisible if you want to, but it must be in the Table Data Section. Then create a second formula in the main report, `total`, `SUM(@sub_total)`. This formula is added to the report footer and will sum up all the sales columns from all of the sub-reports.

1.9. Scripting

Cell scripting or conditional formatting allows you to dynamically modify certain object properties when specific conditions are met. Scripting allows you to highlight certain data values and to present a report that is easier to read. A simple example of this would be a report showing financial results or cash flows. Scripting could be applied to a column field so that negative numbers are shown in red, dynamically highlighting areas of revenue or cash shortages.

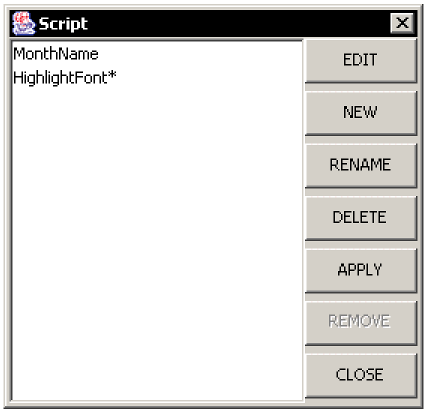
In EspressReport, scripts are element specific, meaning that in order to dynamically change the properties of a report cell, a script must be applied to that specific cell. Scripts can be applied to labels, column fields, formulas, or report sections.

Note

Some of the syntax for EspressReport cell scripts has changed greatly between v2.51 and v3.0. In most cases, the deprecated syntax will continue to work correctly. However, it is recommended that you check older templates to make sure the scripts still function correctly if you have upgraded to v3.0 or higher from v2.51 or lower.

1.9.1. Creating a Script

To create a script, first select a cell you want to apply the script to and then select *Scripting* from the *Format* menu or click the *Scripting* button on the toolbar. This will bring up a list of all the scripts that have been defined within the report.

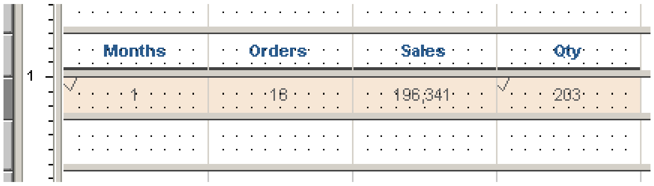


Script List Dialog

Like formulas, scripts are maintained on a report wide basis and can be re-used in multiple places in the report. However, unlike formulas, scripts cannot be inserted into the report, they can only be applied to existing report elements. A star next to a script name in the list indicates that the script is applied to the current cell.

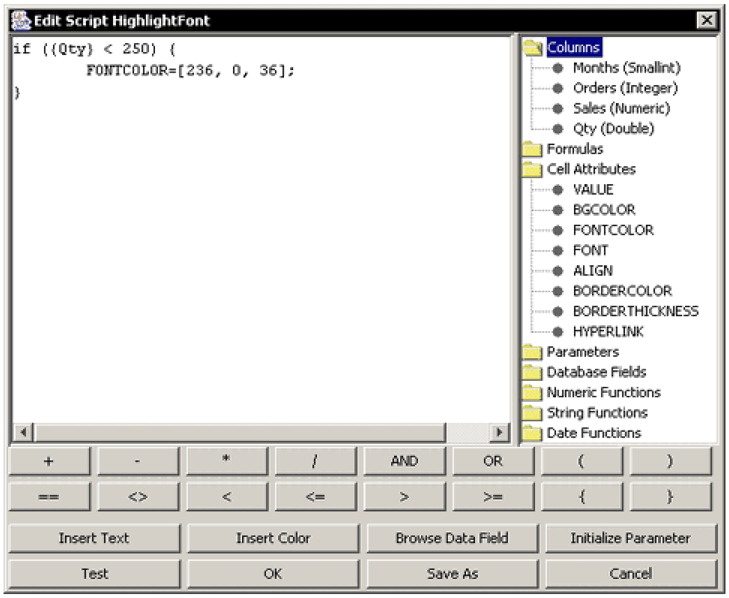
From this dialog, you can edit an existing script, create a new script, rename a script, delete a script, apply a script to the current element, and remove a script from the current element. If a script is being used, it can still be deleted (a message will appear asking if it is ok to proceed) and the script is first removed from any cell before being deleted. If the script is used by a security level, the script has to be removed manually first before it can be deleted. To apply a script to the currently selected report element, select a script from the list and click the *Apply* button. The script list will close and the script will be applied.

Any report element that has a script applied will have a small check mark in the upper left corner of the cell. Scripts will not execute until you preview or export the report.



Script Indicators in Design View

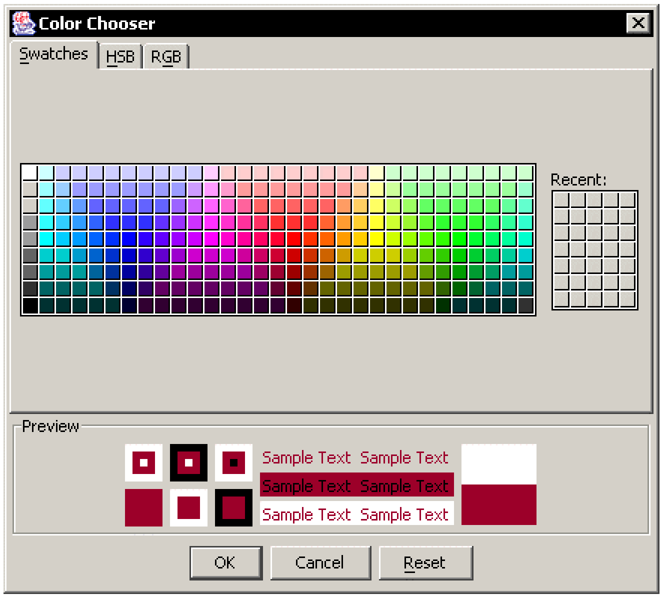
If you select to create a new script by clicking the *New* button from the script list dialog, you will be prompted to specify a name for the new script. Once you specify a name, the Formula Builder will open, allowing you to construct a script.



```
if ((Qty) < 250) {
        FONTCOLOR=[236, 0, 36];
}
```

Formula Builder (Scripting)

Scripts are developed in the same formula builder interface as formulas. Although scripts use a different syntax, all of the formula fields, operators, and functions are also accessible in scripts. For more information about formulas and the Formula Builder, please see Section 1.8 - Using Formulas & the Formula Builder. In addition to the options available for formulas, there are two additional options available when creating scripts. The *Cell Attributes* folder on the right panel contains a list of the cell attributes (in addition to the value) that can be modified with a script. Attributes and their use are discussed in Section 1.9.2.1 - Formatting Actions. The *Insert Color* button will bring up a color selection dialog that allows you to visually pick a color for a cell attribute. Selecting a color will automatically insert an array of RGB values into the script.



Script Color Dialog

1.9.2. Script Syntax

For most scripts the syntax is fairly simple and consists of two basic components: condition and action.

```
if (Condition) {Action(s);}
```

The condition is generally a Boolean expression derived using one of the Boolean operators discussed in Section 1.8.2.7.2 - Boolean Operators For example, `({Quantity} < 5)` would return `true` when the value of the Quantity field is lower than 5. The AND and OR operators can be used to check multiple conditions. For example, `(({Quantity} < 5) AND ({InStock} == "Yes"))` would return `true` when the value of the Quantity field is lower than 5 and the InStock field equals `"Yes"`.

So a simple script that changes the font color to red when negative values are present in the report field would look something like this:

```
if ({SubTotal} < 0) {
    FONTCOLOR=[255,0,0];
}
```

More complex scripts can be created by specifying multiple conditions for multiple actions, using if else syntax.

```
if (Condition 1) {Action 1;}
else if (Condition 2) {<i>Action 2</i>;}
else {<i>Action 3</i>;}
```

There is no limit to the number of nesting levels (if else) that can be specified. Multiple actions for a single condition can also be specified.

```
if (Condition 1) {
    Action 1;
    Action 2;
    Action 3;
}
```

This will perform multiple actions when the condition is met. Note that each line of the script must end with a semicolon.

1.9.2.1. Formatting Actions

The following is a list of element attributes that can be modified using scripts.

VALUE: The action syntax that allows you to change the value of the element to which the script is applied is `VALUE=New Value`;. The new value does not necessarily have to be of the same data type as the report element (for example you can replace number with string). However, changing string to number or number to date can cause formatting irregularities.

BGCOLOR: The action syntax that allows you to change the background color of the element is `BGCOLOR=[R,G,B]`; where R is a number for the red value of the new color, G is a number for the green value of the new color and B is a number for the blue value of the new color. You can automatically generate the RGB array using the *Insert Color* button in the Formula Builder.

FONTCOLOR: The action syntax that allows you to change the font color of the element is `FONTCOLOR=[R,G,B]`; where R is a number for the red value of the new color, G is a number for the green value of the new color and B is a number for the blue value of the new color. You can automatically generate the RGB array using the *Insert Color* button in the Formula Builder.

FONT: The action syntax that allows you to change the font, font style, and font size of the element is `FONT=[Font Name,Font Style,Font Size];`. The font name is the name of the font you want to use (i.e. “Dialog”, “Serif”, etc,). The options for font style are `BOLD`, `ITALIC`, or `BOLD+ITALIC`. The font size is a numeric value indication of the font size. For example, `FONT=[Dialog,BOLD,12];` would change the font to 12 point bold Dialog.

ALIGN: This action allows you to change the horizontal alignment of the data within the cell. The syntax is `ALIGN=Position;`. The position options are `left`, `right`, and `center`.

BORDERCOLOR: The action syntax that allows you to change the border color of the element is `BORDERCOLOR=[R,G,B]`; where R is a number for the red value of the new color, G is a number for the green value of the new color, and B is a number for the blue value of the new color. You can automatically generate the RGB array using the *Insert Color* button in the Formula Builder.

BORDERTHICKNESS: The action syntax that allows you to change the thickness of the element's border is `BORDERTHICKNESS=Thickness;`. The thickness is an integer indicating the number of pixels. For example, `BORDERTHICKNESS=2;` would set the border thickness to two pixels.

HYPERLINK: The action syntax that allows you to set the hyperlink properties of the element is `HYPERLINK=[URL,"Hint","Target"];`. The URL is the location you want the link to point to. The hint is the mouse over hint that will appear and the target is the specified target for the link. If you do not want to specify the hint or target, specify an empty string `""`. Note that you will not be able to click on the hyperlinks in the Preview window if you launched the Report Designer from the `bat` file, but the links will still work if you export the report to DHTML or PDF format.

BOOKMARK: This action allows you to specify a named location within the report. When you export to DHTML, it will become an anchor tag. When you export to PDF, it will become a bookmark within the generated PDF file. The syntax is `BOOKMARK=Name`. The bookmark can be any string value.

XPOSITION: The action syntax that allows you to change the X position of a cell within a section is `XPOSITION=Position;` (in inches or centimeters depending on which measurement is selected). For example, `XPOSITION=1.2;` would set the left edge of the report element 1.2 inches/centimeters away from the left edge of the section.

YPOSITION: The action syntax that allows you to change the Y position of a cell within a section is `YPOSITION=Position;` (in inches or centimeters depending on which measurement is selected). For example, `YPOSITION=0.3;` would set the top edge of the report element 0.3 inches/centimeters away from the top of the section.

WIDTH: The action syntax that allows you to set the width of a report element is `WIDTH=Size;` (in inches or centimeters depending on which measurement is selected). For example, `WIDTH=1.5;` would set the width of the element to 1.5 inches/centimeters.

HEIGHT: The action syntax that allows you to set the height of a report element is `HEIGHT=Size;` (in inches or centimeters depending on which measurement is selected). For example, `HEIGHT=0.25;` would set the height of the element to 0.25 inches/centimeters.

ROTATION: The action syntax that allows you to set the rotation of a report element is `ROTATION=Angle;` (in degrees, where negative value brings counterclockwise rotation, applicable values are `-90, 0` and `90`). For example, `ROTATION=-90;` would set the angle of the element to 90 degrees counterclockwise.

VISIBLE: This action allows you to control whether a cell is visible or not. The syntax is `VISIBLE=True/False;`. Setting visible to false will make the report element invisible.

In addition to modifying these properties for the current cell, you can also retrieve these properties from other report elements. You can retrieve the value from a report element in the same manner that you do in formulas either `{Column Field}`, `@Formula`, `:Parameter`, etc, or using the `id()` function for other cells or labels. For more information about this, please see Section 1.9.2.3 - Using Formulas.

You can retrieve all other attributes using the following syntax `ATTRIBUTE(Report Element)`. For example, `FONTCOLOR(id(RPT_HDR_LB0))` would return an integer array `[R,G,B]` representing the font color of a label in the Report Header section. You can use this to run comparisons based on attributes or to match the attributes of the current cell with those of another report element. For example, `BGCOLOR=BGCOLOR({ProductName});` would change the current cell's background color to match that of the ProductName column.

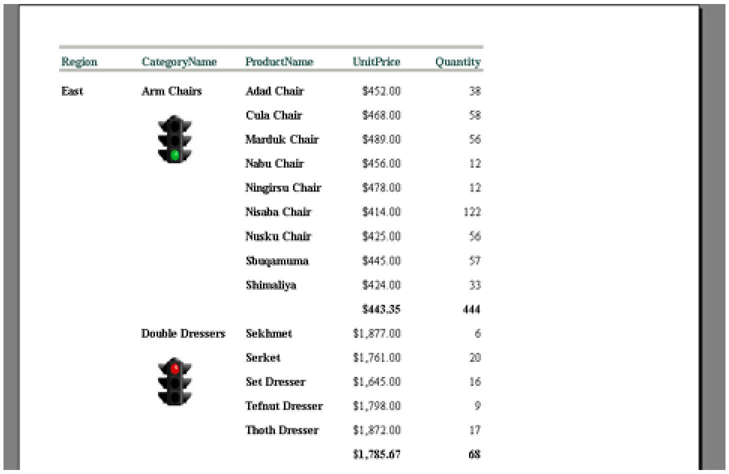
Depending on which type of element the script is being applied to, certain formatting actions may have no effect. For example, you can set scripts on lines and rectangles, but you can only modify their bounds and colors.

1.9.2.1.1. Scripting Image URLs

Utilizing the if else conditionals and the Image URL formatting option, you can create image URL scripts that change images dynamically based on data in your report. Suppose you want to include two simple images to help readers determine if the sales for a particular category has achieved the desired goal. Let's say that the sales goal for each category is 100 units sold. Using this information, you can write the following script to display the success image when the goal is met and the failed image when the goal is not met.

```
if(@SUM_Quantity > 100) {
    value = "http://www.quadbase.com/images/green.jpg";
} else {
    value = "http://www.quadbase.com/images/red.jpg";
}
```

Create a new formula cell that has a string data type (you can simply use `""` for the content) and apply this script to the cell. Then set the data formatting for this cell to be `"Image Url"`. More information about data formatting can be found in Section 1.5.7.3 - Data Formatting for Formulas and Column Fields. Here is how the report will look like using this script:



Region	CategoryName	ProductName	UnitPrice	Quantity
East	Arm Chairs	Adad Chair	$452.00	38
		Cula Chair	$468.00	58
		Marduk Chair	$489.00	56
		Nabu Chair	$456.00	12
		Ningirsu Chair	$478.00	12
		Nisaba Chair	$414.00	122
		Nusku Chair	$425.00	56
		Shuqamuna	$445.00	57
		Shimaliya	$424.00	33
			$443.35	444
	Double Dressers	Sekhmet	$1,877.00	6
		Serket	$1,761.00	20
		Set Dresser	$1,645.00	16
		Tefnut Dresser	$1,798.00	9
		Thoth Dresser	$1,872.00	17
			$1,785.67	68

Scripted Image URL

1.9.2.2. Variables & Arrays

In addition to the if else statements, you can also define variables and arrays in a script. Variables and arrays are defined at the beginning of the script (before any statements).

1.9.2.2.1. Variables

If you have a constant or expression that you intend to use in multiple places in a script, it can often be easier to define a variable rather than re-typing the constant or expression in every place where it is used. A variable can be declared at the beginning of a script using the following syntax: `DataType VariableName;`. The data type is the type for the variable value and can be either Number, Boolean, String, or Date.

Note

`DateVar` is used as the data type declaration for Date variables. This avoids confusion with the date() function. The variable name can be anything except an operator or function name.

The following example uses variables which contain an expression.

```
String Line1;
String Line2;
Line1 = {Address1} + "\n";
Line2 = {City} + ", " + {State} + " " + {Zip};

if ({Address2} == NULL) {

    VALUE = Line1 + Line2;
} else VALUE=Line1 + {Address2} + "\n" + Line2;
```

This script pieces together several report fields to create an address entry. The first two lines of the script declare the variables. The script then concatenates fields and strings depending on whether there is an Address2 entry or not. Using the variables prevents having to re-type the expressions. Note that "\n" is used to break the text to a new line.

You can also assign a value to a variable as part of an action in a script. You can use this to modify the variable value based on some conditions and then return the variable value to the cell. This is useful if you only want to write one value statement. The following example assigns a value to a variable:

```
if ({Flags} == "M") {
    LinkAppend = "Main";
} else if ({Flags} == "R") {
    LinkAppend = "Remote";
} HYPERLINK = ["StatPage.html#" + LinkAppend, "", ""];
```

In this example, a variable is defined without an initial value. The value of the variable is assigned based on a string flag in the report. At the end of the script a hyperlink is defined for the element that uses the variable value to specify a page location.

1.9.2.2.2. Arrays

In addition to variables, you can also declare arrays at the beginning of a script. Specific values from the array can be retrieved later. This is useful if you have a number of fixed values that you will be using in different places in the script like months or days of the week. An array declaration is similar to a variable declaration and uses the following syntax: `DataType[] ArrayName = [Value1, Value2, Value3, ...];`. Data types for arrays are the same as for variables and can be one of Number, Boolean, String, or Date. The variable name can be anything except an operator or function name. The initial value assignment is optional.

You can retrieve a value from an array using the following syntax: `ArrayName[Index].` The following example uses an array to replace the numeric values for a Month column (1-12) and replaces them with the proper month names.

```
String[] MonthNames = ["January","February","March","April",
"May","June","July","August","September","October",
"November","December"];

VALUE = MonthNames[({Month} - 1)];
```

Notice that the index specified is `{Month}-1`. This is because the first month value is 1 and the first index value for the array is 0.

Like variables, you can also assign or overwrite values in an array as part of an action. The syntax for this is `ArrayName[Index]=Value;`.

1.9.2.3. Using Formulas

You can use any of the EspressReport formulas from the main report or any sub-reports (for more on accessing sub-report formulas, see Section 1.8.3 - Subreport Formula Access) in both the condition and action components of the script, as well as for variable values. Most commonly formulas are used to retrieve values either from the current element or other report elements. You can retrieve the value from column fields using `{Column Field}`, the value from existing report formulas using `@FormulaName` and the user supplied values for query and formula parameters using `:ParamName` and `?ParamName` respectively.

In addition to these methods, there is one additional function that can be used in scripts.

this() - This function returns the value of the current element (i.e. the cell to which the script is applied). The syntax is `this()` without any arguments. The function can be useful if you're reusing scripts or if you want to ensure that the script is referring to the cell to which it is applied. For example `If (this() < 0) {FONTCOLOR=[255,0,0];}` would turn the font red, then the value of the current cell is less than zero.

You can also use the formulas to build expressions for comparison or to change an element's value. For more information about EspressReport formulas and their syntax, please see Section 1.8 - Using Formulas & the Formula Builder.

1.9.2.4. Checking for Nulls

Using cell scripts you can check for null values within report columns. To specify a null as a constant, simply use the word `null` without any delimiters. So a simple script that checks for nulls would look like this:

```
if (this() == null) {
    VALUE="yes";
}
```

Note

Even if you have set the null data handler in Report Designer, you can still use script in this manner. This is because the script will execute before the null handler has a chance to convert the content of the cell.

1.9.2.5. Row-Specific Options

Using scripts, you can retrieve and modify values for specific rows of column fields. Normally when you retrieve or replace the value of a column field you will get/replace the first value from the column or the current row if the script is applied to a field in the Table Data section. However, you also have the option of specifying which row you would like to get/replace. To do this, you can append a column field with the following syntax: `.row(Row Index)`. The row index is the index value associated with each row in a report starting with 0. For example `{ProductName}.row(3)` would return the fourth value of the ProductName field.

You can obtain the current row index as well as the total number of rows in the report using the `getRowIndex()` and `getTotalRowIndex()` functions respectively. The following example uses this feature to hide repeated values in a column field.

```
if (({Category}.row(getRowIndex()) == {Category}.row(getRowIndex()-1)) AND
 (getRowIndex() <> 0)) {
    FONTCOLOR=[255,255,255];
}
```

This example assumes that the background color for the report is white and hides the field by adjusting the font color accordingly.

1.9.2.6. Loops

In addition to conditional if else statements, EspressReport also allows you to add loops to scripts. Used in conjunction with variables and row-specific options/formatting, loops are useful if you need to perform running totals/calculations independent of the current row or section of the report. The basic syntax for loops is:

```
while (Condition) {Action(s);}
```

The specified actions will be performed as long as the condition (Boolean expression) is true.

For example, say you have the following report:

Product	Quantity	Price
Chair	6	$327.00

Product	Quantity	Price
Chair	4	$218.00
Chair	12	$418.00
Chair	5	$221.00
Chair	18	$248.00
Table	4	$875.00
Table	2	$1,024.00
Table	8	$967.00
Table	7	$1,106.00

Adding a formula that calculates the total quantity to the Table Footer (`sum({Quantity})`) would return `66`. However, say you only wanted to show the total quantity for chairs in the Table Footer. To do this, you write a loop. Applying the following script to the aggregation.

```
Number i;
Number s;
i = 0;
s = 0;

while (i < getTotalRowIndex()) {

    if ({Product}.row(i) == "Chair") {

        s = s + {Quantity}.row(i);

    }

    i = i + 1;

}

VALUE = s;
```

will return `45` - the sum of the quantity column for `Chair`. The script loops through each row in the report and checks to see if the value for the Product column is “Chair”. If it is, the Quantity value for that row is added to the variable s.

1.9.2.7. Section Scripts

In addition to individual cells, scripts can also be applied to report sections. Based on conditions within the report data, individual section options can be turned on/off and section background colors can be set. To add a script to a report section, click on the button with the section name on the left side of the Designer. You can also right click on a blank portion of the section field and select *Scripting* from the pop-up menu.

Writing section scripts is exactly the same as writing scripts for report elements as they use the same syntax. The only difference is that instead of *Cell Attributes* folder in the Formula Builder, you will see *Section Attributes* folder containing various formatting actions that can be performed with section scripts.

1.9.2.7.1. Formatting Actions for Section Scripts

The following is a list of section attributes that can be modified using scripts (not all options are available for every report section).

BGCOLOR: The action syntax that allows you to change the background color of the element is `BGCOLOR=[R,G,B];` where R is a number for the red value of the new color, G is a number for the green value of the new color and B is a number for the blue value of the new color. You can automatically generate the RGB array using the *Insert Color* button in the Formula Builder.

PRINT_ON_NEW_PAGE: This action allows you to turn on/off the print on new page section option. The syntax is `PRINT_ON_NEW_PAGE=True/False;` For more information about the print on new page feature, please see Section 1.5.3 - Section Options.

RESET_PAGE_NUMBER: This action allows you to turn on/off the reset page number section option. The syntax is `RESET_PAGE_NUMBER=True/False;` For more information about the reset page number feature, please see Section 1.5.3 - Section Options.

VISIBLE: This action allows you to control whether a section is visible or not. The syntax is `VISIBLE=True/False;` Setting visible to false will make the section invisible. Using this attribute coupled with the nested section feature (Section 1.5.1.1 - Nested Sections) allows you to create different headers/footers to display depending on the data.

HEIGHT: This action allows you to set the height of a section. The syntax is `HEIGHT=Size;` (in inches or centimeters depending on which measurement is selected). For example `HEIGHT=0.25;` would set the height of the section to 0.25 inches/centimeters.

> **Note**
>
> Scripts that set or refer to section attributes can only be applied to sections. They cannot be applied to other report elements.

1.9.2.8. Commenting Scripts

You can also add comments to a script to explain its design and operation. A comment is preceded by two slashes `//`. Anything following this on the same line is treated as a comment. You can break comments into multiple lines, but each line must begin with `//`.

1.9.2.9. Accessing Values Computed by Other Scripts

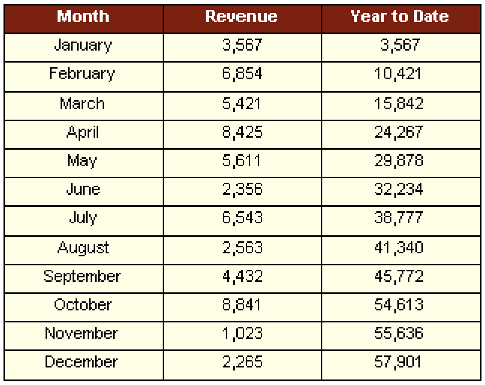
If you have scripts applied to cells in your report, you can access these values using the syntax .`getScriptValue()`. For example, you may have this data:

```
String, int
Month, Revenue
January, 3567
"February", 6854
"March", 5421
"April", 8425
"May", 5611
"June", 2356
"July", 6543
"August", 2563
"September", 4432
"October", 8841
"November", 1023
"December", 2265
```

If you wish to take this data and create a year to date revenue column, you could use the following script, which would display the scripted value from the previous row added to the revenue from the current row. This script is written to be put on a formula consisting of the number 0 as a column in the table data section, refered to within this script as `{formula1}`.

```
if(getRowIndex()==0)

    VALUE={revenue};
else
    VALUE={revenue} + {formula1}.getScriptValue(getRowIndex()-1);
```

This would yield a report like this:

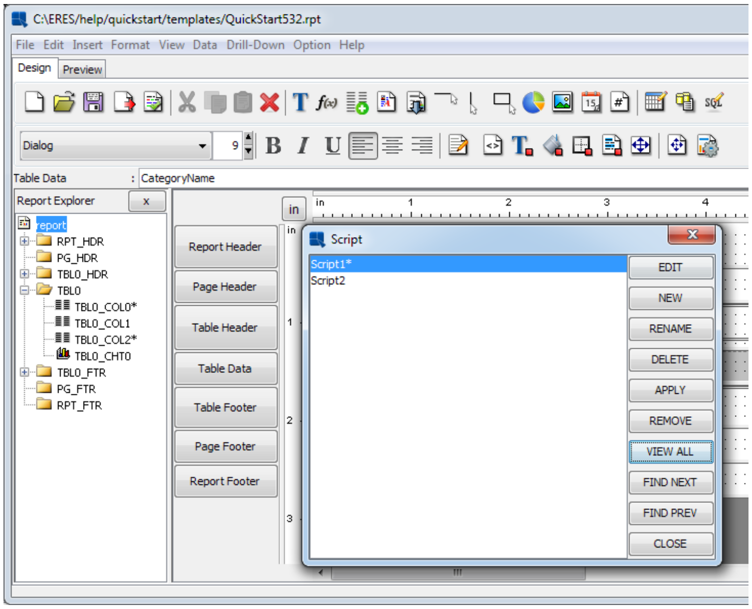


Month	Revenue	Year to Date
January	3,567	3,567
February	6,854	10,421
March	5,421	15,842
April	8,425	24,267
May	5,611	29,878
June	2,356	32,234
July	6,543	38,777
August	2,563	41,340
September	4,432	45,772
October	8,841	54,613
November	1,023	55,636
December	2,265	57,901

1.9.3. Managing Scripts

For reports that contain a number of scripts or scripts applied to a number of elements, there are several options available to help you manage and find the elements that the scripts are applied to. In the script dialog, you will notice that in addition to the standard options such as remove, delete, and rename, there are also buttons to view all, find next, and find previous elements. Make sure that you select a script before using these buttons.

VIEW ALL: This button will open the Report Explorer if it is not already open and highlight every element that the selected script has been applied to with an asterisk. The tree will automatically expand the sections that have highlighted cells. Clicking on one of these elements will center the designer view on that element allowing you to change or remove the applied script. The highlights will be changed if you select another script and click on VIEW ALL again. Closing the script dialog box will remove all the highlights and the explorer tree will collapse.



View All Result

FIND NEXT and FIND PREVIOUS These buttons are used to cycle through elements that the selected script has been applied to. If you already have an element selected in the Designer when clicking on these buttons, the search will begin at the current element. If no elements are selected, the search will begin at the beginning of the report. If you select another script and click FIND NEXT, the asterisk will shift to the newly selected script and the first element that has the script applied will be highlighted in the designer. *FIND NEXT* and *FIND PREVIOUS* only cycle through visible elements. If you want to see invisible elements with the script, use *VIEW ALL* which shows all invisible elements as well as visible elements.

For more information about the Report Explorer, please see Section 1.5.8 - The Report Explorer

1.10. Drill-Down

Often reports are used to display large amounts of data. However, with large data sets (particularly if the data is fairly detailed) displaying everything in one report may not the best way to present the information. One solution to this problem is to create a top-level report that displays only summarized data and allow users to click through to see underlying data. This is the concept of drill-downs.

In EspressReport, users can easily display data in this way. Links can be placed on column fields in the primary (top-level) report to secondary (sub-level) reports. The sub-level reports use parameterized queries that accept the value from the primary report as the input. Hence, sub-level reports will only display data related to the particular value on which the user has clicked.

For example, say you are designing a report to show sales data for an entire year. Rather than creating a report that displays the entire year's worth of data, you could design a top-level report that shows aggregated data by month.

Month	Units Sold	Total Sales
January	5	$10,224.12
February	4	$12,286.14
March	6	$14,415.16
April	5	$12,316.11
May	2	$9,482.64
June	3	$10,116.12
July	7	$17,624.18
August	5	$10,361.54
September	4	$10,116.82
October	3	$8,612.41
November	5	$10,552.24
December	11	$22,614.89

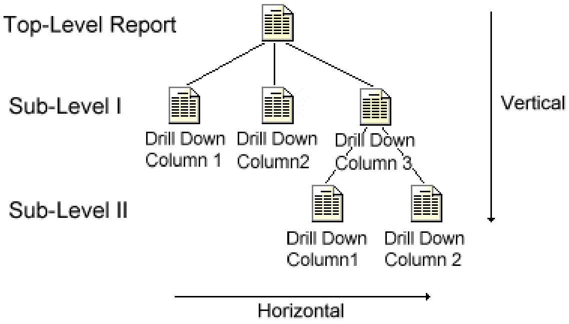
Users viewing the report could then drill-down to see more data for each month. In this case, say the user clicks on *March*. They would then be taken to an underlying report that offers detailed sales information for that month.

Sales Data For March 2001				
Order #	**Customer**	**Product**	**Quantity**	**Total**
10042	Allied Furniture Emporium	Chair	2	$6,425.16
10051	Furniture World	Table	1	$3,114.12
		Dresser	1	$2,116.83
10068	Domus Home & Garden	Chair	1	$800.65
		Dresser	1	$1,958.40
Total:			**6**	**$14,415.16**

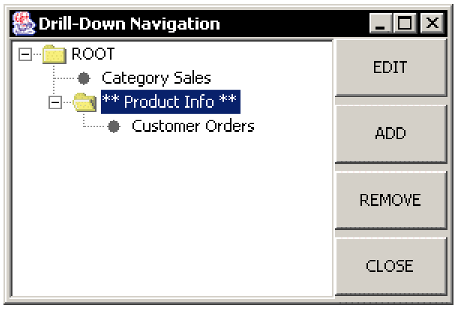
From this report, another layer of drill-down can be added, allowing users to see more detailed customer information. In this case, say the user clicks on *Allied Furniture Emporium*. They would be taken to a third report that shows more information about the customer.

Customer Information		
		Address:
Company:	Allied Furniture Emporium	384 Broad St. Littletown, NY 18322
Contact Name:	Matilda Gladwaller	
Order History:		
Year	**Orders**	**Total Sales**
1999	14	$52,614.18
2000	18	$68,115.89
2001	12	$38,812.12

Using drill-down in this manner allows large amounts of information to be presented in a way that is easy to understand and navigate. EspressReport allows horizontal and vertical layers of drill-down. This means that different columns in that same report can have drill-down (horizontal), or that sub-level reports can have subsequent drill-down reports (vertical). Users can also add drill-down layers to sub-reports.



Within the Report Designer, drill-down is controlled from the Drill-Down menu. Selecting the *Navigate* option from this menu brings up the Navigation window. From this window you can add and edit drill-down reports.



Drill-Down Navigation Dialog

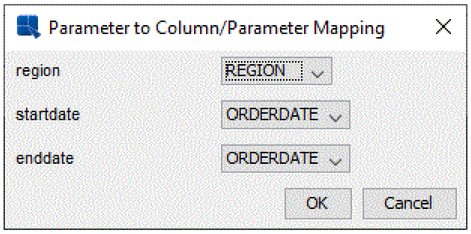
The left side of the Navigation window displays the hierarchy of drill-down reports (see the above diagram). The `ROOT` is the top-level report. The level that you are currently editing is marked with “**”. To edit a different report, select it and click the *EDIT* button on the right side. That report will then open in the Designer, allowing it to be customized. Reports can also be removed by selecting the report and then clicking the *REMOVE* button.

1.10.1. Creating a Drill-Down Report

The first step in creating a drill-down report is to create the top-level report. Because drill-down relies on relationships between data sets, only reports that use parameterized queries, XML queries, or class files can be used for drill-down layers. When designing the top-level report, bear in mind that at least one of the columns or parameters will be passed as a parameter to any sub-level reports. After you finish designing the top-level report, go to the Drill-Down menu and select *Navigate*. The navigation window will open. Select the report under which you would like to add the drill-down layer (if the report has no drill-down, only the ROOT node will be visible), and click the *ADD* button. If you are adding drill-down to a sub-report, first go to the sub-report tab and then load the navigation dialog.

You will then be prompted to create a new report or use an existing report for the drill-down layer. You can use any existing report; however any report for a drill-down layer must have a parameterized query or class file as the data source. If the existing report contains drill-down levels of its own, all drill-down levels will be imported as well. If you select to create a new report, the Report Wizard will open, allowing you to select a data source, report type, and column mapping.

The next step is to map the columns or parameters in the top-level report, to the parameters in the sub-level report. You will be prompted to do this when you select an existing report for the drill-down layer, or when you select the data source for a new report for the drill-down layer. In either case, a dialog will appear prompting you to map the columns or parameters from the top-level report.



region: REGION
startdate: ORDERDATE
enddate: ORDERDATE

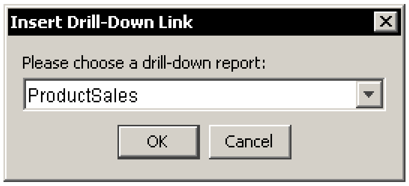
Parameter to Column/Parameter Mapping Window

The options that are available in the drop-down menu are based on data type. For example, if your parameterized query takes a string as a parameter, only columns containing string data can be mapped. If there are multiple parameters for the sub-level query, there will also be the option *None*. This will allow you to create unmapped drill-down reports. For more information, see Section 1.10.5 - Unmapped Drill-Down. Once you specify the drill-down mapping, you will be prompted to specify a name for the sub-level report. Now the sub-level report will open in the Design window, allowing you to format and customize it.

1.10.1.1. Drill-Down Links

By default, a link will be placed on the column in the top-level report which is mapped to the sub-level report. Instead of using the default link, you can place the link on a different column, or multiple columns using the *Drill-Down Link* option from the Drill-Down menu.

To add or remove a drill-down link from a report column, first make sure that you are currently editing the correct template (top or higher level). Then select a column field and select *Drill-Down Link* from the Drill-Down menu. This will bring-up a drop-down list allowing you to select which sub-level report you would like to link to. To remove a drill-down link, select the blank option at the top of the drop-down list.



Please choose a drill-down report:
ProductSales

Drill-Down Link Dialog

1.10.1.1.1. Linking from Charts

In addition to placing links on a column, you can also set drill-down links on a chart. With this feature, users can click on the data point in a chart and traverse to the next level of drill-down. To set a drill-down link on a chart, select the chart object in the Report Designer and select *Drill-Down Link* from the Drill-Down menu. Next, select the sub-level report just as you would for a column.

When you set a drill-down link in the chart depending on the elements in the chart and the number of parameters in the sub-level report, the category, data series, and/or sum by value for the data point that is selected are passed as the parameter value to the sub-level report. In order for this feature to function correctly, the chart will need to use the same data source as the report. The chart should also have the categories mapped to the column that was selected to match the parameter in the sub-level report. For more information about column mapping in charts, please see Section 3.2.2 - Basic Data Mapping and Section 3.4 - Chart Types and Data Mapping.

> **Note**
>
> You can setup linking from any chart type; however, not all chart types will work when exported to PDF. Only two-dimensional bar, column, stack bar, stack column, and 100% column charts will generate links in PDF. The PDF export only supports rectangular image maps.

1.10.2. Multi-Value Drill-Down

In addition to drilling on one field at a time, EspressReport supports the concept of multi-value drill-down. Using this feature, you can drill into several different values from a top-level report at once.

For example say you have a report detailing sales information for 100 different customers. Using the multi-value drill-down, users could select the customers that most interest them and drill into their records at once instead of looking at each customer individually.

1.10.2.1. Creating a Multi-Value Drill-Down Report

Adding layers of multi-value drill-down to a report is almost exactly same as adding regular drill-down layers. The only difference is in the query in the lower-level report. To create a multi-value drill-down layer, the lower-level report must have a multi-value parameter in its query that is mapped to a column in the higher-level report.

Using the previous example, say the query for the lower level has a multi-value parameter for a customer id - something like `Where Customers.CustomerID IN (:CustID)`. If the `CustID` parameter is mapped to the `Customer ID` column in the top-level report, this will automatically create a multi-value drill-down layer. For more information about multi-value parameters, see Section 1.3.2.2.2.1 - Multi-Value Parameters. If you do not wish to create a multi-value drill-down layer, simply use a single value parameter in the lower-level report.

> **Note**
>
> You can have only one report per level when using multi-value drill-down. It does not support linking different columns from the same report to different reports.

You can place column links in multi-value drill-down reports; however, a dialog or form is used to allow users to select multiple column values and drill to the next level. For more information about this, please see Section 1.10.4 - Viewing Drill-Down Reports.

1.10.3. Crosstab Drill-Down

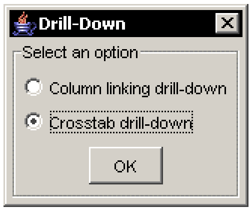
EspressReport provides a special implementation of the drill-down features when the top level report is a crosstab report. Using the crosstab drill-down features, users can click on a cell in the crosstab matrix and drill into information for that particular row or column. For example, assume you have the following report showing sales volume by product and region.

	East	**South**	**Midwest**	**West**
Chair	144	208	131	108
Cabinet	208	114	158	206
Dresser	100	101	112	109

Using standard drill-down configuration, if a user clicks on one of the hyperlinked cells, only the values from that row could be passed to the lower-level report. However, with crosstab drill-down, the column break and row break values for that cell are passed to the lower-level report. Therefore, if the user clicks on the cell value *144*, the values `"Chair"` and `"East"` would be passed to the parameters in the lower-level report.

1.10.3.1. Creating a Crosstab Drill-Down Report

When you select to add a drill-down layer to a crosstab report, you will be prompted whether you would like to create a standard (column linking) drill-down presentation or use crosstab drill-down.



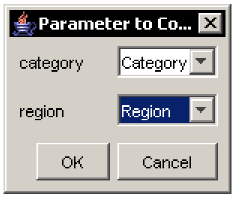
Select an option

○ Column linking drill-down

● Crosstab drill-down

Crosstab Drill-Down Options

If you select to use crosstab drill-down, the setup will continue just as it would for any other drill-down presentation. The only difference comes with parameter mapping. Instead of mapping parameters from the lower level report to columns in the main report, they are mapped to the row break and column break values for the top-level report.



category: Category

region: Region

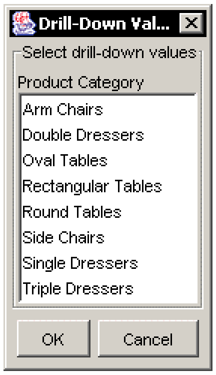
Parameter-Column Mapping for Crosstab Drill-Down

Once you add the drill-down layer, the cells (column break values) in the crosstab matrix for the top level report will automatically be linked to the layer you have added.

1.10.4. Viewing Drill-Down Reports

You can directly navigate and view drill-down reports within the Report Viewer Applet (and the Preview window). When you open a report containing drill-down, you can click on any entry in a column with a drill-down link. You will be taken to the sub-level report, which will take the value for that row as the parameter.

For multi-value drill-down reports, you can select *Select Multiple Drill-Down Values* from the *Data* menu in the Preview window or from the Viewer pop-up menu. This will bring up a dialog containing all the values in the mapped column. You can then select the values you want and click the *Ok* button to go to the next level.



Select drill-down values

Product Category

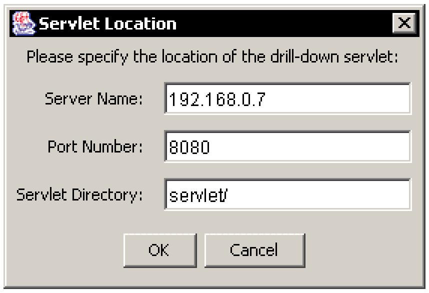
Arm Chairs
Double Dressers
Oval Tables
Rectangular Tables
Round Tables
Side Chairs
Single Dressers
Triple Dressers

Multi-Value Drill-Down Selection Dialog

> **Note**
>
> The templates for sub-level reports are saved in DrillDown directory. If you are viewing reports on another machine, the Report Viewer will look for the templates in that directory, so they have to be moved as well.

Drill-down reports can be exported to any format; however only the DHTML and PDF formats can retain the drill-down functionality (multi-value drill-down reports only support DHTML). Any other format will only export the top-level report. The DHTML and PDF exports work in conjunction with the drill-down report servlet, which is in

DrillDownLinkGenerator directory of the installation. Before exporting the report, you will need to compile servlet code and deploy class file within your application server/servlet runner. When you export to DHTML/PDF from the Designer, a dialog will open prompting you to specify the location of the drill-down servlet.



Please specify the location of the drill-down servlet:

Server Name: 192.168.0.7

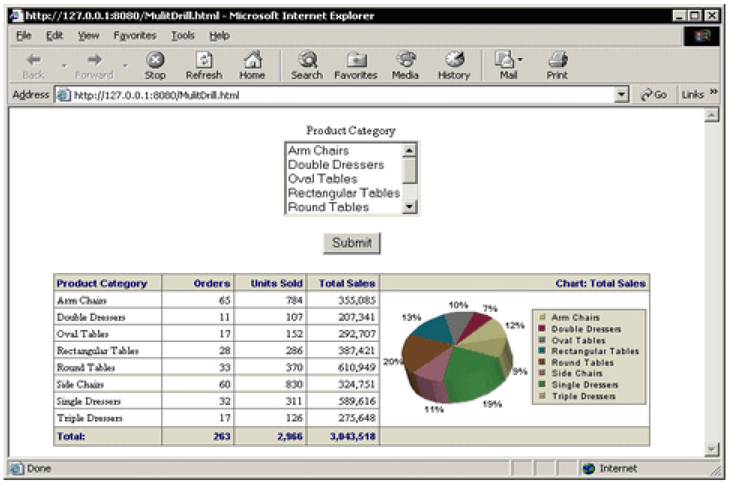
Port Number: 8080

Servlet Directory: servlet/

Drill-Down Servlet Dialog

The first option prompts you for the name of the server. You can either enter a machine name (i.e. machine.domain.com) or the IP address of your application server/servlet runner. The second option prompts you to specify the port number that your application server/servlet runner is using. The third option prompts you to specify the servlet directory of your application server/servlet runner. When you export, this information is used to generate links for the drill-down columns.

For multi-value drill-down reports, the DHTML export generates with a form at the top that allows users to select values to drill into. You will need to specify the same servlet information when exporting a multi-value drill-down report; however, instead of using the drill-down report servlet, it uses the parameter report servlet, which is under ParamReportGenerator directory of the installation. This servlet will need to be compiled and deployed within the application server/servlet runner for these reports to work correctly.



http://127.0.0.1:8080/MulitDrill.html

Product Category

Arm Chairs
Double Dressers
Oval Tables
Rectangular Tables
Round Tables

Product Category	Orders	Units Sold	Total Sales	Chart: Total Sales
Arm Chairs	65	784	355,085	
Double Dressers	11	107	207,341	
Oval Tables	17	152	292,707	
Rectangular Tables	28	286	387,421	
Round Tables	33	370	610,949	
Side Chairs	60	830	324,751	
Single Dressers	32	311	589,616	
Triple Dressers	17	126	275,648	
Total:	263	2,966	3,043,518	

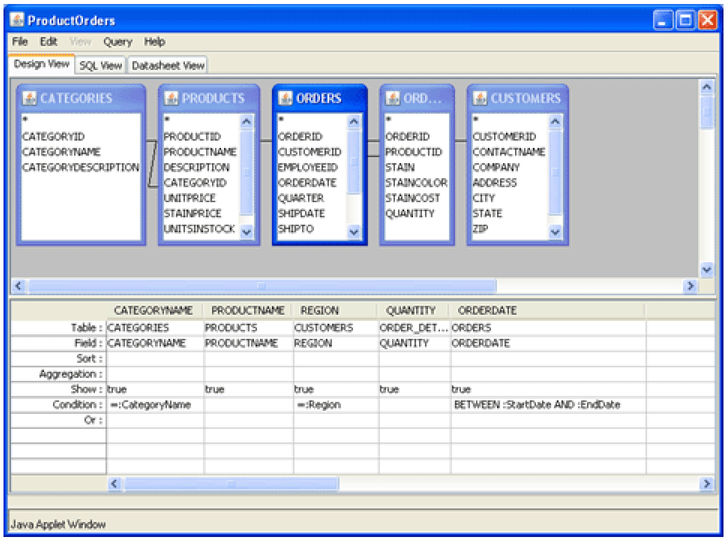
Multi-Value Drill-Down in DHTML

For more information about running drill-down reports, please see Section 2.3.5.1.1 - Sub-Reports, Charts, and Drill-Down Reports.

1.10.5. Unmapped Drill-Down

When you create a sub-level query with multiple parameters, you have the option of leaving some of these parameters unmapped, so that they can be prompted after clicking a drill-down link. To use this feature when selecting mapping for parameters, select **none** for the parameters you wish to remain unmapped. The sub-level report also contains a Data menu option *Preview Parameter Prompt* which determines if the parameter prompt will be displayed when you drill-down from the root level. If this level has never been previewed before, it will be turned off and use the default parameters. Otherwise, it will save the last used parameter values for future use.

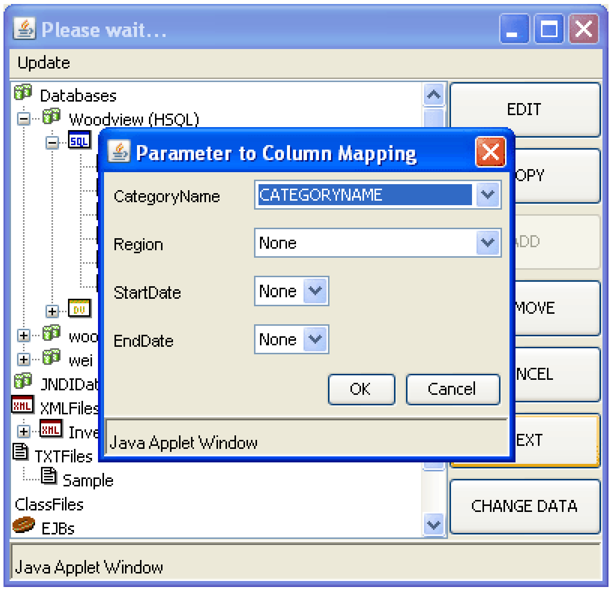
Each sub-level drill-down report requires at least one mapped parameter. Thus, in order to create a report with unmapped parameters, the sub-level query must have at least two parameters. For example:



	CATEGORYNAME	PRODUCTNAME	REGION	QUANTITY	ORDERDATE
Table :	CATEGORIES	PRODUCTS	CUSTOMERS	ORDER_DET...	ORDERS
Field :	CATEGORYNAME	PRODUCTNAME	REGION	QUANTITY	ORDERDATE
Sort :					
Aggregation :					
Show :	true	true	true	true	true
Condition :	=:CategoryName		=:Region		BETWEEN :StartDate AND :EndDate
Or :					

Drill-Down Level Query

Map at least one parameter to a database column and select **none** for the other(s).

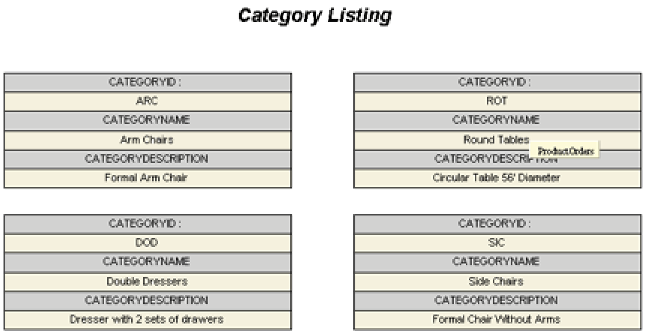


CategoryName: CATEGORYNAME
Region: None
StartDate: None
EndDate: None

Creating Unmapped Parameters

The remainder of the creation of the drill-down report is like any other.

When you preview the report, the root report will be displayed as normal (with a parameter prompt if the root report contains parameters).



Category Listing

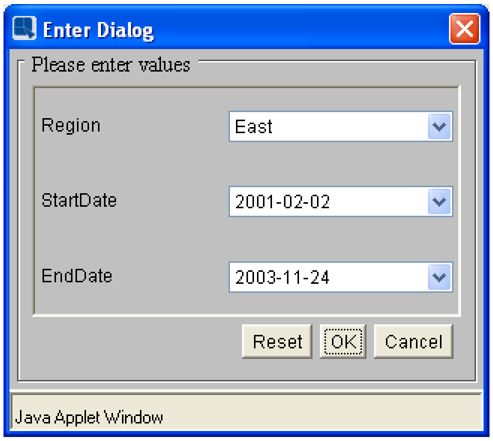
CATEGORYID :
ARC
CATEGORYNAME
Arm Chairs
CATEGORYDESCRIPTION
Formal Arm Chair

CATEGORYID :
ROT
CATEGORYNAME
Round Tables
CATEGORYDESCRIPTION
Circular Table 56' Diameter

CATEGORYID :
DOD
CATEGORYNAME
Double Dressers
CATEGORYDESCRIPTION
Dresser with 2 sets of drawers

CATEGORYID :
SIC
CATEGORYNAME
Side Chairs
CATEGORYDESCRIPTION
Formal Chair Without Arms

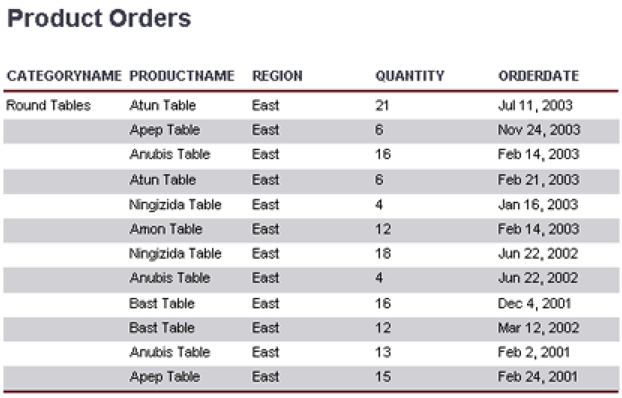
If you click on a link and the *Preview Parameter Prompt* menu option for the drill-down report is checked, you will see the following prompt. Notice that the `CategoryName` parameter is not shown:



Region: East
StartDate: 2001-02-02
EndDate: 2003-11-24

Parameter Prompt for Unmapped Drill-down

The resulting report will look like this:



Product Orders

CATEGORYNAME	PRODUCTNAME	REGION	QUANTITY	ORDERDATE
Round Tables	Atun Table	East	21	Jul 11, 2003
	Apep Table	East	6	Nov 24, 2003
	Anubis Table	East	16	Feb 14, 2003
	Atun Table	East	6	Feb 21, 2003
	Ningizida Table	East	4	Jan 16, 2003
	Amon Table	East	12	Feb 14, 2003
	Ningizida Table	East	18	Jun 22, 2002
	Anubis Table	East	4	Jun 22, 2002
	Bast Table	East	16	Dec 4, 2001
	Bast Table	East	12	Mar 12, 2002
	Anubis Table	East	13	Feb 2, 2001
	Apep Table	East	15	Feb 24, 2001

From the exported file, the default values will be used, but they can be changed from the filter options.

1.11. Sub-Reports

A sub-report is a report within a report. The sub-report is nested entirely within a section of a primary report. Each sub-report has its own data source and design. Using sub-reports you can create reports that use un-related data. You can also create a report that has more than one data table.

In EspressReport, sub-reports are generally unrelated to the primary report (however, parameterized sub-reports can be linked to column fields in the primary report, see Section 1.11.4 - Linked Sub-Reports). Each sub-report uses its own data source and it can be designed independently from the primary report. It also runs entirely within the section in which it is placed in the primary report. You can either use an existing report (`.pak`, `.rpt` or `.xml`) template or create a new one for the sub-report.

1.11.1. Adding a Sub-Report

To add a sub-report to an existing report, click the *Insert Sub-Report* button on the toolbar or select *Insert Sub-Report* from the *Insert* Menu. A second menu will open giving you the option of adding a new sub-report or re-using an existing one. To insert a new sub-report, select *New...*.

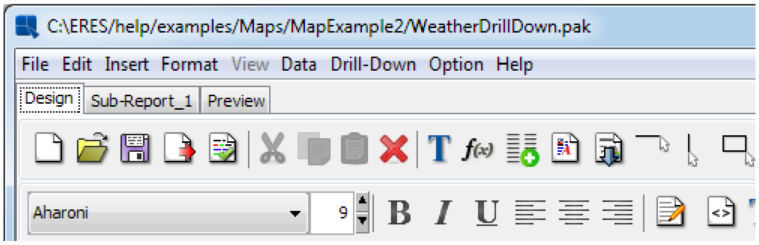
The cursor will turn to cross. Position it where you want to insert the sub-report and click. A dialog box will appear asking you whether you want to use an existing report or create a new one.

You can use any existing report in either `.pak`, `.rpt` or `.xml` format as the sub-report. If you select to use an existing report, you will be prompted to specify the location of the report file. After you specify the file, it will be inserted as a sub-report.

If you select to create a new report for the sub-report, the Report Wizard will launch, allowing you to go through the process of designing the report from data source selection to report mapping. Once you finish with report mapping, the new report will be inserted as a sub-report.

1.11.2. Editing a Sub-Report

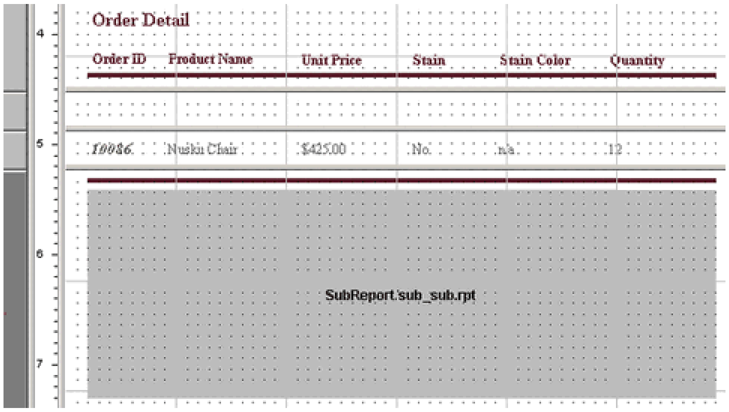
After a sub-report has been added, a new tab will appear at the top of the Report Designer window.



Sub-Report Tab

Selecting the *Sub-Report* tab will open a new design pane that allows you to modify the sub-report. You can modify the sub-report in the same way as you modify any report. For each sub-report you add, a new tab will appear. When you have a sub-report, the *Preview* tab becomes context sensitive. If you click on the *Preview* tab from the *Sub-Report* tab, the preview pane will only contain the sub-report. If you click on the *Preview* tab from the *Design* tab, you will preview the main report with the sub-report embedded.

Within the Design window, the sub-report will appear as a gray rectangle. It can be moved and resized like any object. You can also edit a sub-report by double-clicking on the corresponding rectangle in the Design window.


Order Detail

Order ID	Product Name	Unit Price	Stain	Stain Color	Quantity
10086	Nuski Chair	$425.00	No	n/a	12

SubReport:sub_sub.rpt

Sub-Report in Design Window

Sub-reports run entirely in the section in which they are placed and will generally repeat each time the report section repeats. This means that if the sub-report is placed in the table header section, it will only run once (unless the *Print on Every Page* option is enabled). However, if the sub-report is placed in the Group Header section, the sub-report will repeat for each group in the report.

The exception to this is the table data section. Sub-reports inserted into the table data section will not repeat for each row, but will rather run alongside the data in the primary report. This allows you to create a report with the effect of two data tables running next to each other.

1.11.2.1. Sub-Report sizing

The size of the gray rectangle in the main report Design window represents the size of the sub-report, so if the sub-report is larger than the space you allow in the Design window, portions of the sub-report may be truncated.

Fixing a sub-report's size within the main report may not be the best solution if the number of records in the sub-report vary. You can solve this by setting the sub-report to have variable height. Dynamic sizing for sub-reports is set in the same way as for other report cells using the resize to fit option. For more information about this feature, please see Section 1.5.7.11 - Moving and Resizing Report Elements.

You can also set the width of the sub-report to be dynamic. This feature is most useful for crosstab type reports where the number of columns in the report can vary as the data or filters change. This sets the sub-report to always draw wide enough to fit all the visible columns in the sub-report. To set dynamic width for the sub-report, right click on the sub-report in the Design window and select *Auto Resize Width* from the pop-up menu (To disable this feature, open the pop-up menu and click on this option again).

Note

If you set a sub-report to have variable height, any objects placed directly below the sub-report in the same section of the primary report will not shift to accommodate the resized sub-report. Hence, these objects may be overlapped by the sub-report.

To ensure that objects are not overlapped, it is recommended that you place sub-reports with dynamic height in their own report sections. If necessary, you can add nested sections to the report. For more information about this, see Section 1.5.1.1 - Nested Sections.

1.11.3. Removing a Sub-Report

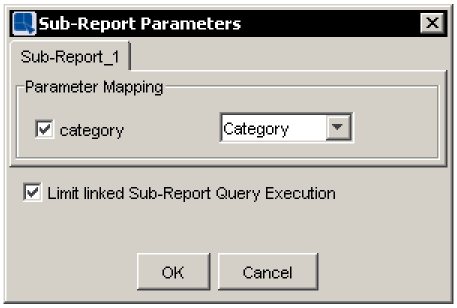
There are two ways to remove a sub-report from the primary report. You can simply select the sub-report object in the primary report and delete it. This will remove the sub-report from the primary report but it will keep it available. This means that the sub-report tab is still active and the sub-report can be re-inserted into the primary report.

The second method is to select *Remove Sub-Report* from the *Edit* menu. A second menu will open with a list of all the sub-reports embedded in the current report. Select the sub-report you want to remove. The sub-report tab will be removed and the sub-report will no longer be available.

1.11.4. Linked Sub-Reports

Sometimes you may want sub-reports to be coordinated with the data in the primary report. For example, if you have grouped data, you may want to filter a sub-report so that it shows data pertinent to each group. You can use linked sub-reports to do this.

Sub-reports are linked to primary reports using query parameters. So to create a linked sub-report, the sub-report must have a parameterized query. After you create the sub-report, you can link it by selecting *Sub-Report Parameter Mapping* from the *Data* menu. This will bring up a dialog allowing you to map column fields from the primary report to query parameters in the sub-report. This operation is similar to the one used in drill-down.



Sub-Report Parameters

Sub-Report_1

Parameter Mapping

☑ category   Category

☑ Limit linked Sub-Report Query Execution

Sub-Report Parameter Mapping Dialog

The dialog contains a tab for each parameterized sub-report. Each parameter in the sub-report can be mapped to a different column field in the primary report.

Normally when linked sub-reports execute, the sub-report will issue its query each time it runs using the current value of the report column. However, if your report is configured such that the sub-report executes a large number of times, these repeated queries can have an impact on performance. If you check the last option in this dialog called *Limit Sub-report query execution*, EspressReport will try to merge the sub-report queries and limit the number of calls to the database when the sub-report runs. Note that this option will only take effect if the sub-report uses a database query to retrieve the data. This will not effect sub-reports that use parameterized classes or xml files as the data source.

When using linked subreports, it is recommended that you do not use the cascading parameters functionality, as this may potentially cause incorrect data to appear (depending on the cascading parameters and the order in which they are presented). The default value of the parameter will always be selected instead of the value passed by a column in the main report.

1.11.4.1. Using Linked Sub-Reports

Although linked sub-reports will run in any configuration and in any report section, it is important to be aware of the behavior of sub-reports when creating a linked presentation. As noted in Section 1.11.2 - Editing a Sub-Report, sub-reports will only run as many times as the section in which they are placed and will only run once when placed in the Table Data section.

Therefore, if you place a linked sub-report in a section that runs only once (Table Header, Report Header, Table Data, etc), the sub-report will only run once and it will use the first value from the column to which it is linked to filter the report. To generate a linked presentation where the sub-report runs for each value in the column to which it is linked, you should use a summary break layout. By setting the column as a row break and placing both the column and the sub-report in the Group Header section, you will get the desired result.

Note

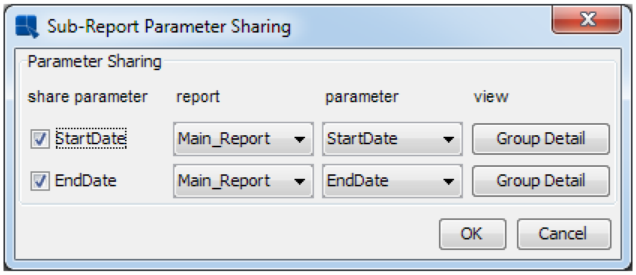
If a relationship between the column and the sub-report already exists in the database, you may want to generate a side-by-side master & details layout instead of linked sub-reports. Using the master & details layout will result in better performance. For more information about this, please see Section 1.4.4.1 - Data Mapping.

1.11.5. Parameter Linking

In addition to taking a value from a column in the main report, a sub-report can also share parameter values with the main report or with other sub-reports. In this arrangement, a user can enter values to filter the main report's data and have the same filters apply to the sub-report data.

Parameter linking is only available if parameters share the same data type. By default, if you create a sub-report that has parameters with the same name and data type as those in the main report, they will be linked. When parameter values are supplied to the main report, they are also automatically passed to the sub-report.

The parameter linking behavior can be modified by navigating to the sub-report that you want to modify and selecting *Sub-Report Parameter Sharing* from the Data menu (in the Sub-Report tab). This will bring up a dialog allowing you to modify the linking behavior for the sub-report parameters.



share parameter	report	parameter	view
☑ StartDate	Main_Report	StartDate	Group Detail
☑ EndDate	Main_Report	EndDate	Group Detail

Parameter Sharing Dialog

For each parameter defined in the sub-report, you can enable/disable the linking. You can also set which parameter in the main report or in another sub-report you want to link to. The *Group Detail* button will launch a dialog showing the current relationships for that parameter.

Once you specify the settings you want for all parameters, click the *Ok* button to apply the changes.

For more about query parameters and initialization, please see Section 1.3.2.2.2 - Parameterized Queries of this guide.

Note

Parameter linking will be overridden if you select to map a column field from the main report to the sub-report parameter as described in Section 1.11.4 - Linked Sub-Reports.

1.12. Template Security

One of the most common needs with a reporting tool is to keep the number of different report templates at a minimum. One way to facilitate template re-use is to make it easy to control certain aspects of a template for different classes of users. To that end, EspressReport allows you to set any number of different security configurations to control both column-level, row-level, and even cell-level appearance depending on which users are viewing the template.

For example, say you have a sales report that shows total sales grouped by sales region. Executives viewing the report need to see the totals as well as the contributions by each region, while individual account managers would only need to see data for their region. The base template for the report would look something like this:

Sales Report				
Total Units Sold: 29 **Total Sales:** $24,449.69				
Eastern Region				
Order #	**Quantity**	**Product**	**Unit Price**	**Total Sales**
1001	2	Chair	$325.16	$650.32
	1	Table	$1,114.18	$1,114.18
			Order Total:	**$1,764.50**
1004	3	Table	$1,114.18	$3,342.54
	2	Dresser	$1,518.60	$3,037.20
			Order Total:	**$6,379.74**
Total for Eastern Region:				**$8,144.24**
Midwestern Region				
Order #	**Quantity**	**Product**	**Unit Price**	**Total Sales**
1003	4	Chair	$325.16	$1,300.25
	2	Table	$1,114.18	$2,228.36

Sales Report				
	1	Dresser	$1,518.60	$1,518.60
			Order Total:	**$5,047.21**
Total for Midwestern Region:				**$5,047.21**
Southern Region				
Order #	**Quantity**	**Product**	**Unit Price**	**Total Sales**
1005	2	Table	$1,114.18	$2,228.35
	1	Dresser	$1,518.60	$1,518.60
			Order Total:	**$3,746.95**
1007	3	Chair	$325.16	$975.48
	1	Table	$1,114.18	$1,114.18
			Order Total:	**$2,089.66**
Total for Southern Region:				**$5,836.61**
Western Region				
Order #	**Quantity**	**Product**	**Unit Price**	**Total Sales**
1006	2	Dresser	$1,518.60	$3,037.20
	4	Chair	$325.16	$1,300.25
	1	Table	$1,114.18	$1,114.18
			Order Total:	**$5,451.63**
Total for Western Region:				**$5,451.63**

Using the security features, this one template could be used to meet the requirements of each group. In this example, assume that the report has a query parameter that filters based on region.

For executives, the view above is what they would want to see when viewing the report, so a security level called `Management` could be created that retrieves data for all regions. This is accomplished by mapping all the available values for the region parameter to the `Management` level. In this example, the values for the Eastern, Midwestern, Southern, and Western regions would be assigned to the security level. Note that in EspressReport, in order to supply more than one value to a query parameter, you need to define a multi-value parameter. For more information about this, please see Section 1.3.2.2.2.1 - Multi-Value Parameters.

For the account managers, different security levels could be created for each region. The above example has four regions, so security levels `East`, `Midwest`, `South`, and `West` would be added. Since account managers should not see the aggregate data, cells containing the summaries in the header would be hidden for the region levels using the cell-level security features discussed in Section 1.12.2 - Cell-Level Security. To make sure that each level returns data for the appropriate regions the parameter value for the Eastern region would be mapped to the `"East"` security level, the parameter value for the Midwestern region would be mapped to the `Midwest` security level, the parameter value for the Southern region would be mapped to the `South` security level, and the parameter value for the Western regions would be mapped to the `West` security level. For more information about setting security levels to apply parameter values, see Section 1.12.2.1 - Security Parameters.

When the report is run, a security level can be supplied (Typically the user's application will determine which security level to use based on a login or some other authorization method. EspressReport does not have its own user authorization mechanism). The cell settings for that security level are applied and the parameter values associated with that level are automatically supplied to the query without prompting the user. For example, here is the same report run with the `East` security settings:

Sales Report
Eastern Region

Sales Report				
Order #	**Quantity**	**Product**	**Unit Price**	**Total Sales**
1001	2	Chair	$325.16	$650.32
	1	Table	$1,114.18	$1,114.18
			Order Total:	**$1,764.50**
1004	3	Table	$1,114.18	$3,342.54
	2	Dresser	$1,518.60	$3,037.20
			Order Total:	**$6,379.74**
Total for Eastern Region:				**$8,114.24**

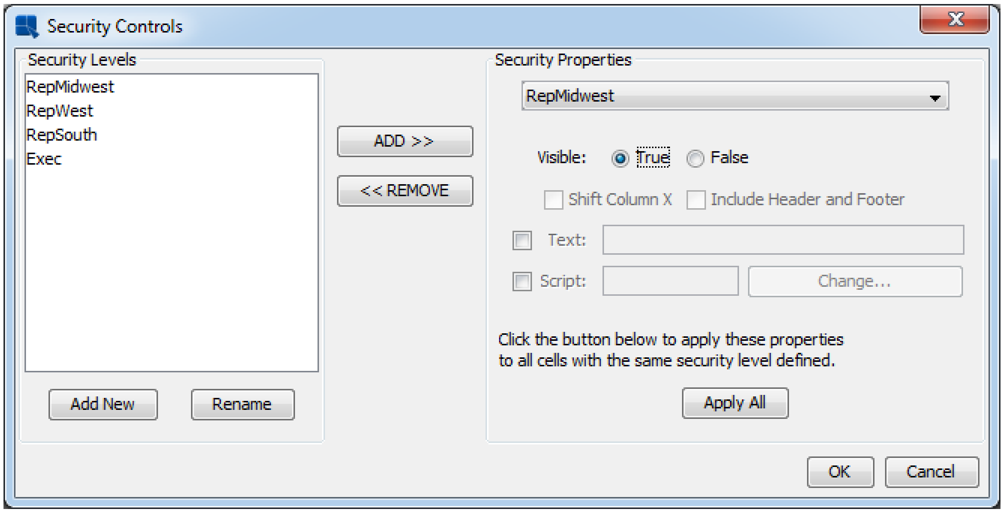
1.12.1. Security Levels

As a toolkit designed to be embedded within other application environments, EspressReport does not maintain a list of users or access privileges. Rather developers can create any number of unique security levels within a report template. For each level they can assign certain cell behavior, as well as data filtering. At run-time, a security level can be supplied to the report through the API. This solution is flexible enough to adapt to most application security models.

There is no limit to the number of different security levels that can be defined within a template. The only requirement is that each level must have a unique name. The security levels are used globally within a report and can have cell-level settings, as well as parameter values assigned to them. Security levels can be defined at the same time as security settings are applied.

1.12.2. Cell-Level Security

For any cell/visible column in the report, you can set different behaviors for different security levels. To apply security settings to a cell, either select the cell and select *Security* from the *Data* menu, or right click on the cell and select *Security* from the pop-up menu. This will bring up a dialog allowing you to specify security settings for that cell.



Security Controls

Security Levels

RepMidwest
RepWest
RepSouth
Exec

Security Properties

RepMidwest

Visible: ● True ○ False

□ Shift Column X □ Include Header and Footer

□ Text:

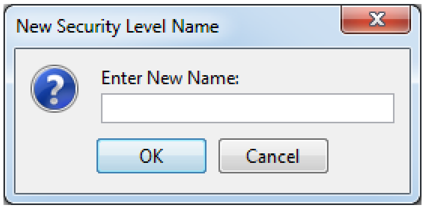
□ Script:

Click the button below to apply these properties to all cells with the same security level defined.

Cell Security Settings Dialog

The dialog contains a list of all the defined security levels for the template.

You can add a new security level by clicking the *Add New* button. This will bring up an additional dialog allowing you to specify a name for the new level.



Add Security Level Dialog

You can enter any name you want for a security level. You can also rename any defined level in the report by selecting the *Rename* button. Security levels in the report that are not associated with report cells or query parameters are automatically removed.

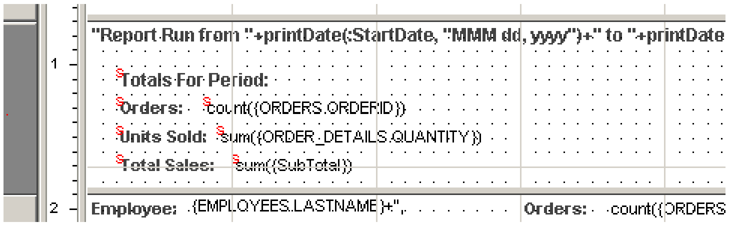
To add security settings for a report cell, select the level for which you want changes to take effect and click the *Add* button (If you create a new security level, it will be automatically added to the cell). The security level will then appear in the drop-down list on the left side of the cell security dialog. Now you can modify the cell for that security level. There are three basic controls available for cells. You can render the cell invisible for a security level. If the cell is visible, you can either provide custom text for the cell or you can apply a cell script. For more information about the cell scripts, please see Section 1.9 - Scripting. If you select to make a column (cell in Table Data section) invisible, make sure to select the column header along with the data cell so they both contain the same security setting, then you can check the *Shift Column X* option to make the other columns shift to the left when the security level is applied.

Note

When you specify to change the text of a cell, it will not work for default column headers. This is because column headers are unique elements that retrieve their value from the data source. If you wish to modify the text of a column header, you will need to convert it to a static label first. For more information about this, please see Section 1.5.7.2 - Editing Elements.

You can propagate this behavior to any cell that has controls for this security level by clicking the button on the right side. You can specify controls for as many different levels of the cell as you want.

Once you finish specifying the security settings for the cell, click the *Ok* button. You will go back to the Design window where a red *s* will appear in the upper left corner to indicate that the security settings have been applied to the cell.

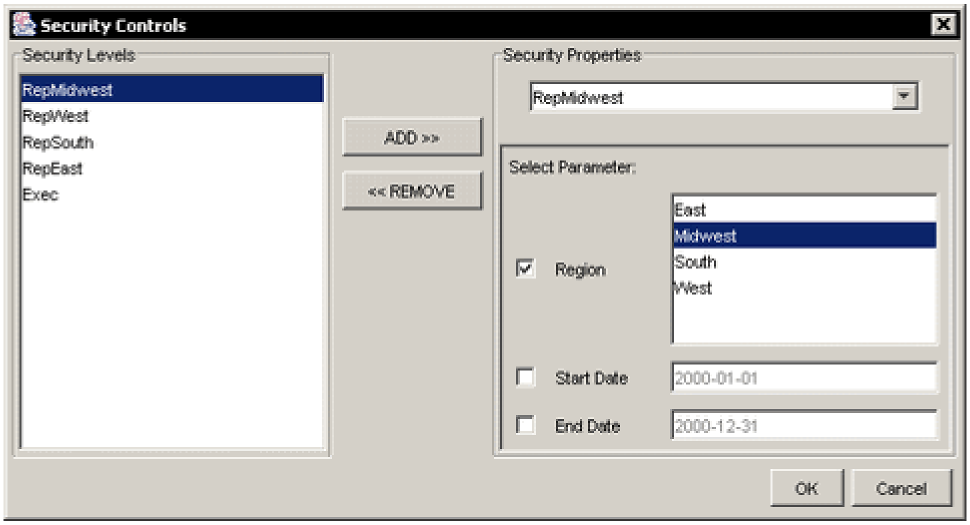

```
"Report Run from "+printDate(:StartDate, "MMM dd, yyyy")+" to "+printDate
Totals For Period:
Orders:       count({ORDERS.ORDERID})
Units Sold:   sum({ORDER_DETAILS.QUANTITY})
Total Sales:  sum({SubTotal})
Employee: {EMPLOYEES.LASTNAME}+",                 Orders:  count({ORDERS
```

Report Cells with Security Settings

1.12.2.1. Security Parameters

In addition to controlling the specific behavior of cells in a report, you can also assign parameter values to a security level. For example, using the scenario described earlier where a sales report is grouped by region, you could add a query parameter that filters by sales region, and then assign the appropriate region value to a security level for that region. When an employee logs in, the report is run with the appropriate security level and the user will only see data for their sales region as the appropriate parameter value has been supplied to the report by the security level. To use security parameters, the report must use either a parameterized database query or a parameterized class as the data source.

To add security settings for parameters, select *Secured Query Parameters* from the *Data* menu. This will bring up a dialog allowing you to select query parameter values.



Security Controls

Security Levels

- RepMidwest
- RepWest
- RepSouth
- RepEast
- Exec

Security Properties: RepMidwest

Select Parameter:

☑ Region: East, **Midwest**, South, West

☐ Start Date: 2000-01-01

☐ End Date: 2000-12-31

Security Parameters Dialog

Like the cell security dialog, the parameter security dialog contains a list of all the security levels defined in the template on the left side. You can create and edit levels in the same way as in the cell-level security dialog.

To add security settings to report parameters, select the level you want to assign parameter values to and click the *Add* button. The security level will then appear in the drop-down list on the right side of the dialog. For each selected security level, you can select which parameters you want to secure using the check box next to the parameter. For each secured parameter, you can select the value(s) that you want to be associated with this security level. Once you finish assigning parameter values, click the *Ok* button to return to the design window.

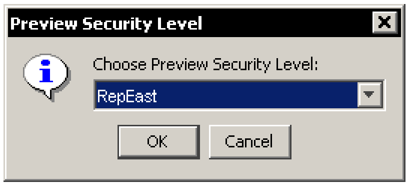
1.12.3. Security for Sub-Reports & Drill-Down Layers

Template security settings can be automatically applied to sub-reports as well as drill-down layers. In both cases, you need to edit the sub-report or drill-down layer (by clicking the sub-report tab, or navigating to the drill-down layer) to add security. In the sub-report or drill-down layer, you can add security settings just as you would for the main report. If you want to apply security to both the main report and the sub-report/drill-down layer, you will need to assign the security levels with the same name(s) as in your main report. When a report is run with a security level, the security level will be automatically applied to the sub-report(s) and/or drill-down layer(s).

For more information about drill-down and sub-reports, please see Section 1.10 - Drill-Down and Section 1.11 - Sub-Reports.

1.12.4. Viewing Secured Templates

You can preview the effects of your security settings in the Preview window of the Report Designer. To do this, select *Set Preview Security Level* from the *Data* menu. This will bring up a dialog prompting you to select the security level that you want to use to view the report.



Preview Security Level

Choose Preview Security Level: RepEast

Preview Security Level Dialog

Select the level you want and then preview the report. You will see the report with security settings applied for the level you selected. This level will remain selected until you change the settings. Selecting the blank entry at the top of the list in this dialog will set the report to run without applying any security settings.

When running secured templates using the API, you can apply the security setting you want to use. The security level can either be passed in as an argument in the `QbReport` constructor, or set using the `setSecurityLevel()`

method. For more information about running secured templates in the API, please see Section 2.3.5.1.5 - Secure Reports.

There is also a servlet example showing how to run a report with security settings included in the EspressReport installation. Source code and setup instructions can be found in `help/examples/servlet/ReportSecurity` directory under the EspressReport installation.

1.12.4.1. Secured Report Designer

When running the Report Designer through the API, you can also apply security. When the Designer is launched, you can pass in a security level using the `setSecurityLevel()` method. When the Report Designer is running in secure mode, users will only be able to preview reports at the specified level. In addition, they will be unable to modify report cells with defined security settings, or cell scripts that have been applied as security settings. For more information about running Report Designer from the API, please see Section 2.3.5.7.11 - Calling Report Designer from Report API.

1.13. Scheduling

Included with EspressReport is a simple interface that allows you to schedule reports, as well as virtually any other event on the server-side. The Scheduler runs in conjunction with EspressManager and you can set schedules through the interface, as well as manage scheduled tasks.

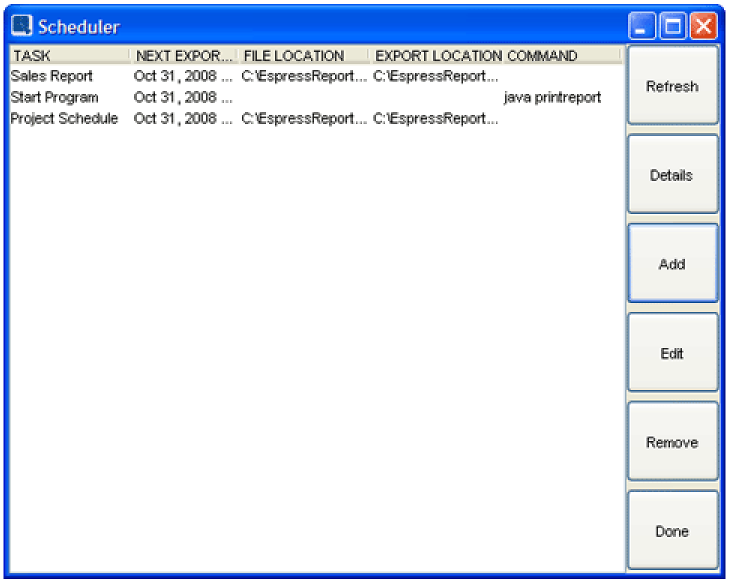
The Scheduler can be run as an application on the server or remotely through a web browser. To run the Scheduler, take the following steps:

Application: If you are running the Scheduler locally, be sure to start EspressManager first by executing the espressmanager batch file. Then start the Scheduler by executing the scheduler batch file. This will launch the application.

Browser: If you are running the Scheduler remotely, make sure EspressManager is running on the server machine, then type the URL for the Scheduler in your browser, i.e. `http://machinename/espressreport/scheduler.html`. This will bring up the Scheduler as an Java Web Start window.

1.13.1. The Scheduler Interface

The main Scheduler interface is a window that contains a list of scheduled jobs.



TASK	NEXT EXPOR...	FILE LOCATION	EXPORT LOCATION	COMMAND
Sales Report	Oct 31, 2008 ...	C:\EspressReport...	C:\EspressReport...	
Start Program	Oct 31, 2008 ...			java printreport
Project Schedule	Oct 31, 2008 ...	C:\EspressReport...	C:\EspressReport...	

Main Scheduler Interface

For each job the window displays the job name, next scheduled export time, location of the report file (if applicable), location of the exported file (if applicable), and scheduled command line (if applicable). The buttons on the right side of the window perform the following functions:

Refresh: This option will refresh the list of scheduled tasks. This is useful if there are other users adding schedules as well.

Details: This option will open a new window displaying detailed information about the currently selected scheduled task.

Add: This option allows you to set up a new schedule.

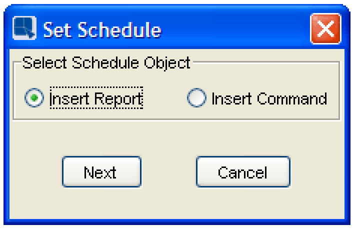
Edit: This option allows you to edit the currently selected scheduled task.

Remove: This option will delete the currently selected scheduled task.

Done: This option will exit the scheduler (This button only appears when you are running the Scheduler as an application).

1.13.2. Setting a Schedule

To create a new schedule, click the *Add* button in the main Scheduler window. This will open a dialog asking you if you want to schedule a report or a command.



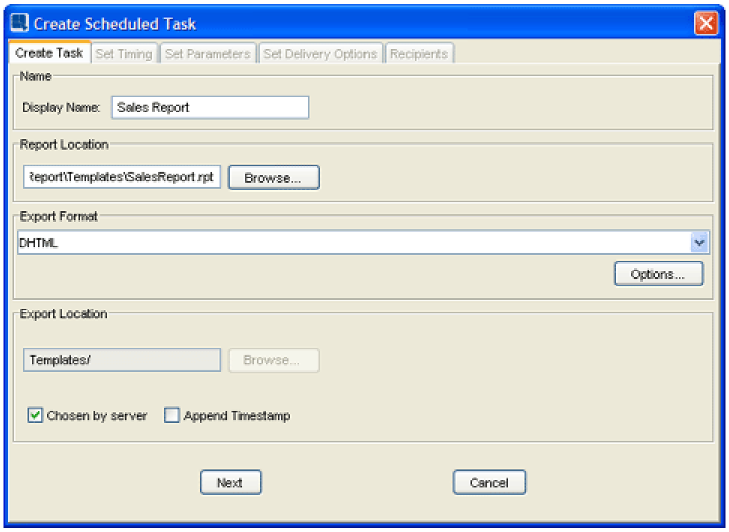
First Schedule Dialog

Report: This option allows you to select a report template (either `.pak, .rpt` or `.xml` format) to schedule. You will be able to specify the periodicity of the schedule as well as delivery options. For more information about this feature, see Section 1.13.2.1 - Scheduling a Report.

Command: The command option allows you to schedule something other than a report file. This feature allows you to write a command that will be executed in the same directory as EspressManager. Hence, you could use this feature to run a Java process that includes the Report API. For more information about this feature, see Section 1.13.2.7 - Scheduling a Command.

1.13.2.1. Scheduling a Report

To schedule a report, add a new schedule in the Scheduler and select the *Insert Report* option. The first dialog of the scheduling process will open.



Name

Display Name: Sales Report

Report Location

Report\Templates\SalesReport.rpt

Export Format

DHTML

Export Location

Templates/

☑ Chosen by server ☐ Append Timestamp

Schedule Dialog

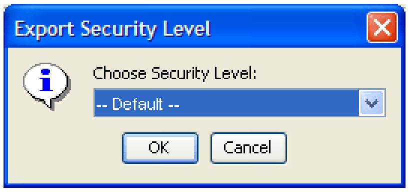
From this screen, you are first prompted to specify a display name for the scheduled event. The second section allows you to specify the template to use for the schedule and the third section allows you to specify the export format for the report. Available formats include DHTML, PDF, CSV, Excel (XLS), Excel 2007 (XLSX), text, rich text, or XML. The *Options...* button allows you to set format-specific options for the exported file. For DHTML, you can set single or multi-page exporting and you can also specify which CSS options to use. For PDF, you can set encryption and for text you can specify the delimiter. For more information about report exporting options, please see Section 1.7.2 - Exporting Reports.

For the DHTML and PDF export formats, the Options... dialog allows you to specify the drill-down servlet location. More information about the drill-down servlet can be found in Section 1.10.4 - Viewing Drill-Down Reports

The last section allows you to specify the report's export location. You can either specify a path and filename for the exported file (without extensions), or allow the server to assign a location and filename. If you enter a location and filename, the server will overwrite that file with the latest version each time the schedule is run. If you allow the server to assign location and filename, the file will be placed in the ScheduledReports directory for exported reports under the server root.

The *Append Timestamp* option allows you to control how the files are saved. Normally, when a schedule job executes, it will overwrite the last version that was generated. However, you can specify to append a timestamp with the export time to the file name each time the schedule executes. This will generate a unique file for each scheduled execution instead of overwriting the last file.

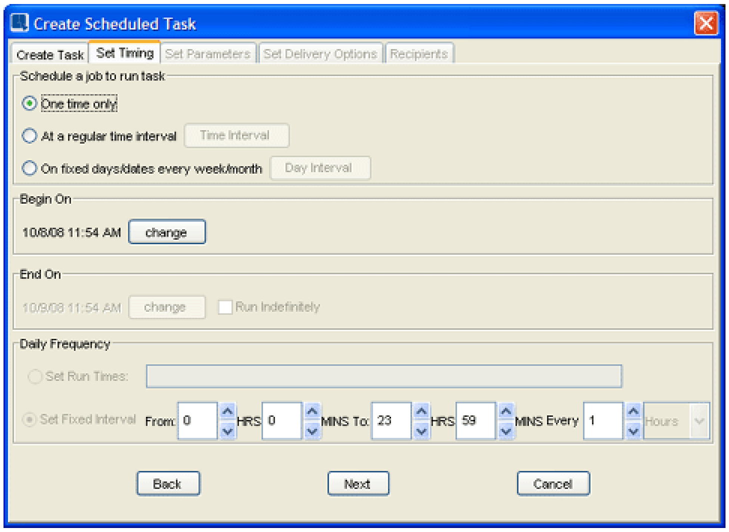
After you finish specifying image and name options, click *Next*. If the report contain security levels, a dialog will pop up asking for the security level to use for this report.



Choose Security Level: -- Default --

Set Security level

From this dialog, you can select one of the security levels defined in the template, or you can use the default setting which will apply none of the security settings to the template. The schedule will then run using the security level you have defined. Once you select the security level to use, click *OK*. You will then be prompted to specify the periodicity of the schedule.



Schedule a job to run task

● One time only

○ At a regular time interval

○ On fixed days/dates every week/month

Begin On

10/8/08 11:54 AM

End On

10/9/08 11:54 AM

□ Run Indefinitely

Daily Frequency

○ Set Run Times:

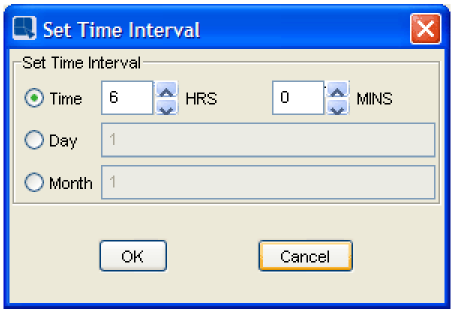
● Set Fixed Interval From: 0 HRS 0 MINS To: 23 HRS 59 MINS Every 1 Hours

Set Start & End Date Dialog

The next screen allows you to select a start and finish date for the scheduled event. The Begin On and End On sections allow you to select the start and end date for the schedule. You can also check *Run Indefinitely* instead of specifying an end date.

The top of the window allows you to select one of three options to run the chart or report: *One time only*, *At a regular time interval*, and *On fixed dates/days every week/month*.

The first option will only run the event once on the start date specified. If you select this option, there is no need to specify an end date. If you select the second option, be sure to specify a start and end date, and click the *Time Interval* button to specify the interval for the schedule. A dialog will appear allowing you to select the time interval, either in hours/minutes, days, or months. The export will begin at the start time and run at the specified intervals there after.



Set Time Interval

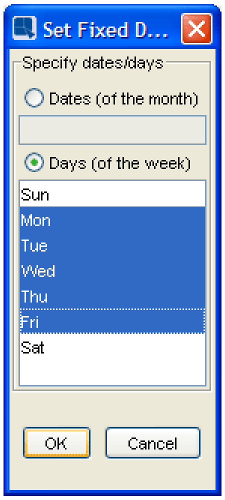
● Time 6 HRS 0 MINS

○ Day 1

○ Month 1

Select Time Interval Dialog

If you select the third option, be sure to specify a start and end date and click the *Day Interval* button to specify the interval for the schedule. A dialog will appear allowing you to enter dates of the month (dates are comma separated, i.e. 5,15), or days of the week for the event to run.



Specify dates/days

○ Dates (of the month)

● Days (of the week)

Sun
Mon
Tue
Wed
Thu
Fri
Sat

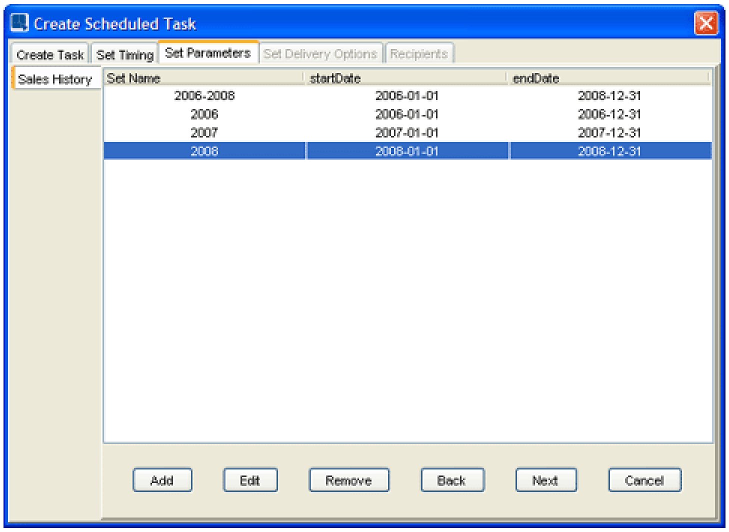
Select Fixed Days/Dates Dialog

Selecting the third option also means that you are required to specify the daily frequency. The Daily Frequency section allows you to fine tune when schedules are run during the specified days. The *Set Run Time* option allows you to specify the exact time(s) to run the export during the day. For example, using the 'Set Run Times' option, you can specify a schedule to run at irregular intervals such as 9:00AM, 5:30PM, and 12:00AM. The *Set Fixed Interval* option allows you to specify the run times based off of regular intervals. For example, you can set up a schedule that runs every 30 minutes between 9:00AM and 5:00PM using this option.

Once you finish specifying the periodicity, click *Next* to continue with the Schedule Wizard.

1.13.2.2. Scheduling Parameterized Reports

If you have selected to schedule a report that contains parameters, the next tab will appear allowing you to set the parameter values you want to use.



Set Name	startDate	endDate
2006-2008	2006-01-01	2008-12-31
2006	2006-01-01	2006-12-31
2007	2007-01-01	2007-12-31
2008	2008-01-01	2008-12-31

Set Schedule Parameters Dialog

To add a set of parameters, select the *Add* button. This will bring up the parameter prompt dialog allowing you to select the set of parameter values you want to use. Once you have selected a parameter set, your choices will appear in the dialog. You can add as many different combinations of parameter sets as you want. A separate file will be generated for each set of parameters you specify.

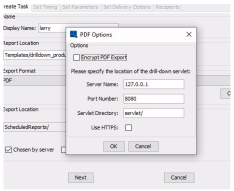
You can also specify a name for each parameter set by double clicking on the first column. The name specified here will be used in later dialogs to help you organize your recipients.

Once you have specified all the parameters you want, click the *Next* button to continue.

1.13.2.3. Scheduling Drill-Down Reports

If you're scheduling a report with drill-downs and exporting to either DHTML or PDF, the drill-down links will be active ("clicable"). Then a link will be open in your web browser containing the drill-down report. For this functionality to work, the DrillDownLinkGenerator will need to be deployed in an application server like Tomcat. More information about the drill-down function can be found in the following section: Section 1.10.4 - Viewing Drill-Down Reports

This function can be used in the scheduler too. To configure the servlet URL in the scheduler, you have to select either the DHTML or PDF export format. Then click on the Options... icon (still on the 1st Scheduler wizard step called "Create Task").



Name

Display Name: larry

Report Location

Templates/drilldown_produ

Export Format

PDF

Export Location

ScheduledReports/

☑ Chosen by server

PDF Options

Options

□ Encrypt PDF Export

Please specify the location of the drill-down servlet:

Server Name: 127.0.0.1

Port Number: 8080

Servlet Directory: servlet/

Use HTTPS: □

PDF Options Dialog

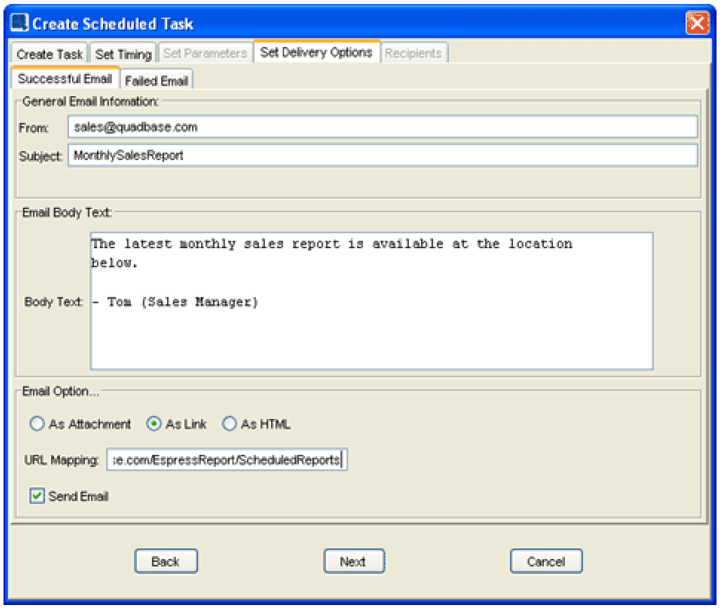
The settings and the servlet deployment are described in the following chapter: Section 1.10.4 - Viewing Drill-Down Reports

Tip

If you don't want to use this feature, you can leave these options at their default values.

1.13.2.4. Email Delivery Options

After you finish setting up the schedule, you will see a two tabbed dialog allowing you to deliver the report to users via email.



General Email Information:

From: sales@quadbase.com

Subject: MonthlySalesReport

Email Body Text:

Body Text:

The latest monthly sales report is available at the location below.

- Tom (Sales Manager)

Email Option...

○ As Attachment ● As Link ○ As HTML

URL Mapping: ;e.com/EspressReport/ScheduledReports

☑ Send Email

Email Delivery Options Dialog

If you choose to deliver the report by email, check the *Send Email* box at the bottom. The first tab contains information for the email sent when the report successfully exports. The *General Email Information* portion of this tab allows you to specify the from addresses for the email as well as the subject. The body text portion allows you to specify a message for the body of the email.

There are three ways in which a scheduled report can be sent via email. You can send the generated report as an email attachment, you can send a link to the generated report, or you can send the report as the email (this option is available for DHTML export only). If you select to send a link to the generated report, you will need to specify http path to the directory in which the report is generated. Otherwise, the link will not form correctly. For example, if you have EspressReport installed under Tomcat's web root and you elected to let the server choose the location/name of the generated files, then the actual directory location would be something like:

```
C:\Tomcat5.5\webapps\ROOT\EspressReport\ScheduledReports
```

And the corresponding URL Mapping would be:

```
http://hostname:8080/EspressReport/ScheduledReports
```

If you do not wish to include email delivery/notification with a particular job, simply un-select the option marked *Send Email*. Once you finished specifying email options, click the *Ok* button and the schedule will be completed.

Note

In order for email delivery to work correctly, you will need to have correctly configured smtp information in the EspressManager config file. For more information about this, see Section 1.2.3 - Configuration.

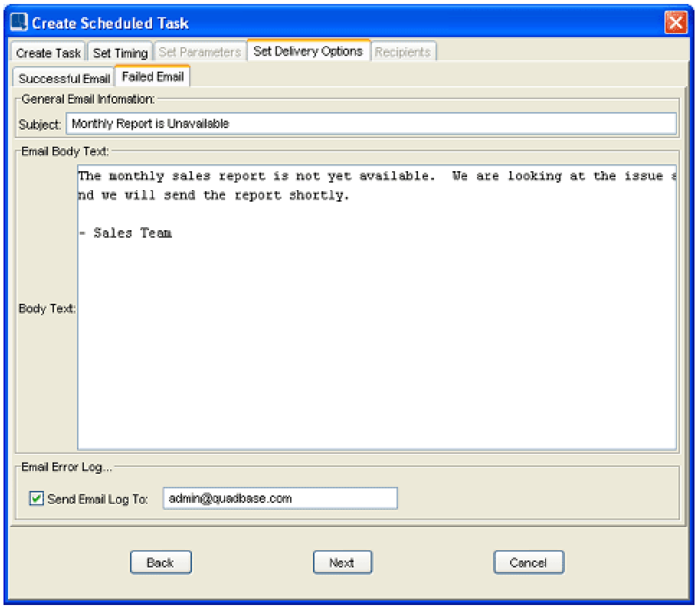
Note

In order to send email correctly, the smtp host needs to be set up in the config file in userdb directory. To set up the smtp host, open the config.txt file. Under the heading `[smtp host]`, add the name of the smtp server you want to use when emailing scheduled reports. If you are using a secure (TLS) smtp server, fill in the following values as well: `[smtp secured]`, `[smtp port]`, `[smtp username]`, `[smtp password]`. For example:

```
[port]
22071

[webroot]
c:\InetPub\wwwroot

[users]
{user-name encoded-password}
larry vJ-58-25D7f24h
guest

[smtp host]
box5109.bluehost.com

[smtp secured]
true

[smtp ssl]
true

[smtp port]
465

[smtp username]
lshen+quadbase.com
```

```
        [smtp password]
        password
```

The *Failed Email* tab allows you to send a different email (subject and body) to your recipients should there be an error running the report.



Subject: Monthly Report is Unavailable

Body Text:

```
The monthly sales report is not yet available.  We are looking at the issue a
nd we will send the report shortly.

- Sales Team
```

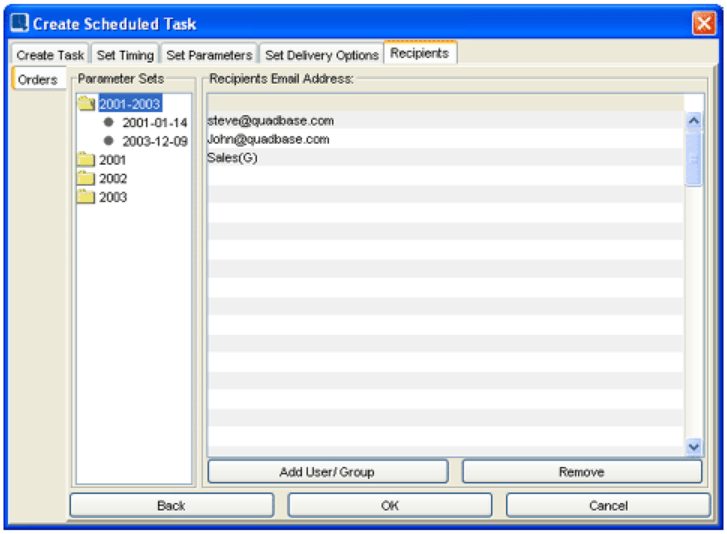
☑ Send Email Log To: admin@quadbase.com

Specifying Failed Email

This tab also allows you to send the error logs to a specific user. Typically, this would be the admin or a technical user who is able to diagnose and troubleshoot the problem.

1.13.2.5. Specifying Email Recipients

If you selected to email the scheduled file, the next tab allows you to specify the recipient list. The recipient list is set up differently depending on the type of report or chart you are scheduling. If the report or chart contains parameters, you will see the following dialog. This allows you to send different parameter sets to different users or groups.



Parameter Sets:

- 2001-2003
  - 2001-01-14
  - 2003-12-09
- 2001
- 2002
- 2003

Recipients Email Address:

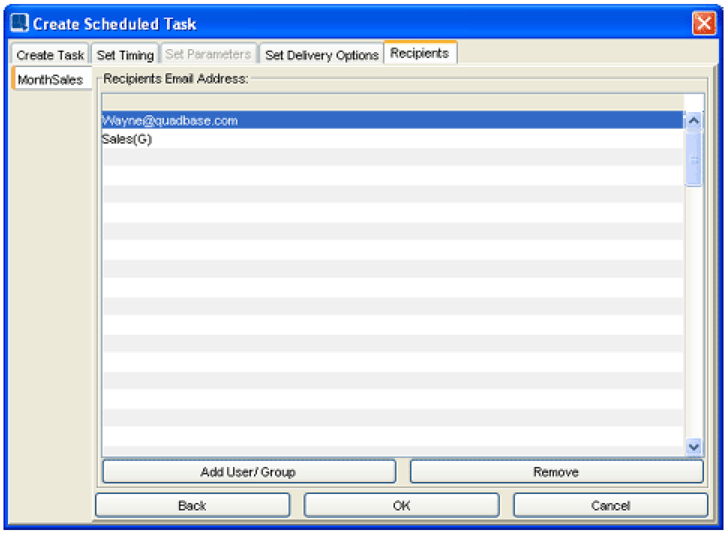
steve@quadbase.com
John@quadbase.com
Sales(G)

Recipients List for Parameterized Report

Entering recipients into the list is simple. Add regular email addresses by double clicking on any row in the recipients list and typing it in. To send different parameter sets to the same user, there are copy and paste options available when you right click on the recipients list. You may also click on *Add User/Group* to view the previously listed recipients.

If you are scheduling a grouped report (such as a summary break report) and the report is using a database as a datasource, then you will see the option for report bursting as well. Screenshot and detailed information for report bursting can be found in Section 1.13.2.6 - Report Bursting.

For all other cases, you will see the following dialog containing only the recipients list.



MonthSales

Recipients Email Address:

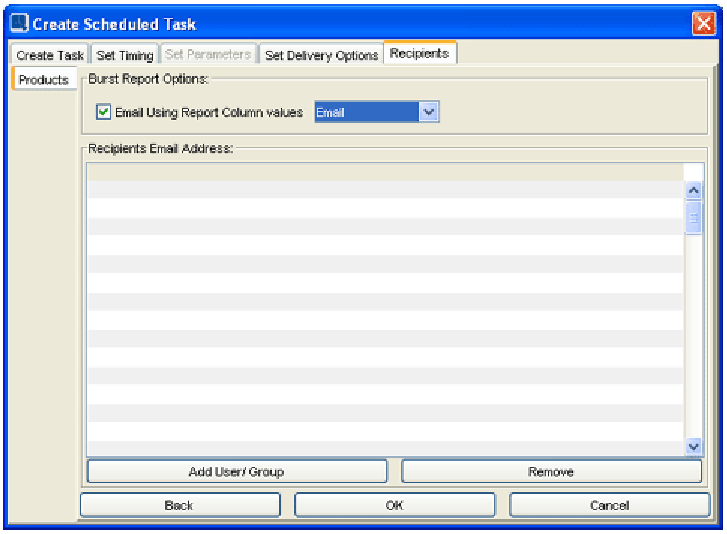
Wayne@quadbase.com
Sales(G)

Recipients List

1.13.2.6. Report Bursting

If the report being scheduled contains grouped data (like a summary break report), you can select to burst the schedule delivery. When a report is burst, the schedule job will generate a separate report for each group of data. You can then deliver each group's report to a specific email address, allowing end users to receive scheduled reports containing data relevant to them. By running large reports once and then breaking up the exports into smaller reports, this feature can offer significant performance and security improvements over running multiple smaller schedule jobs. Note that the report bursting option is only available if the datasource for the report is a database.

On the recipients tab, you will be presented with a dialog that allows you to select email delivery options for the schedule. If the report contains grouped data, the dialog will contain a bursting option at the top.



Products

Burst Report Options:

☑ Email Using Report Column values: Email

Recipients Email Address:

Recipients List for Report Bursting

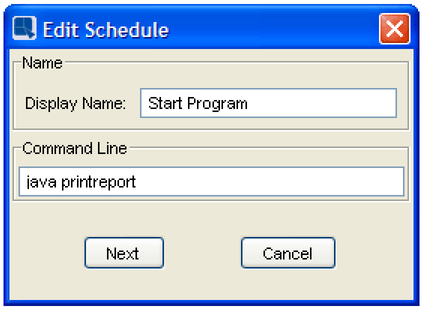
To utilize the burst report feature, the report must have a column that contains email addresses that correctly correspond to the group by (row break) column. For example, if the report is an account statement grouped by a customer

ID column, the report must also contain a column that contains the email address for each customer. Note that the email column does not need to be visible in order to use the bursting feature.

If the report meets the aforementioned requirements, you can enable report bursting by selecting the option *Email Using Report Column Values* at the top of the dialog. This will run a single query for the report, break the report up by groups, export each group, and deliver the exports to the email address listed for each group.

1.13.2.7. Scheduling a Command

In additional scheduling reports, you can also schedule a command execution. The schedule command option allows you to write a command that will be run in the same directory as EspressManager. This is a very flexible option and any command can be used. For example, you could run a Java program that contains custom code modifying and exporting a report (e.g. `Java runreport`). To create a scheduled command, create a new schedule from the Scheduler interface and select the *Insert Command* option.



Edit Schedule

Name

Display Name: Start Program

Command Line

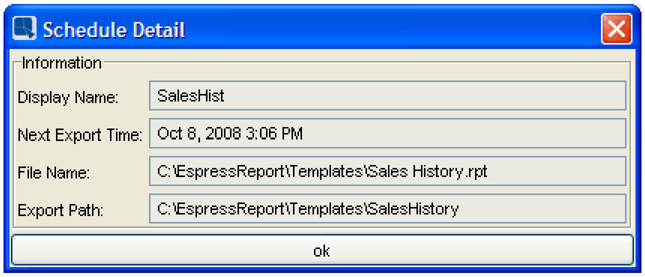
java printreport

Command Options Dialog

From this screen, you are first prompted to enter a display name for the scheduled event. The next option allows you to enter the command you want the Scheduler to execute.

1.13.2.8. Viewing and Editing Schedules

The main Scheduler window shows a list of all scheduled tasks. For more information about a particular task, you can select it and click the *Detail* button. This will bring up a new window displaying schedule details.



Schedule Detail

Information

Display Name: SalesHist

Next Export Time: Oct 8, 2008 3:06 PM

File Name: C:\EspressReport\Templates\Sales History.rpt

Export Path: C:\EspressReport\Templates\SalesHistory

Schedule Detail Dialog

To edit a schedule, you can select a scheduled task in the main window and click the *Edit* button. This will reopen the schedule dialogs, allowing you to specify different details.

In order for the schedule to run, EspressManager must be running. Schedules cannot run without EspressManager. If you shutdown EspressManager, the schedule will resume when you restart it.

Chapter 2. Programming Guide

2.1. Introduction to Report Viewer

2.1.1. Introduction

All reports created by Report Designer can be saved in various static file formats that can be viewed later. These formats include DHTML and PDF. In addition, Report Designer provides the option of saving a report as a .rpt file.

Report Viewer is an applet that enables you to view and manipulate a report dynamically through a web browser. Report Viewer reads the file (in .rpt format) as outputted by Report Designer or Report API and displays the report. The small size of the file makes it suitable to distribute the report over the web. The data is encrypted while being transferred from EspressManager to Report Viewer. Therefore, there is a degree of security to sensitive information.

Files in the .rpt format are viewed using Report Viewer. When a web page that contains a .rpt file is viewed using a browser, the most up-to-date data will be obtained automatically by Report Viewer (via EspressManager). Thus a .rpt file can be used to supply up-to-the-minute reports to users in real time.

You can navigate to different sections and pages of the report by using Report Viewer. There are built-in callback mechanisms that would allow users to click on a data element to jump to a related URL. Report Viewer is written in pure Java, thus, runs on all platforms that support Java. Report Viewer also supports scheduled refresh (whereby a report's data is updated at regular intervals specified by the Designer) and parameter serving where a report's parameters are provided at load time.

2.1.2. Launching Report Viewer

To launch the Report Viewer, you can either embed Report Viewer in an applet or simply click on the *Preview* tab in the Report Designer.

To embed the Report Viewer in an applet, simply use the following code:

```
    <applet codebase = ".." code =
"quadbase.reportdesigner.ReportViewer.Viewer.class" width = 100% height =
100% archive="ReportAPIWithChart.jar">

        <PARAM name = "filename" value = "yourReport.rpt">

    </applet>
```

The parameter value of `filename` specifies the name of the file that contains the report. The filename value can also be specified by a URL. For example, you can prefix it by `http://` for accessing a remote data file. When viewed in Report Viewer, the .rpt file will connect to the data source specified, dynamically fetch the data and generate the report.

Note that on the server side, you will need to copy the `ReportAPIWithChart.jar` file from `lib/` directory of the EspressReport Installation to a directory on your web server that is accessible by the web client.

2.1.3. The Report Viewer Parameters

Without using Report Designer, you can also pass data via parameters to Report Viewer applet. That is, you can use Report Viewer to directly view a data file, or pass the data directly in the form of lines of data in the HTML code. The following is a list of parameters:

comm_protocol: The protocol to be used in case there is a firewall or if you wish to connect to EspressManager running as a servlet.

comm_URL: The URL to connect to EspressManager, in case of a firewall or if EspressManager is running as a servlet.

servlet_context: The context to connect to EspressManager running as a servlet.

server_address: The IP address of the EspressManager connection.

server_port_number: Port number of the EspressManager connection.

ParameterServer: Port number of the parameter server which pushes the data into the client.

RefreshInterval: Specifies the interval in seconds to refresh the data (i.e. getting the data from the data source).

FileName: Specifies the name of the .rpt file.

ReportData: Specifies the report as a string.

SourceDB: Specifies the database information.

SourceFile: Specifies the data file information.

SourceData: Specifies the data information.

ReportType: Specifies the report type.

Aggregation: Specifies the Aggregation Method.

ColInfo: Sets the column mapping.

RowBreak: Specifies the column(s) to be used as Row Breaks.

ColumnBreak: Specifies the column to be used as Column Break.

ColumnBreakValue: Specifies the column to be used as the values for the Column Break columns.

PrimaryKey: Specifies the column to be used as Primary Key.

MasterKey: Specifies the column(s) to be used as Master Key.

EspressManagerUsed: If false, EspressManager is not used. The default is true.

RefreshData: If false, the report cannot be refreshed. The default is true.

ExportEnabled: If false, report cannot be exported. The default is true.

Example: With the parameter RefreshInterval you can insert the following line:

```
<PARAM name="RefreshInterval" value="60">
```

In the above example, the Report Viewer applet will fetch data from the specified data source (with the help of EspressManager) and redraw the report every 60 seconds - all transparently. It is useful for accessing databases which are updated frequently.

Note that when specifying parameters that can have multiple values (or needs to be have multiple values), the separator is a space (“ ” without the double quotes). For example:

```
<PARAM name="RowBreak" value="0 1 2">
```

In the above example, the Row Break is set on `Column 0`, `Column 1` and `Column 2` of the Column Mapping (i.e. the first three columns). The values for the parameter have a space (“ ” without the double quotes) separating them.

2.1.4. Specifying the Data Source for Report Viewer

2.1.4.1. Data Read From Database

Here is a sample HTML code that uses Report Viewer to view a report with data extracted from a database:

```
    <applet code = "quadbase.reportdesigner.ReportViewer.Viewer.class"
 width=640 height=480>
    <PARAM name="sourceDB" value="jdbc:odbc:DataSource ,
        sun.jdbc.odbc.JdbcOdbcDriver,
        username ,
        password ,
        select * from products">

    <PARAM name="ColInfo" value="0 1 3">
    <PARAM name="ReportType" value="SummaryBreak">
    <PARAM name="RowBreak" value="0">
    <PARAM name="Aggregation" value="SUM">
    </applet>
```

The arguments of `ColInfo` specify the mapping for the report. If you want more information, please see the chapter on Column Mapping in Introduction to EspressReport API.

The argument `ReportType` specifies the type of report to be displayed and it can be one of:

1. SimpleColumnar
2. CrossTab
3. MasterDetails

The argument `RowBreak` specifies the column on which the row break is applied.

The argument `Aggregation` specifies the aggregation method applied and it can be one of:

1. NONE
2. SUM
3. MAX
4. MIN
5. COUNT
6. AVG
7. FIRST
8. LAST
9. SUMSQUARE
10. VARIANCE
11. STDDEV
12. COUNTDISTINCT

2.1.4.2. Data Read From a Data File

This HTML code generates a report using data from a data file.

```
<applet code = "quadbase.reportdesigner.ReportViewer.Viewer.class" width=640
 height=480>
<PARAM name="sourceFile" value="http://.../test.dat">
<PARAM name="ColInfo" value="1 0 2 3">
<PARAM name="ReportType" value="SimpleColumnar">
</applet>
```

2.1.4.3. Data Read From Argument

It is possible to have Report Viewer read in data directly from the HTML file itself rather than from a data file or database.

```
    <applet code = "quadbase.reportdesigner.ReportViewer.Viewer.class"
 width=640 height=480>
    <PARAM name="SourceData" value="int, string, int |
        value, name, vol |
        10, 'John', 20 |
        3, 'Mary', 30 |
        8, 'Kevin', 3 |
        9, 'James', 22">

    <PARAM name="ColInfo" value="1 0 2">
    <PARAM name="ReportType" value="SimpleColumnar">
    </applet>
```

The format for the parameter `SourceData` is the same as the data file format, except that each line is ended by a vertical bar “|”.

2.1.5. Using Report Viewer

Using Report Viewer, you can browse through the report using the Pop-Up Menu. Right clicking on the report displays a Pop-Up Menu which can be used to traverse the report. The pop-up Menu displays various options:

Back: Go back to the previous report. This option is shown when you click on a hyperlink.

Section: Contains several options to navigate through the various sections.

- **First Section:** Go to the first section.
- **Previous Section:** Go to the previous section.
- **Next Section:** Go to the next section.
- **Last Section:** Go to the last section.

Page: Contains several options to navigate through the various pages.

- **First Page:** Go to the first page.
- **Previous Page:** Go to the previous page.
- **Next Page:** Go to the next page.
- **Last Page:** Go to the last page.

Output: Contains various formats to output the report to.

- **Generate HTML (Single Page):** Export the report to HTML as a single file.
- **Generate HTML (Multiple Pages):** Export the report to HTML as multiple files, one file per page.
- **Generate DHTML (Single Page):** Export the report to DHTML as a single file.
- **Generate DHTML (Paginated Single Page):** Export the report to DHTML as a single file, with breaks to simulate pages.

	Generate DHTML (Multiple Pages):	Export the report to DHTML as multiple files, one file per page.
	Generate PDF:	Export the report to PDF.
	Generate CSV:	Export the report to CSV.
	Generate EXCEL (XLS):	Export the report to Excel (xls file).
	Generate EXCEL 2007 (XLSX):	Export the report to Excel 2007 (xlsx file).
	Generate TXT:	Export the report to TXT.
	Generate RTF:	Export the report to RTF.
	Generate XML (Data + Format):	Export the complete report to XML.
	Generate XML (Pure Data):	Export the report data only to XML.
	Print:	Print the report.
Refresh:	Refresh the data from the data source.	
Go To:	Go to the specified page and section.	
Zoom:	Specify the zooming factor to be applied to the report.	
Sort by (ascend):	Sort the report based on a selected column's ascending values.	
Sort by (descend):	Sort the report based on a selected column's descending values.	
Show/Hide Report Toolbar:	This option either shows or hides the navigation toolbar at the bottom of the Viewer window.	

In addition to the pop-up menu, you can also:

1. Left click on the hyperlink associated with the cell (Note that this creates a new browser window if the associated URL is **NOT** a .rpt file. Users cannot use the Back button in the browser to return to the previous report).

2. Run the mouse over a cell containing a hyperlink to see the Hint text associated with the link.

2.1.6. Parameter Server

The Parameter Server allows EspressReport Viewer to listen for requests from the specified port using TCP/IP and update the data dynamically. The syntax is

```
<applet code = "quadbase.reportdesigner.ReportViewer.Viewer.class" width=640
 height=480>
<PARAM name="ParameterServer" value="machine:portno">
</applet>
```

For security reasons, the machine is usually the same as the web server machine. Therefore, you can leave machine empty (i.e. `value=":portno"`). For more information about the parameter server, please refer to Appendix 2.A - Parameter Server.

2.1.7. Connecting to EspressManager

2.1.7.1. EspressManager Running as Application

EspressManager runs primarily as an application. If you wish to use Report Viewer to connect to the EspressManager running as an application, you will have to add the following tags to your HTML code:

```
<PARAM name="server_address" value="Machine">
<PARAM name="server_port_number" value="PortNumber">
```

Adding the above parameters to your HTML code will make the Report Viewer connect to the EspressManager running on `Machine` and listening on `PortNumber`.

2.1.7.2. EspressManager Running as Servlet

EspressManager can be run as a servlet as well. If you wish to use the Report Viewer to connect to the EspressManager running as servlet, you will have to add the following tags to your HTML code:

```
<PARAM name="comm_protocol" value="servlet">
<PARAM name="comm_URL" value="http://Machine:PortNumber">
<PARAM name="servlet_context" value="EspressReport/servlet">
```

Adding the above parameterss to your HTML code will make the Report Viewer connect to the EspressManager at `http://Machine:PortNumber/EspressReport/servlet`.

2.1.8. Swing Version

A JFC/Swing version of the Report Viewer can also be used to refer to `SwingReportViewer.jar` instead of `ReportViewer.jar` in `EspressReport/lib` directory.

2.1.9. Exporting from Viewer

This section pertains to exporting from the Viewer in an Applet only. When using the Report Viewer API to create a Component that displays a report, the user can right click on the Component to launch a pop-up menu with various options. One of the options, (Output → Server → Generate...) is for exporting the report to different file formats. This will result in creating the exported file on the server side. However, there is an API feature that utilizes a server side Java Servlet to stream back the exported content to the client's browser. The following information describes how to use this feature:

First, you need to deploy the servlet that will stream the exported file back to the client side. This servlet, called `ViewerExportServlet`, is located at `<espress-report-install-dir>/ViewerExport/ViewerExportServlet.java`. Simply compile the servlet and deploy it in your servlet container.

Next, in the code that retrieves the report Component, two lines of code must be added:

```
Viewer viewer = new quadbase.reportdesigner.ReportViewer.Viewer();
Component comp = viewer.getComponent(report);
viewer.setExportServlet("http://host:port/servlet/ViewerExportServlet");
viewer.setDynamicExport(true, "host", port, "servlet/");
```

The argument for the setExportServlet method is the URL location of your deployed `ViewerExportServlet`. Please see the javadoc Specification for more details about the arguments of these two methods.

Then, when the user views the report Component, there will be new options under Output → Client → Generate.... All these options will result in a pop-up browser window that contains the streamed content of the report.

2.2. Introduction to Page Viewer

2.2.1. Introduction

The Report Viewer component offers users the capability of viewing and manipulating a report through a Web browser. The viewer loads the report template on the client and shows it to the user with the latest data. Report Viewer, however, is not the ideal deployment solution when the report contains a large amount of data. In these situations, loading the entire report at the client would cause the client to quickly run out of memory.

For situations like these, EspressReport provides another viewer called Page Viewer. Instead of loading the entire report at the client, Page Viewer employs a page serving technology that allows the client to load only one page of the report at a time. This allows the report to load faster and conserves the client memory.

Page Viewer works by exporting the report into a series of page (.page) files that contain the drawing information for a set of pages in the report. The .page files are then loaded in the viewer at the client. Users running with

EspressManager can directly pass in a report template as part of Page Viewer code. The report will be exported on the server with the latest data and the client will load the .page files. Users running without EspressManager can export the report in `view` format either from the API or Report Designer. These exported files can be loaded directly in Page Viewer without the EspressManager.

2.2.2. Launching Page Viewer

To launch the Page Viewer, you can embed it in an applet using a .rpt file (EspressManager) or an exported view (.view) file (with or without EspressManager). Page Viewer can also be used to load a report directly from the Preview window in Report Designer.

To embed the Page Viewer in an applet, simply use the following code:

```
   <applet codebase = ".." code
= "quadbase.reportdesigner.PageViewer.Viewer.class" width = 100% height =
100% archive="PageViewer.jar">

       <PARAM name = "filename" value = "yourReport.rpt">

   </applet>
```

The parameter value of `filename` specifies the name of the file that contains the report. If you are running Page Viewer with EspressManager on the back-end, you can specify the name of a report template. You can also use an exported view file for this parameter. When using a view file with Page Viewer, you do not have to have EspressManager running. The filename value can also be specified by a URL. For example, you can prefix it by `http://` for accessing a remote data file.

You may also use Page Viewer in an application. To do so, simply use the following code to get a `java.awt.Component Object`.

```
quadbase.reportdesigner.PageViewer.Viewer.getComponent(java.awt.Frame frame,
 java.lang.String fileName, long bufferTimeInSec)
```

In addition to passing in the template file (or view file), you can also pass in a `QbReport` object. Note that instead of using the Viewer class, you would be using the ViewerAPI class.

```
quadbase.reportdesigner.PageViewer.ViewerAPI.getComponent(java.awt.Frame
 frame, QbReport report, long bufferTimeInSec, java.lang.String
 securityLevel);
```

The above methods are the simpliest of all methods for getting Page Viewer as a Component. For more information about the parameters for this constructor or information on other methods related to Page Viewer, please consult the EspressReport API Specification.

Note

On the server side, you will need to copy the `PageViewer.jar` file from `lib/` directory of the EspressReport Installation to a directory of your web server that is accessible by the web client.

2.2.2.1. Launching from Report Designer

Page Viewer can also be invoked from the Report Designer. When previewing a report with large amount of data, the report can be loaded into a Page Viewer window. This allows users to preview and view the entire report without loading all the data into memory. For more information about this feature, please see Section 1.6.2.4 - Using Page Viewer.

2.2.3. The Page Viewer Parameters

Without using Report Designer, you can also pass data via parameters to Page Viewer applet. That is, you can use Page Viewer to directly view a data file, or pass the data directly in the form of lines of data in the HTML code. The following is a list of parameters:

Page Parameters

comm_protocol: The protocol to be used, in case there is a firewall or if you wish to connect to EspressManager running as a servlet.

comm_URL: The URL to connect to EspressManager, in case of a firewall or if EspressManager is running as a servlet.

servlet_context: The context to connect to EspressManager running as a servlet.

server_address: The IP address of the EspressManager connection.

server_port_number: Port number of the EspressManager connection.

server_hosts: Any other name(s) that the EspressManager host may can be specified by using a comma as a separator.

FileName: Specifies the name of the .rpt file.

EspressManagerUsed: If false, the EspressManager will not be used. The default is true. If the Page Viewer is being used to run a report template, the EspressManager must be used. If a view file is used, then the EspressManager does not have to be running.

BufferTime: The time, in seconds, that controls how often to get new pages from the EspressManager. It looks at the local version of the .page file (if available). If the time since it was last updated exceeds the bufferTimeInSec parameter, the new version of the page is requested again from the EspressManager. Otherwise, it uses the local version of the .page file. If the local version of the .page file does not exist, it will get the page from the EspressManager.

DataHintOffsetX: When the user moves the mouse over a data point (a bar on a bar graph or a point on a line graph) on a chart, a hint box that shows the details of the data point is shown. This parameter sets the horizontal offset position of the hint box relative to the data point.

DataHintOffsetY: When the user moves the mouse over a data point (a bar on a bar graph or a point on a line graph) on a chart, a hint box that shows the details of the data point is shown. This parameter sets the vertical offset position of the hint box relative to the data point.

DataHintBgColor: When the user moves the mouse over a data point (a bar on a bar graph or a point on a line graph) on a chart, a hint box that shows the details of the data point is shown. This parameter sets the background color of the hint box.

DataHintFontColor: When the user moves the mouse over a data point (a bar on a bar graph or a point on a line graph) on a chart, a hint box that shows the details of the data point is shown. This parameter sets the font color of the hint box.

DataHintFont: When the user moves the mouse over a data point (a bar on a bar graph or a point on a line graph) on a chart, a hint box that shows the details of the data point is shown. This parameter sets the font type of the hint box.

Printing: Valid values: "true" | "false"

Sets whether the print option is shown in the pop-up menu.

RefreshData: Valid values: "true" | "false"

If false, report cannot be refreshed.The default is true.

PopupMenu: Valid values: "true" | "false"

Sets whether the pop-up menu is shown.

EnableExport: Valid values: "true" | "false"

If false, report cannot be exported.The default is true.

ShowDataHint: Valid values: "true" | "false"

Sets whether to show a data hint box when mouse over a data point in a chart.

ShowLinkHint: Valid values: "true" | "false"

Sets whether to show a data hint box when mouse over a hyperlink.

Example: With the parameter `BufferTime`, you can insert the following line:

```
<PARAM name="DataHintBgColor" value="ff00aa">
```

2.2.4. Using Page Viewer

Using the Page Viewer, you can browse through the pages of the report using the pop-up menu. Right clicking on the report displays a pop-up menu that can be used to traverse the report. The pop-up menu shows various options:

Back: Go back to the previous report. This option is shown when you click on a hyperlink.

Section: Contains various options to navigate through the various sections.

First Section: Go to the first section.

Previous Section: Go to the previous section.

Next Section: Go to the next section.

Last Section: Go to the last section.

Page: Contains various options to navigate through the various pages.

First Page: Go to the first page.

Previous Page: Go to the previous page.

Next Page: Go to the next page.

Last Page: Go to the last page.

Output: Contains various formats to output the report to.

Generate HTML (Single Page): Export the report to HTML as a single file.

Generate HTML (Multiple Pages): Export the report to HTML as multiple files, one file per page.

Generate DHTML (Single Page): Export the report to DHTML as a single file.

Generate DHTML (Paginated Single Page): Export the report to DHTML as a single file, with breaks to simulate pages.

Generate DHTML (Multiple Pages): Export the report to DHTML as multiple files, one file per page.

Generate PDF: Export the report to PDF.

Generate CSV: Export the report to CSV.

Generate EXCEL (XLS): Export the report to Excel (xls file).

Generate EXCEL 2007 (XLSX): Export the report to Excel 2007 (xlsx file).

Generate TXT: Export the report to TXT.

Generate RTF: Export the report to RTF.

Generate XML (Data + Format): Export the complete report to XML.

Generate XML (Pure Data): Export the report data only to XML.

Print: Print the report.

Refresh: Refresh the data from the data source.

Go To: Go to the specified page and section.

Zoom: Specify the zooming factor to be applied to the report.

Sort by (ascend): Sort the report based on a selected column's ascending values.

Sort by (descend): Sort the report based on a selected column's descending values.

Show/Hide Report Toolbar: This either shows or hides the Navigation toolbar at the bottom of the viewer window.

In addition to the pop-up menu, you can also:

- Left click to jump to the hyperlink associated with the cell (note that this creates a new browser window if the associated URL is **NOT** a .rpt file. Users cannot use the Back button on the browser to return to the previous report).

- Run the mouse over a cell containing a hyperlink to see the Hint text associated with the link.

2.2.5. Buffer Time

A problem working with the view (.view) and page (.page) files associated with Page Viewer is that while saving memory and boosting performance by caching the report page by page on a file system, the Page Viewer's files data might become obsolete in time. In other words, they might not reflect the latest report data. The solution to this is to have a mechanism for updating the Page Viewer files during certain time intervals.

Page Viewer provides the mechanism to adjust the time elapsed until a new view file needs to be fetched from the server. This time interval, also known as the `Buffer Time`, can be adjusted by setting the parameter `BufferTime` in the Page Viewer applet. Simply pass in the buffer time in seconds as the value to the parameter `BufferTime`. The Page Viewer will then fetch new pages whenever the current version of the VIEW file becomes older than the specified buffer time.

Also, the Page Viewer has a built-in mechanism that automatically eliminates the old view and page files when they have not been updated for three or more days.

2.2.6. Connecting to EspressManager

2.2.6.1. EspressManager Running as Application

EspressManager runs primarily as an application. If you wish to use the Page Viewer to connect to the EspressManager running an application, you will have to add the following tags to your HTML code:

```
<PARAM name="server_address" value="Machine">
<PARAM name="server_port_number" value="PortNumber">
```

Adding the above parameters to your HTML code will make the Page Viewer connect to the EspressManager running on `Machine` and listening on `PortNumber`.

2.2.6.2. EspressManager Running as Servlet

EspressManager can be run as a servlet as well. If you wish to use the Page Viewer to connect to the EspressManager running as servlet, you will have to add the following tags to your HTML code:

```
<PARAM name="comm_protocol" value="servlet">
<PARAM name="comm_URL" value="http://Machine:PortNumber">
<PARAM name="servlet_context" value="EspressReport/servlet">
```

Adding the above params to your HTML code will make the Page Viewer connect to the EspressManager at `http://Machine:PortNumber/EspressReport/servlet`.

2.2.7. Swing Version

A JFC/Swing version of the Page Viewer can also be used to refer to `SwingPageViewer.jar` instead of `PageViewer.jar` in `EspressReport/lib` directory.

2.2.8. Exporting from Viewer

This section pertains to exporting from the Page Viewer in an applet only. When using the Page Viewer API to create a Component that displays a report, the user can right click on the Component to launch a pop-up menu with various options. One of the options (Output → Server → Generate) is for exporting the report to different file formats. This will result in creating the exported file on the server side. However, there is an API feature that utilizes a server side Java servlet to stream back the exported content to the client's browser. The following describes how to use this feature:

First, you need to deploy the servlet that will stream the exported file back to the client side. This servlet, called ViewerExportServlet, is located in `<espress-report-install-dir>/ViewerExport/ViewerExportServlet.java`. Simply compile the servlet and deploy it in your Servlet container.

Next, in the code that retrieves the report Component, four lines of code need to be added:

```
Viewer viewer = new quadbase.reportdesigner.PageViewer.Viewer();
Component comp = viewer.getComponent((Applet)null, reportTemplate, 0);
viewer.setExportServlet("http://host:port/servlet/ViewerExportServlet");
viewer.setDynamicExport(true, "host", port, "servlet/");
```

where reportTemplate is a string containing the name and path of the report template file. You can also pass in a QbReport object using the following code:

```
ViewerAPI viewer = new quadbase.reportdesigner.PageViewer.ViewerAPI();
Component comp = viewer.getComponent((Applet)null, report, 0, null);
viewer.setExportServlet("http://host:port/servlet/ViewerExportServlet");
viewer.setDynamicExport(true, "host", port, "servlet/");
```

where report is a `QbReport` object.

The argument for the `setExportServlet` method is the URL location of your deployed `ViewerExportServlet`. Please see the javadoc specification for more details about the arguments of these two methods.

Then, when the user views the report Component, there will be new options under Output → Client → Generate. All of these options will result in a pop-up browser window that contains the streamed content of the report.

2.3. EspressReport Report API

2.3.1. Introduction and Setup

EspressReport provides an easy-to-use application programming interface (API) that enables users to create and customize reports within their own applications (and applets) on either server side or client side. It is written in

100% pure Java and thus can be run on any platform with little or no changes. Any and every part of the report is customizable using the API. You can use as little as a single line of code to generate a report.

The main class, `QbReport`, extends `java.awt.Component`. Associated with this component is a set of auxiliary classes consisting of 5 packages: `quadbase.reportdesigner.ReportAPI`, `quadbase.reportdesigner.ReportElements`, `quadbase.reportdesigner.ReportViewer`, (and its swing counterpart `quadbase.reportdesigner.ReportViewer.swing`), `quadbase.reportdesigner.lang` and `quadbase.reportdesigner.util`. The remainder of this document explains the constituents of the API and their usage. Please note that the complete API documentation is located at help/apidocs/index.html.

To use the API, add `ReportAPIWithChart.jar` and `qblicense.jar` (located in `EspressReport/lib` directory) to your `CLASSPATH`. Please note that if you are using a SOAP data source or XML file to read or write data, you will also need to add `xercesImpl.jar` and `xml-apis.jar` (also located in the same directory) in that order to the `CLASSPATH`. If you are exporting the charts or images to PNG format, you will also need to include `ExportLib.jar` to your `CLASSPATH`. To export the report to Excel file (i.e. .xls file), you must include `poi.jar` into your `CLASSPATH`. If you want to use parameterized database queries as your data sources, add `jsqlparser.jar` to your CLASSPATH. If you want to export to MS Excel 2007 format (OOXML - extension .xlsx), you must also include these files to your `CLASSPATH`: `poi-ooxml.jar, poi-ooxml-schemas.jar, commons-codec.jar, commons-collections.jar, commons-compress.jar, commons-io.jar, commons-math3.jar, log4j-api.jar, SparseBitSet.jar, xmlbeans.jar`. If your application is on Windows or Solaris machine, you will have to add the following environment variable (depending on the platform):

```
For Windows) set PATH=%PATH%;&lt;path to EspressReport root directory&gt;
\lib

For Solaris) export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:&lt;path to
 EspressReport root directory&gt;/lib
```

Please note that third-party jar files may also be required, depending on what your application does. For example, if you create a JDBC connection to database, you will need to include the JDBC jar files to the CLASSPATH as well.

2.3.2. Recommended Approach for using Report API

EspressReport provides API through which reports can be generated programmatically from scratch. However, this generally involves a significant amount of code. We recommend using Report Designer and creating a Report File (.rpt file) first, which can function as a template. This template can then be passed in the `QbReport` constructor and the template's look and feel can be applied to the newly created report object. Thus, at the time of the report generation, a major portion of the look and feel would have been already set. You can also open a Report File (either a .RPT or a .PAK file) using the API. Using either approach, you can then write code to modify, add, or remove the properties. This approach will save you coding time and will also improve performance.

You have the option of connecting to EspressManager when using EspressReport Report API. While a connection is required when using the Report Designer, you do not need to connect to EspressManager when running any application that utilizes EspressReport Report API. We generally recommend that you do not connect to EspressManager and thereby avoid another layer in your architecture. For more details, please refer to the next section.

All examples and code given in the manual follow the above two recommendations: a template based approach and no connection to EspressManager. Unless otherwise noted, all code examples will use a template (templates can be downloaded from corresponding chapters) and will not connect to EspressManager.

Also note that if you have applets that use EspressReport Report API, the browser must have at least 1.5 JVM plugin.

2.3.3. Interaction with EspressManager

Before we go into details of how to create and use reports, let's explore the options of using EspressManager in an application. EspressReport is generally used in conjunction with EspressManager. The report component connects to EspressManager in order to read and write files to access databases and perform data pre-processing required for certain advanced features (such as aggregation). However, the report component can also be used in a stand-alone mode, in which it performs file I/O and database access directly, without the use of EspressManager.

For instance, the applet or application may be running on machine `Client` and may require data from a database on machine `DbMachine`. However, machine `DbMachine` may be behind a firewall or a direct connection may not be allowed to machine DbMachine from machine Client due to security restrictions. EspressManager can be run on machine Server and the applet/application can connect to EspressManager on machine Server. JDBC can then be used to connect to machine DbMachine from machine Server and get the data. The data is then delivered to machine Client and the report generated. This is useful when you want to keep the data secure and non-accessible from your client machines and make all connections come through a server machine (a machine running EspressManager). You can also utilize this option to keep a log of all the clients accessing the data through EspressReport (you can have a log file created when starting EspressManager. The log file is called espressmanager.log). Note that this functionality comes at a cost. You will face a slight performance decrease because your code is connecting to the data through EspressManager (i.e. another layer has been added).

By default, a report component requires the presence of EspressManager. To change the mode, use the QbReport class static method at the beginning of your applet/application (before any QbReport objects are created):

```
static public void setEspressManagerUsed(boolean b)
```

Both applications and applets can be run with or without accessing EspressManager. Communication is done using the http protocol. The location of the server is determined by an IP address and port number passed in the API code. The instructions on how to connect to the EspressManager are as follows:

2.3.4. Connecting to EspressManager

2.3.4.1. EspressManager Running as Application

EspressManager primarily runs as an application. If you wish to use the ReportAPI to connect to EspressManager running as an application, you can use API methods to specify the IP address/machine name where EspressManager is located and the port number that EspressManager is listening on.

You can use the following two API methods to set the connection information:

```
static void setServerAddress(java.lang.String address);
static void setServerPortNumber(int port);
```

For example, the following lines of code:

```
QbReport.setServerAddress("Machine");
QbReport.setServerPortNumber(PortNumber);
```

will connect to EspressManager running on Machine and listening on PortNumber.

Please note that if EspressManager connection information is not specified, the code will attempt to connect to EspressManager on the local machine and listening to the default port number (22071).

Please note that these methods exist in QbReport, QbChart, QbReportDesigner and QbScheduler classes.

2.3.4.2. EspressManager Running as Servlet

EspressManager can also be run as a servlet. If you wish to use Report API to connect to EspressManager running as a servlet, you will have to use the following methods:

```
public static void useServlet(boolean b);
public static void setServletRunner(String comm_URL);
public static void setServletContext(String context);
```

For example, the following lines of code:

```
QbReport.useServlet(true);
QbReport.setServletRunner("http://Machine:PortNumber");
QbReport.setServletContext("EspressReport/servlet");
```

will connect to EspressManager running at http://Machine:PortNumber/EspressReport/servlet.

Please note that these methods exist in `QbReport`, `QbChart`, `QbReportDesigner` and `QbScheduler` classes.

2.3.5. Using the API

The following section details how to utilize the API. Again, the API examples and code is designed using the above recommendation (template based and no EspressManager).

Unless otherwise noted, all examples use the Woodview HSQL database, which is located in `<ERInstall>/help/examples/DataSources/database` directory. In order to run the examples, you will need to add database HSQL JDBC driver (`hsqldb.jar`) to your classpath. The driver is located in `<ERInstall>/lib` directory.

Also, all the API examples will show the core code in the manual. To compile the examples, make sure the `CLASSPATH` includes `ReportAPIWithChart.jar` and `qblicense.jar`.

For more information on the API methods, please refer to the API documentation [https://data.quadbase.com/Docs71/er/help/apidocs/index.html].

2.3.5.1. Loading a Report

Reports can be saved to a file using the RPT format (a proprietary format) or XML format. A RPT/XML file stores all report information except actual data (although an option exists to save the entire data within a RPT file). This format can be used to reconstruct a report object. The data is automatically reloaded from the original data source each time the RPT/XML file is opened.

It is important to note that the RPT/XML file does **NOT** contain the data by default. It contains the specified data source along with the report template information (i.e. the look and feel of the report). Therefore, the data for the report is obtained by querying the data source when loading a RPT/XML file. This format can be obtained by using the Report Designer or the export method provided in the `QbReport` class. You can also choose to include the complete data in the RPT file (along with the data source information).

The following example, which can run as an applet or application, reads a RPT file and reconstructs a report:

```
Component doOpeningTemplate(Object parent) {

 // Do not use EspressManager
 QbReport.setEspressManagerUsed(false);

 // Open the template
 QbReport report = new QbReport(parent, // container
   "OpeningTemplate.rpt"); // template

 // Show report in Viewer
 return (new Viewer().getComponent(report));
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/OpeningTemplate.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/OpeningTemplate.pdf]

The constructor in this example is:

```
public QbReport(Object parent, String file);
```

where parent is the container and file is a RPT/XML file name.

Note

File names may be specified either as URL strings or using relative/absolute paths.

Path names are interpreted as being relative to the current directory of EspressManager or to the current application if EspressManager is not used.

2.3.5.1.1. Sub-Reports, Charts, and Drill-Down Reports

A report template may contain charts, sub-reports and/or drill-down reports. These ancillary templates are saved separately from the main report template. Chart templates are saved in the `chart` directory, sub-report templates in the `SubReport` directory, and drill-down templates in the `DrillDown` directory. A relative URL (relative to the EspressReport installation directory) and the template name is then specified and put in the main report template.

The following example, which can run as an applet or application, reads a RPT file and sets the Chart Path:

```
Component doSetChartPath(Object parent)  {

 // Do not use EspressManager
 QbReport.setEspressManagerUsed(false);

 // Open the template
 QbReport report = new QbReport (parent,     // container
         "setChartPath.rpt"); // template

 // Set Chart Path
 report.setChartPath(".");

 // Show report in Viewer
 return (new Viewer().getComponent(report));
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SetChartPath.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/SetChartPath.pdf]

The above code sets the location of the chart template file, thus avoiding the need to have a Chart subdirectory under the working directory of your class file.

The following methods work similarly:

```
QbReport.setDrillDownPath(String directory);
QbReport.setSubReportPath(String directory);
QbReport.setImagePath(String directory);
```

2.3.5.1.2. PAK File

You can also pack your report in PAK format and deploy it as a PAK file. This alternative simplifies the deployment procedures by including all the chart, sub-report, drill-down, and image files that are associated with your main report into one .pak file. Your code can uses the .pak file name (instead of the .rpt/.xml) and creates the report. With the .pak file approach, you do not need to specify the directories where the chart, sub-report, and/or drill-down templates are located.

2.3.5.1.3. Backup Data

By default, report templates are always saved with two rows of backup data. This is to ensure that any template can be used to create the `QbReport` object, even if the data source is not present. When you create a `QbReport` object based on an existing template, it looks for the data source specified in the template. If the data source is not found, the backup data is used. However, you can force the API to use the backup data instead of searching for the data source using the following constructor:

```
public QbReport(Object parent, String file, boolean isEnterpriseServer,
                boolean optimizeMemory, boolean multiPageExp,
                boolean useBackupData);
```

To force the API to use the backup data, only the last argument in the above constructor has to be set to true. The values of the other boolean arguments do not matter.

2.3.5.1.4. Parameterized Reports

Reports can also contain parameters to either limit the data in some form or to provide additional information. Typically, query (or IN) parameters are used by the data source to limit the data while parameterized formula are used to include more data within the report.

Query parameters can be both single-value and multi-value parameter types while formula parameters are single-value only.

When a parameterized template is opened using following constructor:

```
QbReport(Object parent, String templateName);
```

a dialog box appears, asking for the value(s) of the parameter(s). This dialog box is the same as the one that appears in Designer when the report is previewed.

2.3.5.1.4.1. Object Array

A parameterized report can also be opened without the dialog box prompting for any value(s). This can be done by passing in two object arrays, one for the query parameters and another for the formula parameters. Each element in the array represents the value for that particular parameter.

The order of the array must match the order in which the parameters were created in Designer. For correct results, the data type of the value must also match the data type of the parameter.

Query parameters can also be multi-value parameter types. For multi-value parameters, a vector is passed to the object array. The vector contains all the values for the multi-value parameter.

The following example, which can run as an applet or application, passes in the parameter values when opening a template:

```
Component doObjectArray(Object parent) {

 // Do not use ERES Server
 QbReport.setEspressManagerUsed(false);

 // Object array for Query Parameters
 Vector vec = new Vector();
 vec.add("CA");
 vec.add("NY");

 GregorianCalendar beginDate = new GregorianCalendar(2001, 0, 4);
 GregorianCalendar endDate = new GregorianCalendar(2003, 1, 12);

 long beginLong = beginDate.getTimeInMillis();
 long endLong = endDate.getTimeInMillis();

 Date beginDateTime = new Date(beginLong);
 Date endDateTime = new Date(endLong);

 Object queryParams[] = new Object[3];
 queryParams[0] = vec;
 queryParams[1] = beginDateTime;
 queryParams[2] = endDateTime;

 // Object array for Formula Parameter
 Object formulaParams[] = new Object[1];
 formulaParams[0] = "Sarat";

 // Open the template
 QbReport report = new QbReport(parent, // container
   "ObjectArray.rpt", // template
   queryParams, // Query Parameters
```

```
    formulaParams); // Formula Parameters

  // Show report in Viewer
  return (new Viewer().getComponent(report));
 }
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ObjectArrayERES.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/ObjectArray.pdf]

2.3.5.1.4.2. getAllParameters method

In addition to the above, you can also pass in the parameters using the `getAllParameters` method. The `getAllParameters` method returns a list of all parameters in the report (this includes any parameters from the sub-report that are not shared). Each parameter is obtained and the value is then set.

The following example, which can be run as an applet or application, take the same report as above and passes in the parameters using the `getAllParameters` method:

```
Component doGetAllParameters(Object parent) {

  // Do not use EspressManager
  QbReport.setEspressManagerUsed(false);

  // Object array for Query Parameters
  Vector vec = new Vector();
  vec.addElement("CA");
  vec.addElement("NY");

  GregorianCalendar beginDate = new GregorianCalendar(2001, 0, 4);
  GregorianCalendar endDate = new GregorianCalendar(2003, 1, 12);

  long beginLong = beginDate.getTimeInMillis();
  long endLong = endDate.getTimeInMillis();

  Date beginDateTime = new Date(beginLong);
  Date endDateTime = new Date(endLong);

  Object queryParams[] = new Object[3];
  queryParams[0] = vec;
  queryParams[1] = beginDateTime;
  queryParams[2] = endDateTime;

  // Object array for Formula Parameter
  Object formulaParams[] = new Object[1];
  formulaParams[0] = "Sarat";

  // Open the template with backup data
  QbReport report = new QbReport(parent, // container
    "ObjectArray.rpt", // template
    false, false, false, true);

  // Pass in parameters using getAllParameters
  report.getAllParameters().get(0).setValues((Vector) queryParams[0]);
  report.getAllParameters().get(1).setValue(queryParams[1]);
  report.getAllParameters().get(2).setValue(queryParams[2]);
  report.getAllParameters().get(3).setValue(formulaParams[0]);

  try {
   // Get the data for the report
   report.refreshWithOriginalData();
```

```
 } catch (Exception ex)
 {
  ex.printStackTrace();
 }

 // Show report in Viewer
 return (new Viewer().getComponent(report));
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/GetAllParameters.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/GetAllParameters.pdf]

The parameter prompt dialogs in Report Designer, Page Viewer, and Menu Page will display the parameters in ordered sequence. Use the overloaded method with the parameter value set to true to get a list of ordered parameters.

```
report.getAllParameters(boolean ordered)
```

The results of this method will maintain two qualities. First, formula parameters will always be return first. Second, cascading parameter ordering will be maintained. For more information regarding cascading parameters, please see Section 1.3.2.2.2.3 - Cascading Parameters.

2.3.5.1.5. Secure Reports

Information on the report can be changed by passing in a Security parameter when creating the report. The security levels are created in Designer and the appropriate security level is passed by using the API.

The following example, which can be run as an applet or application, opens a secure report that shows sales information for every region except West:

```
Component doSecurity(Object parent) {

 // Do not use EspressManager
 QbReport.setEspressManagerUsed(false);

 // Open the template
 QbReport report = new QbReport(parent, // container
   "Security.rpt"); // template

 try {
  // Set Security Level
  report.setSecurityLevel("NoWest");
 } catch (Exception ex)
 {
  ex.printStackTrace();
 }

 // Show report in Viewer
 return (new Viewer().getComponent(report));
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/Security.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/Security.pdf]

You can also pass in the security level using a Properties object. This is especially useful when query/formula parameters are secure.

The following example, which can be run as an applet or application, opens a secure report that shows sales information for every region except West:

```
Component doSecurityProperties(Object parent) {
 // Do not use EspressManager
 QbReport.setEspressManagerUsed(false);

 // Set up Properties
 Properties props = new Properties();
 props.put("security level", "NoWest");

 // Open the template
 QbReport report = new QbReport(parent, // container
   "Security.rpt", // template
   props); // properties

 // Show report in Viewer
 return (new Viewer().getComponent(report));
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SecurityProperties.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/SecurityProperties.pdf]

2.3.5.2. Applying a Report Template

When a `QbReport` object is created from scratch (see Appendix 2.C - Creating the Report), the report is created using default attributes. However, you can use a report template (either .rpt or .xml) to specify user defined attributes during report construction. Almost all the attributes (except for data source and report type) are extracted from the template and applied to the `QbReport` object. The template name usually appears as the last argument in the `QbReport` constructors.

You can also specify the template name using the `applyTemplate(String fileName)` method in the `QbReport` class.

The following example, which can be run as an applet or application, applies a template onto the `QbReport` object:

```
Component doApplyingTemplate(Object parent) {

    // Do not use EspressManager
    QbReport.setEspressManagerUsed(false);

    String templateLocation = "..";

    // Apply the template
    QbReport report = new QbReport (parent, // container
                    QbReport.SUMMARY, // report type
                    data, // data
                    columnMapping, // column mapping
                    templateLocation); // template

    // Show report in Viewer
    return (new Viewer().getComponent(report));

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ApplyingTemplate.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/ApplyingTemplate.pdf]

Please note that the above code is not complete and is there as a guide. However, the link contains a complete application code that can be run.

You can also take the column mapping from the template and have it applied on the `QbReport` object being created. This is done by passing in “null” (without the double quotes) instead of a `ColInfo[]` object.

In addition to the column mapping, you also obtain the database connection information from the template (assuming the template uses a database as the datasource). This is done by passing in “null” (without the double quotes) for any of the DBInfo or SimpleQueryFileInfo parameters.

By default, when you apply templates, formulas in the table data section are not applied. To apply formulas and/or scripts from the template .rpt/.xml file to the QbReport object, you will need to use the following method:

```
QbReport.applyTemplate(String templateName, boolean applyFormula);
```

2.3.5.3. Modifying Data Source

You can create report templates in Designer and open those templates using the API. The QbReport object created uses the same data source as the template and attempts to fetch the data. However, it may be that while the template has all the look and feel needed, the data source may be an incorrect one. The following sections show how to open the template with backup data and switch the data source, without recreating the entire report.

Please note that for best results, the number of columns and the data type of each column must match between the two data sources (i.e., the one used to create the template in Designer and the new data source).

After switching the data source, the QbReport object must be forced to fetch the new data. This can be done by calling the refresh method in the QbReport class.

2.3.5.3.1. Data from a Database

Switching the data source to point to a database is simple. All you would need to do is provide the database connection information as well as the query to be used and pass that to the QbReport object. You can provide the database connection information (as well as the query) in a DBInfo object (for more information on creating a DBInfo object, please refer to Appendix 2.B.1 - Data from a Database).

The following example, which can be run as an applet or application, switches the data source of the QbReport object to a database:

```
Component doSwitchToDatabase(Object parent) {

    // Do not use EspressManager
    QbReport.setEspressManagerUsed(false);

    // Open the template with backup data
    QbReport report = new QbReport (parent, // container
                    "SwitchToDatabase.rpt", // template
                    false, false, false, true);

    // New database connection information
    DBInfo newDatabaseInfo = new DBInfo(.....);

    try {
        // Switch data source
        report.getInputData().setDatabaseInfo(newDatabaseInfo);

        // Refresh the report
        report.refresh();

    } catch (Exception ex)
    {
        ex.printStackTrace();

    }

    // Show report in Viewer
    return (new Viewer().getComponent(report));

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SwitchToDatabase.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/SwitchToDatabase.pdf]

Please note that the above code is not complete and is there as a guide. However, the link contains a complete application code that can be run.

The above approach changes the data source for the current report level only (it can be either the main report or the sub-report).

If your report template has any sub-reports, drill-down reports, or independent charts (i.e., charts that do not use the report data as the data source) and you wish to change the data sources for all of them, then you can use the `setAllDatabaseInfo` method under the `IInputData interface`. Please note that with this approach, a new query cannot be specified and the original query will be used. Only the database connection information will be changed.

The following example, which can be run as an applet or application, uses the `setAllDatabaseInfo` method to switch the data source:

```
Component doSwitchToDatabaseSetAll(Object parent) {

  // Do not use EspressManager
  QbReport.setEspressManagerUsed(false);

  // Open the template with backup data
  QbReport report = new QbReport(parent, // container
    "SwitchToDatabase.rpt", // template
    false, false, false, true);

  try {
   // Switch data source
   report.getInputData().setAllDatabaseInfo("jdbc:hsqldb:woodview",
     "org.hsqldb.jdbcDriver", "sa", "");

   // Refresh report
   report.refresh();
  } catch (Exception ex)
  {
   ex.printStackTrace();

  }

  // Show report in Viewer
  return (new Viewer().getComponent(report));
 }
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SwitchToDatabaseSetAll.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/SwitchToDatabaseSetAll.pdf]

Please note that the above code is not complete and is there as a guide. However, the link contains a complete application code that can be run.

2.3.5.3.1.1. Parameterized

Just as with a regular query, you can switch the data source to a parameterized query. With a parameterized query, the parameter(s) properties as well as the database connection information and the query, must be specified.

The following example, which can be run as an applet or application, switches the data source of the QbReport object to a parameterized query:

```
Component doSwitchToDatabaseParam(Object parent) {
    // Do not use EspressManager
    QbReport.setEspressManagerUsed(false);

    // Open the template with backup data
    QbReport report = new QbReport (parent, // container
                    "SwitchToDatabaseParam.rpt", // template
                    false, false, false, true);

    // New database connection and parameter information
    SimpleQueryFileInfo newDatabaseInfo = new SimpleQueryFileInfo(.....);

    try {
        // Switch data source
        report.getInputData().setDatabaseInfo(newDatabaseInfo);

        // Refresh the report
        report.refresh();

    } catch (Exception ex)
    {
        ex.printStackTrace();

    }

    // Show report in Viewer
    return (new Viewer().getComponent(report));

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SwitchToDatabaseParam.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/SwitchToDatabaseParam.pdf]

Please note that the above code is not complete and is there as a guide. However, the link contains a complete application code that can be run.

Again, just like with regular queries, you can use the `setAllDatabaseInfo` method to switch to the new data source. In this approach, the original query and parameter information is used while the database connection information is altered. After switching the database information, the parameter value(s) must be specified before refreshing the report.

The following example, which can be run as an applet or application, uses the `setAllDatabaseInfo` method to switch the data source:

```
Component doSwitchToDatabaseParamSetAll(Object parent) {

    // Do not use EspressManager
    QbReport.setEspressManagerUsed(false);

    // Open the template with backup data
    QbReport report = new QbReport (parent, // container
                    "SwitchToDatabaseParam.rpt", // template
                    false, false, false, true);

    try {
        // Switch data source
        report.getInputData().setAllDatabaseInfo(...);

        // Pass in parameter value
        report.getAllParameters().get(0).setValue("TRD");

        // Refresh the report
        report.refresh();

    } catch (Exception ex)
    {
        ex.printStackTrace();

    }

    // Show report in Viewer
    return (new Viewer().getComponent(report));

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SwitchToDatabaseParamSetAll.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/SwitchToDatabaseParamSetAll.pdf]

Please note that the above code is not complete and is there as a guide. However, the link contains a complete application code that can be run.

2.3.5.3.1.2. JNDI

You can also change the data source to a JNDI data source. This is done by specifying the JNDI connection information in a `DBInfo` object and then passing it to the `QbReport` object.

The following example, which can be run as an applet or application, switches the data source of the QbReport object to a JNDI database:

```
Component doSwitchToDatabaseJNDI(Object parent) {

 // Do not use EspressManager
 QbReport.setEspressManagerUsed(false);

 // Data Source.  Replace comp with computer and env with environment.
 // The environment hashtable is empty for tomcat.
 // If other application server is used, need to set INITIAL_CONTEXT_FACTORY
 and PROVIDER_URL.
 DBInfo newDatabaseInfo = new DBInfo(
   "java:comp/env/jdbc/Woodview",
   "SELECT Categories.CategoryName, Products.ProductName,
 Products.UnitPrice, Products.StainPrice, Products.UnitsInStock FROM
 Products, Categories WHERE (Products.CategoryID = Categories.CategoryID)",
   new Hashtable());

 // Open the template with backup data
 QbReport report = new QbReport(parent, // container
   "SwitchToDatabaseJNDI.rpt", // template
   false, false, false, true);

 try {
  // Switch data source
  report.getInputData().setDatabaseInfo(newDatabaseInfo);

  // Refresh report
  report.refresh();
 } catch (Exception ex)
 {
  ex.printStackTrace();
 }

 // Show report in Viewer
 return (new Viewer().getComponent(report));
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SwitchToDatabaseJNDI.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/SwitchToDatabaseJNDI.pdf]

Please note that the above code is not complete and is there as a guide. However, the link contains a complete application code that can be run. Note that for the application code to run, the Woodview database needs to be set up as a JNDI data source in the Tomcat environment and the application code changed to match the connection information.

2.3.5.3.2. Data from a Data File (TXT/DAT/XML)

You can switch the data source to a text file as long as the text file follows the Quadbase guidelines (for more details, please refer to Section 2.3.5.3.2 - Data from a Data File (TXT/DAT/XML)).

The following example, which can be run as an applet or application, switches the data source of the `QbReport` object to a text file:

```
Component doSwitchToDataFile(Object parent) {

 // Do not use EspressManager
 QbReport.setEspressManagerUsed(false);

 // Open the template with backup data
 QbReport report = new QbReport(parent, // container
   "SwitchToDataFile.rpt", // template
   false, false, false, true);

 try {
  // Switch data source
  report.getInputData().setDataFile("sample.dat");

  // Refresh report
  report.refresh();
 } catch (Exception ex)
 {
  ex.printStackTrace();
 }

 // Show report in Viewer
 return (new Viewer().getComponent(report));
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SwitchToDataFile.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/SwitchToDataFile.pdf]

Please note that the above code is not complete and is there as a guide. However, the link contains a complete application code that can be run.

You can also specify whether the data is sorted (this improves performance) and/or the encoding for the text file. This can be done using the following method in `IInputData`:

```
setDataFile(String dataFile, boolean sortedData, String encoding);
```

2.3.5.3.3. Data from an XML Data Source

You can switch the data source to your custom XML data as long as there is a .dtd or .xml schema accompanying your data. The XML data information is specified (for more details, please refer to Section 2.3.5.3.3 - Data from an XML Data Source) and then passed to the `QbReport` object.

The following example, which can be run as an applet or application, switches the data source of the QbReport object to XML data:

```
Component doSwitchToXMLData(Object parent) {

    // Do not use EspressManager
    QbReport.setEspressManagerUsed(false);

    // Open the template with backup data
    QbReport report = new QbReport (parent, // container
                    "SwitchToXMLData.rpt", // template
                    false, false, false, true);

    // XML data source information

    XMLFileQueryInfo newData = new XMLFileQueryInfo(...);

    try {
        // Switch data source
        report.getInputData().setXMLFileQueryInfo(newData);

        // Refresh the report
        report.refresh();

    } catch (Exception ex)
    {
        ex.printStackTrace();

    }

    // Show report in Viewer
    return (new Viewer().getComponent(report));

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SwitchToXMLData.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/SwitchToXMLData.pdf]

Please note that the above code is not complete and is there as a guide. However, the link contains a complete application code that can be run.

You can also specify whether the data is sorted (this improves performance) by using the following method in `IInputData`:

```
setXMLFileQueryInfo(XMLFileQueryInfo xmlInfo, boolean sortedData);
```

2.3.5.3.4. Data from Custom Implementation

In addition to the regular data sources, you can also pass in your own custom data. The custom data is passed to the QbReport object using either the IDataSource [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/reportdesigner/util/IDataSource.html] or IParameterizedDataSource [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/reportdesigner/util/IParameterizedDataSource.html] interfaces. For more details, please refer to Appendix 2.B.5 - Data passed in a Custom Implementation.

The following example, which can be run as an applet or application, switches the data source to a custom implementation:

```
Component doSwitchToCustomData(Object parent) {

 // Do not use EspressManager
 QbReport.setEspressManagerUsed(false);

 // Open the template with backup data
 QbReport report = new QbReport(parent, // container
   "SwitchToCustomData.rpt", // template
   false, false, false, true);

 try {
  // Switch data source
  report.getInputData().setClassFile("Furniture_Report");

  // Refresh report
  report.refresh();
 } catch (Exception ex)
 {
  ex.printStackTrace();
 }

 // Show report in Viewer
 return (new Viewer().getComponent(report));
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SwitchToCustomData.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/SwitchToCustomData.pdf]

Please note that the above code is not complete and is there as a guide. However, the link contains a complete application code that can be run.

You can also specify whether the data is sorted (this improves performance) and/or to prompt the parameter dialog by using the following method in `IInputData`:

```
setClassFile(String classname, boolean sortedData, boolean
 showPromptDialog);
```

2.3.5.3.4.1. Parameterized

You can have a custom implementation that requires parameter values, to be the new data source. In this scenario, the custom implementation must use the IParameterizedDataSource [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/reportdesigner/util/IParameterizedDataSource.html] interface. For more details, please refer to Appendix 2.B.5 - Data passed in a Custom Implementation. After switching the data source information, the parameter value(s) must be specified before refreshing the report.

The following example, which can be run as an applet or application, switches the data source to a parameterized custom implementation:

```
Component doSwitchToCustomDataParam(Object parent) {

 // Do not use EspressManager
 QbReport.setEspressManagerUsed(false);

 // Open the template with backup data
 QbReport report = new QbReport(parent, // container
   "SwitchToCustomDataParam.rpt", // template
   false, false, false, true);

 try {
  // Switch data source
  report.getInputData().setClassFile("ProductParamInfo", false, false);

  report.getAllParameters().get(0).setValue(new Integer(25));

  // Refresh report
  report.refresh();
 } catch (Exception ex)
 {
  ex.printStackTrace();
 }

 // Show report in Viewer
 return (new Viewer().getComponent(report));
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SwitchToCustomDataParam.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/SwitchToCustomDataParam.pdf]

Please note that the above code is not complete and is there as a guide. However, the link contains a complete application code that can be run.

2.3.5.3.5. Data passed in an Array in Memory

You can also pass in data using arrays. The array data is usually stored in memory and passed to the `QbReport` object (for more details, please refer to Appendix 2.B.4 - Data passed in an Array in Memory).

The following example, which can be run as an applet or application, switches the data source to an array in memory:

```
Component doSwitchToArrayData(Object parent) {

    // Do not use EspressManager
    QbReport.setEspressManagerUsed(false);

    // Open the template with backup data
    QbReport report = new QbReport (parent, // container
                    "SwitchToArrayData.rpt", // template
                    false, false, false, true);

    // Create array data
    DbData newData = new DbData(...);

    try {
        // Switch data source
        report.getInputData().setData(newData);

        // Refresh the report
        report.refresh();

    } catch (Exception ex)
    {
        ex.printStackTrace();

    }
    // Show report in Viewer
    return (new Viewer().getComponent(report));

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SwitchToArrayData.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/SwitchToArrayData.pdf]

Please note that the above code is not complete and is there as a guide. However, the link contains a complete application code that can be run.

You can also specify whether the data is sorted, which improves performance, by using the following method in `IInputData`:

```
setData(IResultSet rs, boolean sortedData);
```

2.3.5.3.6. Drill-Down with DrillDownReportServlet

When changing the data source for a Drill-Down report, you use the `setAllDatabaseInfo` method in `IInputData`. However, if you are exporting the report and using the `DrillDownReportServlet` (for more details, please see Section 2.3.5.7.10.2 - DrillDownReportServlet), you need to get a handle to the session and use the method `setDrillDownDatabaseInfo` in `QbReport` so that the drill-down layers will know what the new data source is.

The following example, which can be run as an applet or application, switches the data source of a drill-down report:

```
public void doGet(HttpServletRequest req, HttpServletResponse res) throws
 ServletException,
 IOException
{
 HttpSession session = req.getSession(true);

 // Set the "content type" header of the response
 res.setContentType("text/html");

 // Get the response's OutputStream to return content to the client.
 OutputStream toClient = res.getOutputStream();

 try
 {
  // Do not use EspressManager
  QbReport.setEspressManagerUsed(false);

  // Open report with backup data (data source will be switched later)
  QbReport report = new QbReport(null, "SwitchDrillDownServlet.pak",
    false, false, false, true);

  // New database connection information
  String newDatabaseURL = "jdbc:hsqldb:woodview";
  String newDatabaseDriver = "org.hsqldb.jdbcDriver";
  String newDatabaseUID = "sa";
  String newDatabasePassword = "";

  // Switch data source
  report.getInputData().setAllDatabaseInfo(newDatabaseURL,
 newDatabaseDriver,
    newDatabaseUID, newDatabasePassword);

  // Put new data source information in session for DrillDownReportServlet
  report.setDrillDownDatabaseInfo(session, newDatabaseURL,
 newDatabaseDriver,
    newDatabaseUID, newDatabasePassword);

  report.setDynamicExport(true, "localhost", 8080);

  ByteArrayOutputStream tempStream = new ByteArrayOutputStream();

  report.refresh();

  // Export the report to DHTML
  report.export(QbReport.DHTML, tempStream);

  tempStream.writeTo(toClient);
 } catch (Exception e) {
  e.printStackTrace();
 }

 // Flush the outputStream
 toClient.flush();

 // Close the writer; the response is done.
 toClient.close();
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SwitchDrillDownServlet.zip]

Exported Results Root [https://data.quadbase.com/Docs71/help/manual/code/export/SwitchDrillDownServlet-Root.html]

Exported Results Drill-Down Level [https://data.quadbase.com/Docs71/help/manual/code/export/SwitchDrill-DownServletDrill.html]

Please note that the above code is not complete and is there as a guide. However, the link contains a complete application code that can be run. You would have to change the location of the template and deploy the servlet in order to run the code successfully.

You can also specify a connection object for the database by using the following method in `QbReport`.

```
setDrillDownConnection(Object session, Connection conn);
```

2.3.5.4. Modifying Column Mapping

Just as the data source of a report can be changed, the column mapping can be modified using the API as well. However, this is not recommended as the report may have formulas and/or scripts that are data dependent. If the number of columns and/or the data of the columns do not match the original mapping, certain formulas and/or scripts may not work. While this section shows how to switch the mapping, it is recommended that in such a scenario a new `QbReport` object be created (see Appendix 2.C - Creating the Report) and a template applied on it.

The following example, which can be run as an applet or application, modifies the column mapping of the report:

```
Component doModifyColumnMapping(Object parent) {

    // Do not use EspressManager
    QbReport.setEspressManagerUsed(false);

    // Open the template with backup data
    QbReport report = new QbReport (parent, // container
                    "ModifyColumnMapping.rpt", // template
                    false, false, false, true);
    // Create new column mapping
    ColInfo[] newColumnMapping = new ColInfo[..];

    try {
        // Switch mapping
        report.getInputData().setMapping(newColumnMapping);

        // Refresh the report
        report.refreshWithOriginalData();

        // Reapply template
        report.applyTemplate("ModifyColumnMapping.rpt");

    } catch (Exception ex)
    {
        ex.printStackTrace();

    }

    // Show report in Viewer
    return (new Viewer().getComponent(report));

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ModifyColumnMapping.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/ModifyColumnMapping.pdf]

Please note that the above code is not complete and is there as a guide. However, the link contains a complete application code that can be run.

2.3.5.5. Report Components

An EspressReport report is made of different segments. Each segment can be set and modified independently from the others. Listed below are the various parts of the report :

2.3.5.5.1. ReportElement

A `ReportElement` object forms the core of objects in Report API. A `ReportElement` object provides a way to manipulate the contents as well as the look and feel of the individual elements. All the other report objects (such as `ReportCell`, `ReportSection`, `ReportColumn` etc.) extend the `ReportElement` class. Every part of the report is comprised of a `ReportElement` object, or an extension of it. Within each `ReportElement` object, properties such as font, color, data format, etc can be manipulated. You can also copy `ReportElement` objects or just apply the template of one `ReportElement` object to another. When a template is applied, the data of the object does not change. Only the look and feel of the `ReportElement` object is modified.

When modifying it, you do not get a handle to the `ReportElement`, but rather use the methods of the subclass (`ReportCell`, `ReportSection`, `ReportColumn` etc.)

2.3.5.5.2. ReportCell

A `ReportCell` object is used to insert labels, formulas, text, charts, or even images into different sections of the report (i.e., in the ReportSection(s) of the report). You can also use `ReportCell` objects to help with the formatting of the cells in the various parts of the report.

You can get a handle to a cell by using its numeric index (i.e., the position of the cell in the section), its ID (which is a string), or its custom ID (assuming the cell has one). You can use the following methods, in ReportSection, to get the desired cell:

```
getData(int i);
getData(String ID);
```

You can also get all the report cells, in a given section, by using the following method in ReportSection:

```
getData();
```

2.3.5.5.3. ReportSection

A report is divided into many sections. The following shows the different sections of a report:

Page Header :- This section serves as a header to the page. It appears at the top of everypage of the report.

Table Header :- This section serves as a header to the detail or data section of the report. By default it only appears once in the report.

Group Header:- This section appears in reports with grouped data (i.e. Master & Details report), or data with row breaks inserted (i.e. Summary Break report). It repeats at the top of each grouping within the report.

Table Data:- This is the main section of the report that contains most of the data. Data columns that have been selected for the report are placed in this section, and repeated for each entry in the column.

Group Footer:- This section appears on reports with grouped data (i.e. Master & Details report), or data with row breaks inserted (i.e. Summary Break report). It repeats at the bottom of each grouping within the report.

Table Footer:- This section serves as the footer to the detail or data section of the report. By default it appears only once in the report.

Page Footer:- This section serves as a footer to the page. It appears at the bottom of every page of the report.

Report Footer:- This is the last summary or footer section of the report. It only appears once at the end of the report.

You can get a handle to the Page Footer, Page Header, Report Footer, Report Header, and Report Table sections using the following methods in `QbReport`:

```
getPageFooter();
getPageHeader();
getReportFooter();
getReportHeader();
getTable();
```

and you can get a handle to the Group Footer, Group Header, Table Header, and Table Footer sections using the following methods in Report Table:

```
getRowBreakFooter(int breakLevel);
getRowBreakHeader(int breakLevel);
getFooter();
getHeader();
```

2.3.5.5.4. ReportColumn

A `ReportColumn` object is an array of `ReportCell` objects that appears in the column selected. Using `ReportColumn` objects, formatting can be done through the entire column once, instead of applying the formatting one cell at a time.

You can get a handle to a `ReportColumn` by using the following method in `ReportTable`:

```
getColumn(int index);
```

2.3.5.5.5. ReportTable

This is the main part of the report. This is the section that contains the data columns that have been selected for the report. This section contains the data that is used for the Group Footers and for the various summaries.

You can get a handle to the ReportTable section by using the following method in `QbReport`:

```
getTable();
```

2.3.5.5.6. ReportImage

`ReportImage` objects are used to add images to the report. The formats supported by EspressReport are GIF, JPEG, and PNG. The ReportImage object can be considered as a cell that contains a image. Care is to be taken to define the dimensions of the cell so that the image is clearly visible and is not truncated.

You can get a handle to all the `ReportImage` objects by using the following method in `QbReport`:

```
getReportImages();
```

2.3.5.5.7. ReportChartObject

ReportChartObjects are used to add charts of type TPL or CHT (Quadbase's proprietary formats) to the report. Adding in the `ReportChartObject` is different from adding in a `ReportImage` object. Here, the location of the TPL or template file is given. EspressReport then takes the template and creates a chart using the data from the relevant section. Thus, using the same template, you can have different charts at different points of the report, all of which share the same look and feel, even though the data might be different.

The chart template files are first created in Report Designer. Thus, the mapping of the chart is based on the mapping of the data in the report. We recommend using chart templates from the same type of report containing similar data mapping (i.e., the same kind of data and data type) as the report being generated using the API.

ReportChartObjects can also be used to add stand-alone charts (i.e., charts whose data is not from the report). Stand-alone charts are created from the API and the data source for the chart can be independent from the report.

Again, a `ReportChartObject` object can be considered as a cell that contains a chart. Similar care should be taken to define the dimensions of the cell so that the chart is clearly visible and not truncated.

You can get a handle to all the `ReportChart` objects by using the following method in `QbReport`:

```
getReportChartObjects();
```

2.3.5.5.8. ChartObject

ChartObjects are intermediate objects created before adding a chart, created completely through the API, to the report. The chart obtains its data from the report and the mapping for the chart is based on the columns of the report. These charts are different from stand-alone charts as the data for these charts is the report itself whereas the data for stand-alone charts can be independent from the report. Other chart properties such as dimension, chart type and mapping are also specified and the `ChartObject` created.

The ChartObject is then added to a `ReportChartObject`. This `ReportChartObject` is then added to the report.

For more details on how to use `ChartObject`, please refer to the Chapter 3 - Charting Guide.

2.3.5.5.9. ReportDocument

`ReportDocument` is used to represent a clob (character large object) when adding a clob to the report. Depending on the length of the clob, it is either stored as a string in the `ReportDocument` or is stored in a file and the filename is stored in the `ReportDocument`.

Again, a `ReportDocument` object can be considered as a cell that contains a clob. Similar care should be taken to define the dimensions of the cell so that the content is clearly visible and not truncated.

We recommend that the `ReportDocument` object not be modified using the API and that all changes be done in the template.

2.3.5.5.10. ReportRTFObject

`ReportRTFObject` is used to add content from a RTF file to the report. The file content is streamed to a `ReportRTFObject` and this object is later added to the report.

Again, a ReportRTFObject object can be considered as a cell that contains RTF content. Similar care should be taken to define the dimensions of the cell so that the content is clearly visible and not truncated.

We recommend that the ReportRTFObject object is not be modified by using the API and that all changes be done in the template.

2.3.5.5.11. SubReportObject

SubReportObject is used to add sub-reports to the report. The sub-report content can either be from a file or created completely from the API.

Again, a SubReportObject object can be considered as a cell that contains a subreport. Similar care should be taken to define the dimensions of the cell so that the content is clearly visible and not truncated.

You can get a handle to all the SubReportObject objects by using the following method in QbReport:

```
getSubReports();
```

You can also get a handle to the subreport from the SubReportObject by using the following method in SubReportObject:

```
getSubReport(IReport qbReport);
```

The SubReport object obtained can then be casted to QbReport.

2.3.5.5.12. ReportGrid

ReportGrid objects are used to draw grids within different sections of the table. The grid will encompass each row and each column (and each cell in the section, if applicable). Currently, the line styles available are dash, double, and solid line styles.

We recommend that the ReportGrid object not be modified using the API and that all changes be done in the template.

2.3.5.5.13. ReportLine

ReportLine objects are used to draw a line, or lines, in different sections of a table. Lines can be inserted in any part of a report. As with ReportGrid objects, ReportLine objects have a choice of three styles; dash, double, and solid line styles

We recommend that the ReportLine object not be modified using the API and that all changes be done in the template.

2.3.5.6. Modifying Report Attributes

EspressReport has hundreds of properties, which provide a fine control over the various elements of a report. As a developer, you can customize the look and feel of a report dynamically at run-time. In order to facilitate ease-of-use, most properties have been categorized into groups and exposed in the form of interfaces. An application first obtains a handle to a group interface using a getXXX method and then manipulates the report's properties directly by calling methods on that interface. Most interfaces are contained in the package quadbase.reportdesigner.util.

2.3.5.6.1. Adding New Cells

EspressReport allows you to create new cells and position them where you want them in the report.

To add a cell to the report, you will need to create a ReportCell object (though it does not have to be a ReportCell object. It can also be a ReportImage object, a ReportChartObject, a ReportLine object or a ReportGrid object depending on the type of information you are adding). After creating and specifying the ReportCell properties, you can add it to any part of the report.

In the following example, a cell is added as the Page Footer:

```
ReportCell cell = new ReportCell("Inventory Report");
cell.setWidth(7);
cell.setAlign(IAlignConstants.ALIGN_RIGHT);
```

```
report.getPageFooter().addData(cell);
```

You can also apply another cell's look and feel onto the newly created ReportCell object by using the following method in ReportCell:

```
applyTemplate(ReportCell templateCell, boolean applyScript);
```

2.3.5.6.2. Adding Images/Charts

Both images and charts can be included in practically any part of the report. Images of type GIF, JPEG, PNG, and charts of type TPL (Quadbase's proprietary format) are supported by EspressReport. Adding charts/images to a report generally allows for better presentation and imparts more information and makes the report easier to read and understand.

To add an image, you will need to define a ReportImage object. Within the ReportImage object, you will have to pass in the URL for the image, either as an http:// or a file:// URL. It is recommended that you use an http:// URL so that if the report is exported to an DHTML format because the DHTML report picks up the image consistently. You can also specify the image type, dimensions, and alignment of the ReportImage object and then add it to the location desired in the report.

In the following example, an image called logo3 of type gif is added to the Page Header:

```
ReportImage reportImage = new ReportImage();
try {

    reportImage.setImageType(IExportConstants.GIF);
    java.net.URL imageLocation =
            new java.net.URL ("http://someMachineName/Gifs/logo3.gif");
    reportImage.setImageURL(imageLocation);
    reportImage.setWidth(7);
    reportImage.setHeight(1);
    reportImage.setAlign(IAlignConstants.ALIGN_LEFT);

} catch (Exception ex)  {

    ex.printStackTrace();

}

report.getPageHeader().addData(reportImage);
```

You can also specify a relative link to be used instead of the complete URL specified in the code (or in Report Designer). This can be done by passing in the relative path in the following method in ReportImage:

```
public void setImagePath(String path);
```

For example, passing ../../logo3.gif in the above method would result in the DHTML export having a relative link in the <img src> tag rather than the absolute URL.

To add a chart, please refer to Appendix 3.C.15 - Adding a Chart to a Report.

2.3.5.6.3. Adding Hyperlinks

You can also add hyperlinks to a cell or several cells in the report and have different hyperlinks for different cells. Hints to the link and targets can be set up in the cell and the link is then preserved when the report is exported. Note that links to RPT files can also be inserted in a report. However, the link to such files will not work outside of an applet environment. To view RPT files in DHTML format, we recommend you to create separate DHTML files and link to those files instead.

In the following example, a cell is created which contains a link to a page and is added to the Page Header:

```
ReportCell cell = new ReportCell("ABC Incorporated);
cell.setLink(new String("http://www.quadbase.com"));
cell.setHint(new String("Click here to go to Quadbase's Homepage"));
cell.setTarget(new String("ABC Home Page"));
report.getPageHeader().addData(cell);
```

2.3.5.6.4. Adding Grid Lines and Lines

While you can add a ReportGrid and ReportLine to a report using the API, we recommend that all grids and lines be inserted into a report using the Designer. To add grid lines to a report, you must define a ReportGrid object. Within the ReportGrid object, you can define various properties such as color, line style, thickness, width etc. before specifying a section to be encompassed by the grid.

In the following example, a blue grid line is added around the Report Table:

```
ReportGrid reportGrid = new ReportGrid();
reportGrid.setBorderColor(Color.blue);
reportGrid.setGridStyle(ReportGrid.DOUBLE);
reportGrid.setBorder(1);
reportGrid.setWidth(7);
report.getTable().addImage(reportGrid);
```

To add lines to the report, you must define a ReportLine object. Here, you can also define its various properties such as color, line style, thickness, width... etc before specifying the section it is added to.

In the following example, a line is added to the Page Header:

```
ReportLine reportLine = new ReportLine(false);
reportLine.setBorderColor(Color.red);
reportLine.setLineStyle(ReportLine.DOUBLE);
reportLine.setBorder(1);
reportLine.setWidth(7);
report.getPageHeader().addData(reportLine);
```

2.3.5.6.5. Adding Rich Text Field Objects

To add rich text fields to a report, you must define a ReportRTFObject object. Within the ReportRTFObject object, you can specify the location of the .rtf file and then specify the other properties of the cell before specifying a section to add the object to. Please note that a .rtf file must exist in order to add a ReportRTFObject to the report.

We recommend, however, that you do not include RTF content using this approach but rather add it to the template in the Designer.

In the following example, the content of the file RText1.rtf is added to the table data section:

```
ByteArrayOutputStream fileOne = new ByteArrayOutputStream();
FileInputStream fileIn = new FileInputStream("RText1.rtf");

int b = fileIn.read();
while (b != -1) {

    fileOne.write(b);
    b = fileIn.read();

}

ReportRTFObject rtfObjectOne = new ReportRTFObject(fileOne.toByteArray());
rtfObjectOne.setWidth(1);
rtfObjectOne.setHeight(.3);
rtfObjectOne.setResizeToFitContent(true);
rtfObjectOne.setX(report.getTable().getColumn(0).getWidth() +
 report.getTable().getColumn(1).getWidth() +
      report.getTable().getColumn(2).getWidth());    // Set Object at the
 end of table data
```

```
report.getTable().addRTFObject(rtfObjectOne);
// report is an object of type QbReport
```

The RTF file content can also be displayed in multiple column format. Thus, instead of the default column count of 1, you can show the content in 2 or more columns. The column count and the spacing between the columns can be set using the following methods:

```
ReportRTFObject.setColumnCount(int columnC);
ReportRTFObject.setColumnSpacing(double space);
```

For example, in the above example, if the following lines of code were added:

```
rtfObjectOne.setColumnCount(3);
rtfObjectOne.setColumnSpacing(.3);
```

before adding the `ReportRTFObject` to the report, the report would now show the RTF portion in a three column format with .3 inches separating two adjacent columns.

For more details on rich text field content, please refer to the Section 1.5.7.7 - Rich Text Fields.

2.3.5.6.6. Adding Nested Sections

To add in a nested section (nested sections) in a report, you must get a handle to the parent section and created a child section. Nested sections are extremely useful when using in conjunction with any cell or object that need resizing without the overlapping any objects below. Please note that nested sections inherit most of the section options from their parent sections except the page-breaking option.

In the following example, two nested sections are being created within the Table Header section and a `ReportCell` object (`reportCell`) is being added to one of the sections:

```
ReportSection tableHeader = report.getTable().getHeader();
tableHeader.addSection();
tableHeader.insertSection(0);
tableHeader.getSection(0).setHeight(.5);
tableHeader.getSection(1).setHeight(.5);
tableHeader.getSection(0).addData(new ReportCell("Test Cell"));
```

The index used for nested sections follows a vector index. Please note that a section will not appear even if the height is set, if there are no cells within the section.

2.3.5.6.7. Modifying Background Color, Font .....

Properties for any section of the report can be modified by getting the appropriate handle and calling the methods. Since all parts of the report extends `ReportElement`, almost all the properties (which can be applied to a single `ReportElement` object) are here and can be applied one at a time, onto a group, or in groups.

For instance, the code below will set the background color of the `ReportCell` object to red:

```
ReportCell cell;
cell.setBgColor(Color.red);
```

The code fragment below would set the background color of the first column of the report to be blue:

```
ReportTable table = report.getTable();
table.getColumn(0).setBgColor(Color.blue);
```

To set all the columns of the table to blue, you need to run through a for-loop, such as one given below:

```
for (int i = 0; i < table.getColumnCount() ; i++)
    table.getColumn(i).setBgColor(Color.blue);
```

Note that writing the following code

```
table.setBgColor(Color.blue);
```

does **NOT** set the columns in the table to have the color blue. It merely sets the color of the table section (i.e., the area behind the cells in the column) to be blue.

Similarly, the code below:

```
ReportCell cell;
cell.setFont(new Font("Arial", Font.BOLD, 14);
table.getColumn(0).setFont(new Font(table.getFont().getName(), Font.BOLD,
 16));
```

sets the font for the cell and for the first column of the report. Note that setting the font for any part of the report other than a `ReportCell` or a `ReportColumn` object (such as a `ReportTable` or a `ReportSection`) involves setting the font for each individual cell in the section. You can create your own method, which sets the font for any section of the report. For instance, the code below:

```
void setFont(Font font, ReportElement elts[]) {

    if (elts == null) return;

    for (int i = 0; i < elts.length; i++) {
        elts[i].setFont(font);

}

setFont(new Font(table.getFont().getName(), Font.BOLD, 16),
 rowBreakHeaderZero.getData());
setFont(new Font(table.getFont().getName(), Font.BOLD, 18),
 tableHeader.getData());
```

creates your own method and uses it to set the font for the Row Break Zero Header and for the Table Header.

If you are planning to export your report in PDF format, True Type Fonts can also be mapped and used with the following method:

```
report.setFontMapping(String fontName, int style, String ttf);
```

where `fontName` is the name of the font, style is a `QbReport` FONT constant, and `ttf` is the path and filename to the installed .ttf font. For example:

```
report.setFontMapping("Dialog", QbReport.BOLDITALIC, "C:/EspressReport/help/
examples/fonts/bookosbi.ttf");
```

For a more detailed description of PDF Font Mapping, please see the Section 1.7.2.1 - PDF Font Mapping.

Similarly, other properties can be set using the methods in the interfaces.

2.3.5.6.8. Modifying the Format

You can also change the format of any data (be it Numeric, boolean, or String) to conform to your requirements. The format can be set for individual `ReportCell` objects or for `ReportColumn` objects.

For example, the code below:

```
NumericFormat contentFormat = new NumericFormat();
contentFormat.decimal = 2;
contentFormat.currencySymbol = '$';

for (int i = table.getStartOfColumnBreakColumn(); i <
 table.getColumnCount(); i++ )
    table.getColumn(i).setDataFormat(contentFormat);
```

sets the format of the numeric data of the column break columns in the cross tab report. It adds a “$” symbol and sets the cells to show two decimal places.

Similarly, the properties for string or boolean data can also be modified to fit your requirements.

2.3.5.6.9. Modifying a Column to Show Bar Codes

You can also represent the data (string and numeric) in a column as bar codes. This is helpful as most data sources do not have the capacity to store bar codes. And as such, only the information imprinted on the bar codes is saved, either as string or numeric data. This data can be added to the report as a column and then the data format modified to show bar codes. The format can be set for individual `ReportCell` objects or for `ReportColumn` objects. The bar code symbologies supported are Code 39, UPC A, EAN 13, Interleaved 2 of 5 and Codabar.

For example, the code below:

```
BarcodeFormat barCode = new BarcodeFormat(BarcodeFormat.UPCA);
table.getColumn(0).setDataFormat(barCode);
```

encodes the data in Column 0 as bar codes using the UPC A symbology.

2.3.5.6.10. Modifying a Report to be a Top N report

You can modify the report to show a set of the highest values or the lowest values within the group. This is helpful when you want to show a report with the highest revenues or the lowest incidences of errors.

The number of records as well as the ordering is specified using the following method:

```
QbReport.createTopNReport(int colIndex, int topN, boolean ascending);
```

The `colIndex` argument refers to the column in the report, the `topN` argument specifies the number of records and the ascending argument specifies whether to show the highest or the lowest records.

For example, the code below:

```
report.createTopNReport(3, 20, true);
```

takes the highest 20 records within Column 3 (depending on the grouping of the report) and displays them in order.

Note that only Columnar, Summary Break, and Master & Details reports can be modified to a Top N report.

2.3.5.6.11. Cell Scripts

You can also develop your custom cell scripts. These scripts can be assigned to the cells or columns and can be run when certain conditions or requirements are met. These scripts can change the format of the cell(s) so that it looks different from its surrounding cells. Please note that cell scripting using the API is different from cell scripting in Report Designer.

The following example, which can be run as an application or as an applet, sets up a script that changes the font of any data that is less than 0:

```
public class myScript implements ICellScript  {

    // Format the cells according to the specified parameters
    public ReportCell formatCell(int rowIndex, ReportCell cell, Object
 originalData, IFormat dataFormat) throws Exception {
        if (originalData instanceof Double)
            if (((Double)originalData).intValue() < 0)
                cell.setFontColor(java.awt.Color.red);

        return cell;

    }

}
```

You can use the following method in `ReportColumn` to apply the script to a particular column:

```
setCellScript(ICellScript script);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/CellScript.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/CellScript.pdf]

Note that you can have different scripts on different columns although a column can have only one script.

2.3.5.7. Exporting the Report

EspressReport API has the capability to export reports in a variety of formats. These include PDF, DHTML, TXT, and CSV formats etc. In addition, any charts included in the report can be exported into JPEG, GIF, and PNG formats as well; although by default, the charts in the report are exported as JPEGs. The format for the chart can be changed, when creating the `ReportChartObject` object, by specifying the image type (use the method `setImageType(int)`).

A report may also be exported to the proprietary PAK format. An PAK file stores all information, except actual data. The PAK file can then be used to construct a report object. For an PAK file, the data is automatically loaded from the original data source at the time the report object is constructed. (Note PAK files can be directly viewed using a Report Viewer applet.)

To export the report, use the following method:

```
public export(int format, String filename)
```

In the above method, format is one of the format constants listed below and filename is the output filename (with or without an extension).

The following list shows the format constants available for exporting the report components:

QbReport.CSV	Comma delimited text file
QbReport.TXT	Delimited text file
QbReport.DHTML	Dynamic Hyper Text Markup Language (DHTML)
QbReport.HTML	Hyper Text Markup Language (HTML)
QbReport.PDF	Portable Document Format, with password protection option (PDF)
QbReport.RPT	Report File Format (RPT)
QbReport.RPT_DATA	Report File Format (RPT) With Data
QbReport.PAK	Pack Format (PAK)
QbReport.XML_DATA_AND_-FORMAT	Extensible Markup Language, Data and Report Template (XML)
QbReport.XML_PURE_DATA	Extensible Markup Language, Data Only (XML)
QbReport.XML_TEMPLATE	Extensible Markup Language, Report Template Only (XML)
QbReport.EXCEL	Excel Format (XLS)
QbReport.EXCEL_OOXML	xcel 2007 Format (XLSX)
QbReport.RTF	Rich Text Format (RTF)
QbReport.VIEW	View File (VIEW)

In addition, any charts included in the report can be exported to one of the following image formats (format given with its corresponding constant):

QbReport.GIF	GIF
QbReport.JPEG	JPEG
QbReport.PNG	PNG

The following code, which can be run as an applet or application, shows how to construct and export a report:

```
Component doExportReport(Object parent) {
 QbReport.setEspressManagerUsed(false);
 ColInfo colInfo[] = new ColInfo[4];
 for (int i = 0; i < colInfo.length; i++) {
  //   Map data column to the Report column
  colInfo[i] = new ColInfo(i);
 }

 QbReport report = null;
 try {
  report = new QbReport
    (parent, // applet
      "ExportReport.rpt"); // template
   ReportChartObject chartObject = new ReportChartObject();
  String chartLocation0 = new String("ExportReport0.tpl");
  chartObject.setText(chartLocation0);
  chartObject.setWidth(7);
  chartObject.setHeight(3);
  chartObject.setImageType(QbReport.JPEG);

  report.getReportFooter().addData(chartObject);

  report.export(QbReport.DHTML, "ExportReport");

 } catch (Exception ex) {
  System.out.println("Cannot create the report");
  ex.printStackTrace();
 }

 return (new Viewer().getComponent(report));
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ExportReport.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/ExportReport.html]

Please note that when you export the report to a text file, the default delimiter is a tab space. However, you can set another delimiter (available delimiters are “,”, “;” and “ ”) by using the following code:

```
report.setExportDelimiter(IDelimiterConstants.COMMA);    // where report is
 an Object of type QbReport
```

When exporting the report to an Excel or Excel 2007 file, you can also set a method so that each numeric value only occupies one cell. This may be helpful if the end user intends to use Excel functions on the exported report. To use this feature, set the following method to true before exporting the report.

```
QbReport.setExcelExportFitCell(boolean b)
```

You can also export the `QbReport` object to a byte array using the method

```
public byte[] exportReportToByteArray();
```

This will give the .rpt equivalent in the form of a byte array. This is useful if you need to serialize the QbReport object. Please note that you will still need to specify the directories for any sub-reports, drill-down, and/or charts (if you are not using EspressManager) when recreating the QbReport object from the byte array.

2.3.5.7.1. Multiple Page Exporting

You can also export the report into several “pages”, instead of a single “page” i.e., the report can be exported to various files and then shown piecemeal. This option is only available for HTML, DHTML, and XML (Data + Format) export.

The following code is used to export the report into multiple files:

```
report.setExportToMultiPages(true);
```

When using this option, several files are generated (the number of files generated is equal to the number of pages in the report). The first file has the same name as the filename specified in the export method. The subsequent files have the corresponding page number attached at the end of the filename, with the file containing the last page having “LAST” in the filename instead of the page number. For example, if a three page report is exported to SalesSummary. The resulting files that are created are SalesSummary.html (contains the first page), SalesSummary_2.html (contains the second page), and SalesSummary_LAST.html (contains the last page).

You can also export a specific page at a time (instead of the complete report) and export the page to a specific file. This is done using the method:

```
export(int format, OutputStream out, int pageNumber);
```

For example, to export page 3 of a report into the html file ThirdPage, the following code is used:

```
FileOutpuStream dout3 = new FileOutputStream("ThirdPage.html");
report.export(QbReport.DHTML, dout3, 3);
dout3.close();
```

Note that you can pass in “-1” (without the double quotes) as the argument for the pageNumber to export the last page. You also do not need to setExportToMultiPages to true in order to export a specific page.

2.3.5.7.2. PDF Exporting Options

In certain constructors of the QbReport class, the parameters userPass, ownerPass, and permissions are available. These parameters are used to set the user password, owner password, and permissions for the user of the PDF document, respectively. All permissions will be granted to the owner of the PDF document (who uses the owner password to open the PDF document) while only the specified permissions in the permissions argument will be available to the user (who uses the user password to open the PDF document). Please refer to the EspressReport Java API Documentation for more detail about exporting to PDF with options.

The export method that sets PDF options is listed here for the reader's convenience.

```
public void export(int format,
                   java.io.OutputStream out,
                   int exportPage,
                   java.lang.String userPass,
                   java.lang.String ownerPass,
                   int permissions);
```

Additionally, you can also pass embed java scripts in the PDF export so that when the PDF content is streamed to a client browser, the java script is run. The following method allows java script to be embedded to a PDF content

```
public void export(int format,
                   java.io.OutputStream out, // Or java.lang.String
 specifying the file Name
                   java.lang.String userPass,
                   java.lang.String ownerPass,
                   int permissions,
                   java.lang.String javaScript);
```

For example, the following method sets the PDF to automatically print when it is viewed in a client browser.

```
QbReport report = new QbReport(......);
.......
.......
String javaScript = "this.print(true);\r";
report.export(QbReport.PDF, someOutputStream, null, null, QbReport.AllowAll,
 javaScript);
```

The above call would export the report as PDF content (with all permissions) to a stream (`someOutputStream`) passing data to a client browser. When the PDF is loaded on the client browser, the javascript is automatically run (which in this case, tells the browser to print the content).

2.3.5.7.2.1. PDF Font Mapping in API

In order to render Unicode in exported PDF, you need to do font mapping. In Report Designer, it can be done from the Font Mapping dialog: https://data.quadbase.com/Docs71/er/help/manual/SavingandExportingReports.html#DesignerPDFFontMapping

In API, the font mapping method is:

```
void setFontMapping(String fontName, int style, String ttf, String encoding,
 boolean embed)
```

Parameters:

fontName: the name of the font

style: one of the following. QbReport.PLAIN, BOLD, ITALIC, BOLDITALIC

ttf: the path of the .ttf file

encoding(CMAP): the encoding of this font

embed: whether to embed this font to the document. by default true.

> **Note**
>
> Embed is always true for encoding “Identity-H” and “Identity-V”, even if you specify embed to false. Identity-H: horizontal writing systems, Identity-V: vertical writing systems

The following example maps “Dialog” to font file NotoSansSC-Regular.ttf:

```
qbreport.setFontMapping("Dialog",QbReport.BOLD, "C:\\tmp\\testedFonts\
\NotoSansSC-Regular.ttf", BaseFont.IDENTITY_H, true);
```

If you prefer a smaller file size, CJK fonts don’t need to be embedded. In directory ReportDesigner.jar\quadbase\common\util\output\pdf\regularfonts\cmaps\, there are a series of *.properties and *.cmap files. The file cjk_registry.properties states registry and CMAP(encoding) names, for example:

```
Adobe_GB1=Adobe-GB1-0 Adobe-GB1-1 Adobe-GB1-2 Adobe-GB1-3 Adobe-GB1-4 Adobe-
GB1-5 GB-EUC-H GB-EUC-V GB-H GB-V GBK-EUC-H GBK-EUC-V GBK2K-H GBK2K-V GBKp-
```

```
EUC-H GBKp-EUC-V GBpc-EUC-H GBpc-EUC-V GBT-EUC-H GBT-EUC-V GBT-H GBT-V
 GBTpc-EUC-H GBTpc-EUC-V UniGB-UCS2-H UniGB-UCS2-V UniGB-UTF16-H
```

And other properties files are font information, you can find font registry there, for example, in file STSongStd-Light.properties:

Registry=Adobe_GB1

Table: CJK fonts

Language	Font	CMap(encoding) name
Chinese (simplified)	STSong-Light STSongStd-Light	Adobe-GB1-0 UniGB-UCS2-H UniGB-UCS2-V and more ……
Chinese (traditional)	MHei-Medium MSung-Light MSungStd-Light	Adobe-CNS1-0 UniCNS-UCS2-H UniCNS-USC2-V and more ……
Japanese	HeiseiMin-W3 HeiseiKakuGo-W5 KozMinPro-Regular	90msp-RKSJ-H 90pv-RKSJ-H UniJIS-UCS2-H UniJIS-UCS2-V UniJIS-UCS2-HW-H UniJIS-UCS2-HW-V and more ……
Korean	HYGoThic-Medium HYSMyeongJo-Medium HYSMyeongJoStd-Medium	Adobe-Korea1-2 UniKS-UCS2-H UniKS-UCS2-V and more ……

Note

If you open a file using CJK fonts in Adobe Reader, and those fonts are missing, a dialog will pop up asking if you want to update Adobe Reader. If you say yes, it will download and install the necessary fonts. These fonts can only be used in Adobe Reader and nowhere else, unless you have a special license from Adobe.

The following example is a font mapping to a CJK font "STSong-Light":

```
qbreport.setFontMapping("Dialog",QbReport.PLAIN, "STSong-Light", "Adobe-
GB1-0", false);
```

If you want to use a CJKFont in a different style, you can add one of the following modifiers to the font name: Bold, Italic, or BoldItalic. For example:

```
qbreport.setFontMapping("Dialog",QbReport.BOLD, "STSong-Light,Bold", "Adobe-
GB1-0", false);
```

The link Example [https://data.quadbase.com/Docs71/help/manual/code/src/PdfFontMappingExampleER.zip] contains a complete API example with Chinese Font mapping, Japanese Font mapping, and Korean Font mapping.

2.3.5.7.3. Exporting DHTML Content with Style Sheets

You can specify a style sheet to be used while exporting the report to DHTML or HTML format. You can specify an internal style sheet or an external style sheet before exporting the report.

To export the report using an internal style sheet, you would use the following method in `QbReport`:

```
public void setUseStyleSheet(boolean state);
```

To export the report using an external style sheet, you first need to specify the style to be used for the specified `ReportElement` object by using the following method in `ReportElement`:

```
public void setStyleName(String newStyleName);
```

You then specify the name of the external style sheet file using the following method in `QbReport`:

```
public void setExternalStyleSheetName(String css);
```

For example, the following code:

```
report.getTable().getColumn(0).setStyleName("style_1"); // where report is
 an object of type QbReport
report.setExternalStleSheetName("http://someMachine/someDirectory/
styles.css");
```

will set the first column of the report to use `style_1` from the specified style sheet file (in this case, it is `http://someMachine/someDirectory/styles.css`).

2.3.5.7.4. HTML Block Export

You can export reports as a block of HTML code rather than a complete HTML file. This allows for custom content to be added to the generated HTML report.

The code given below:

```
report.setHeadTagIncluded(false);        // where report is an object of
 type QbReport
```

generates the report as an DHTML table when the report is exported. This block can then be included in another page with more content than just the report.

2.3.5.7.5. Custom Links for Multi-page DHTML/HTML Export

You can also create your own links at the top of the page or remove links generated at the top of the pages for single page and multi page exports. This allows custom links to be added to both HTML and DHTML reports.

Custom links can be generated by implementing IHTMLLinksProvider [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/reportdesigner/util/IHTMLLinksProvider.html]. interface. For example, the code below, which can be run as an application or an applet, demonstrates creating custom links:

```
Component doCustomLinks(Object parent) {
 QbReport.setEspressManagerUsed(false);
 QbReport report = new QbReport(parent, "CustomLinks.rpt");
 report.setHTMLLinksProvider(this);
 QbReport.setExportToMultiPages(true);

 try
 {
  report.export(IExportConstants.DHTML, "CustomLinks.html");
 } catch (Exception ex) {
  ex.printStackTrace();
 }
 return (new Viewer().getComponent(report));

}

public HTMLBlock getLinksForDHTML(int cPage, int tPage, String
 prefix, boolean top)
{

 String linksText = "<CENTER><font face=\\\"verdana\\\" size=\\\"2\\\">" +

   "<a href=\\\"" + HTMLBlock.getFirstFileName(prefix) + "\\\">FIRST</a> |
 ";
 if (cPage <= 2) {
  linksText += "<a href=\\\"" + HTMLBlock.getFirstFileName(prefix) + "\\
\">PREV</a> | "
    + "<a href=\\\"" + HTMLBlock.getNextFileName(cPage, tPage, prefix)
    + "\\\">NEXT</a> | " + "<a href=\\\"" +
 HTMLBlock.getLastFileName(prefix)
    + "\\\">LAST</a> [" + "<a href=\\\""
    + HTMLBlock.getCurrentFileName(cPage, tPage, prefix)
    + "?x=x\\\" target=\\\"print\\\">Print Version</a>]" + "</font></
CENTER>";
 } else {
  linksText += "<a href=\\\"" + HTMLBlock.getPreviousFileName(cPage, tPage,
 prefix)
    + "\\\">PREV</a> | " + "<a href=\\\""
    + HTMLBlock.getNextFileName(cPage, tPage, prefix) + "\\\">NEXT</a> | "
    + "<a href=\\\"" + HTMLBlock.getLastFileName(prefix) + "\\\">LAST</a> ["
    + "<a href=\\\"" + HTMLBlock.getCurrentFileName(cPage, tPage, prefix)
    + "?x=x\\\" target=\\\"print\\\">Print Version</a>]" + "</font></
CENTER>";
 }

 String text = "<SCRIPT>\n" + "document.write(\"" + linksText + "\");\n"
 + "</SCRIPT>";
 return new HTMLBlock(text, 100);

}

public HTMLBlock getLinksForHTML(int cPage, int tPage, String
 prefix, boolean top)
{
 return getLinksForDHTML(cPage, tPage, prefix, top);
}

public HTMLBlock getTOCLinkForDHTML(int cPage, int tPage, String body) {
 return null;
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/CustomLinks.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/CustomLinks.html]

Similarly other links can be generated and set to the report.

2.3.5.7.6. Setting the Pixels Per Inch Ratio for DHTML/HTML Export

EspressReport allows a pixels per inch ratio to be specified while exporting HTML or DHTML content. This is especially useful when you are moving and deploying reports between platforms or when exporting reports in a system without any graphics. You can specify the ratio using the following method in `QbReport`:

```
public void setPixelPerInchForExport(int pixelPerInchRatio);
```

2.3.5.7.7. Pre-load charts

EspressReport allows charts with independent data sources to be pre-loaded before the report is exported. The preloading is simultaneous, so it improves export performance (especially if there are multiple charts in the report). You can specify to pre-load the charts by calling the following method before exporting the report:

```
public void preloadChartObjects();
```

2.3.5.7.8. IExportThreadListener for .view files

Reports can be exported to .view files. These files are used by Page Viewer to show the report a few pages at a time (as opposed to loading the entire report in Report Viewer). Exporting to .view files is similiar to exporting to multiple pages in that the one export method results in multiple pages (the more the number of pages in the report, the more files that are generated). However, Page Viewer cannot be started unless the .view file is generated (the pages of the report are generated as .page files). EspressReport provides an interface, IExportThreadListener [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/reportdesigner/util/IExportThreadListener.html], which you can implement to perform actions when the first page is exported and when the export is completed.

Given below is an example using the IExportThreadListener:

```
public class ExportView extends Applet implements IExportThreadListener {
 static final long serialVersionUID = 1;

 public static void main(java.lang.String[] args) {

  try {
   ExportView report = new ExportView();
   report.exportView(null);
  } catch (Exception ex) {
   ex.printStackTrace();
  }

 }

 // creates report and return it
 void exportView(Object parent) throws Exception {

  //  Turn off ReportServer as it is not needed.
  QbReport.setEspressManagerUsed(false);

  //  Create the colinfo array to be used in the QbReport constructor
  QbReport report = new QbReport(parent, "ExportView.rpt");

  report.setMultiPageExp(true);
  report.export(QbReport.VIEW, "ExportView", new Properties(), this);
  System.out.println("DONE!");

 }

 public void endAction() {
  System.out.println("END ACTION");
 }

 public void firstPageFinishedAction() {
  System.out.println("FIRST PAGE FINISHED ACTION");
  quadbase.reportdesigner.PageViewer.Viewer.main(new String[]
 { "ExportView.view" });
 }
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ExportView.zip]

2.3.5.7.9. Virtual Memory / Paging

In addition to memory optimized exporting, EspressReport also has a paging feature to handle large amounts of data. With paging, you specify the amount of memory and a temp directory. When the amount of memory used exceeds the number specified, the data is compacted and then stored on disk in the temp directory specified.

The feature works with or without EspressManager. However, if using the EspressManager, it is recommended to use the same option values in both the client code and the EspressManager.

To use memory paging export, the following code must be added before calling the QbReport constructor:

```
QbReport.setTempDirectory
          (<specify the temp directory to store data.  Default is ./temp>);
QbReport.setMaxFieldSize(int fieldSize);
QbReport.setPagingThreshold(int pagingThreshold);
QbReport.setPageBufferSize(int bufferSize);
QbReport.setTotalPageBufferSize(int totalBufferSize);
```

Max Field Size:

This argument specifies the expected maximum field size for large field types such as varchars. Fields that are larger than this value are **not** necessarily truncated. The behavior depends on the current memory usage in relation to the Total Paging Buffer Size allowed. As the amount of available memory decreases the amount of data stored for these large fields will decrease until it reaches this value. For example, a user is running a report with a field size that is larger than the specified Max Field Size value. When the server is not busy and there is ample memory available, the report will generate the full field without any truncation. However, when the server is heavily loaded and the available memory is near zero, the field in the report will be truncated to the Max Field Size. For numeric, boolean, date, time, and character data types, the data will never be truncated. For those database fields that are defined with a size limit smaller than the Max Field Size (e.g. Max Field Size = 500, but the field is defined as varchar(20)), the database limit will be used in place of the Max Field Size. This value is set in Bytes.

Paging Threshold:

This property specifies when the paging feature will be activated. Once this threshold is reached, the server will begin paging the data to temporary files on the system. If this argument is set to -1 then data will never be paged. This value is independent of Page Buffer Size option. If this value is larger than the Page Buffer Size, the system will not begin paging until the threshold is reached. Therefore, each report or chart might use more memory than the amount specified in the Page Buffer Size. This value is set in Megabytes and the default is -1.

Paging Buffer Size:

This argument allows you to set the amount of memory that each report or chart will use when the paging feature is invoked. The size of the buffer affects performance. The larger the buffer size, the faster the report or chart is generated, but the more memory is used. When the amount of total memory used by the paging system approaches the `Total Paging Buffer Size`, the amount of memory provided to new reports will begin to diminish in order to avoid exceeding the specified total amount. This value is set in Megabytes.

Total Paging Buffer Size:

This argument allows you to set the total amount of memory used by the paging feature across all reports and charts. The memory allocated to each report will diminish as the memory usage approaches this value. This value is set in Megabytes.

2.3.5.7.10. Streaming Reports

In addition to exporting to local drives, reports can also be exported as a byte stream and streamed directly to a web browser. However, in order for charts, drill-downs, and parameters to function correctly, this export method requires that several support servlets be available. The three servlets that does this are `RPTImageGenerator`, `DrillDownReportServlet`, and `ParamReportGeneratorServlet`. They are located in the `<EspressReport>/WEB-INF/classes/` directory. The three files must be copied to the servlet directory of your servlet runner. Use the following code to connect to the servlets:

```
report.setDynamicExport(true, "Machine Name or IP Address", Port);
report.setServletDirectory("Servlet Directory");
```

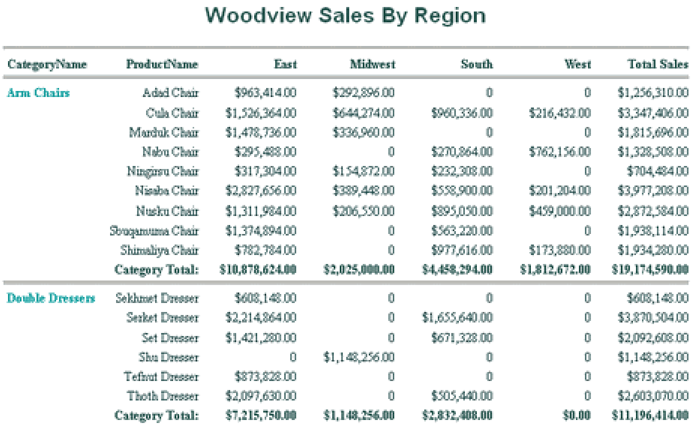
These methods sees to it that any call to these servlets are pointing to the correct machine and port. The servlets get called using the URL, `http://<MachineName:>:<Port:>/<ServletDirectory:>/<Servlet:>`. You must set dynamic export when streaming your report, but setting the servlet directory is optional. If you do not specify which servlet directory to use, it will be automatically set to `servlet/`.

Given below is an example that exports a report to DHTML and streams it to the browser. In order to run the example, you will need to configure and compile the source code, then deploy the class file in the servlet directory of your servlet runner. Replace the `reportTemplate` variable with either an absolute path or a path relative to the working directory of your application server. Remember that the working directory is usually not the same as the servlet directory. In addition, make sure to add `ReportAPIWithChart.jar` and `qblicense.jar` to the classpath of your application server. Both jar files can be found in `<EspressReport>/lib/`.

```
public class StreamingReport extends HttpServlet {
 static final long serialVersionUID = 1;

 @Override
 public void doGet(HttpServletRequest req, HttpServletResponse res) throws
 ServletException,
   IOException
 {
  // Do not use EspressManager
  QbReport.setEspressManagerUsed(false);

  // Location of report template
  String reportTemplate = "StreamingReport.rpt";

  // Create QbReport object
  QbReport report = new QbReport(null, reportTemplate);

  // Set up connection information for RPTImageGenerator and
 DrillDownReportServlet
  report.setDynamicExport(true, "localhost", 8080);
  report.setServletDirectory("servlet/");

  // Export the report to DHTML and stream the bytes
  ByteArrayOutputStream reportBytes = new ByteArrayOutputStream();

  try {
   report.export(QbReport.DHTML, reportBytes); // where report is an object
 of type QbReport
   //   report.export(QbReport.PDF, reportBytes);   // where report is an
 object of type QbReport
  } catch (Exception ex)
  {
   ex.printStackTrace();
  }

  res.setContentType("text/html"); // where res is the HttpServletResponse
  //  res.setContentType("application/pdf");  // where res is the
 HttpServletResponse
  res.setContentLength(reportBytes.size());

  OutputStream toClient = res.getOutputStream();
  reportBytes.writeTo(toClient);
  toClient.flush();
  toClient.close();
 }
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/StreamingReport.zip]

To run the example, make sure to copy the report templates to the location accessed by the code. If using a relative path (as shown in the example), remember that the current directory is the working directory of your application server. For example, since the working directory in Tomcat is `<Tomcat>/bin`, you will need to create the directory `<Tomcat>/templates/` and copy the report templates there to run the code. The resulting report is shown below:



Woodview Sales By Region

CategoryName	ProductName	East	Midwest	South	West	Total Sales
Arm Chairs	Adad Chair	$963,414.00	$292,896.00	0	0	$1,256,310.00
	Cula Chair	$1,526,364.00	$644,274.00	$960,336.00	$216,432.00	$3,347,406.00
	Marduk Chair	$1,478,736.00	$336,960.00	0	0	$1,815,696.00
	Nabu Chair	$295,488.00	0	$270,864.00	$762,156.00	$1,328,508.00
	Ningirsu Chair	$317,304.00	$154,872.00	$232,308.00	0	$704,484.00
	Nisaba Chair	$2,827,656.00	$389,448.00	$558,900.00	$201,204.00	$3,977,208.00
	Nusku Chair	$1,311,984.00	$206,550.00	$895,050.00	$459,000.00	$2,872,584.00
	Sbuqamuma Chair	$1,374,894.00	0	$563,220.00	0	$1,938,114.00
	Shimaliya Chair	$782,784.00	0	$977,616.00	$173,880.00	$1,934,280.00
	Category Total:	**$10,878,624.00**	**$2,025,000.00**	**$4,458,294.00**	**$1,812,672.00**	**$19,174,590.00**
Double Dressers	Sekhmet Dresser	$608,148.00	0	0	0	$608,148.00
	Serket Dresser	$2,214,864.00	0	$1,655,640.00	0	$3,870,504.00
	Set Dresser	$1,421,280.00	0	$671,328.00	0	$2,092,608.00
	Shu Dresser	0	$1,148,256.00	0	0	$1,148,256.00
	Tefnut Dresser	$873,828.00	0	0	0	$873,828.00
	Thoth Dresser	$2,097,630.00	0	$505,440.00	0	$2,603,070.00
	Category Total:	**$7,215,750.00**	**$1,148,256.00**	**$2,832,408.00**	**$0.00**	**$11,196,414.00**

Streaming Report

2.3.5.7.10.1. RPTImageGenerator

The `RPTImageGenerator` is used when you export reports containing charts or BLOB images to either HTML or DHTML format. HTML and DHTML files are purely textual; therefore, charts and images must be stored separately. When you export a report locally, the chart is stored as an image file and the file path is appended to the IMG tag of the HTML page.

```
<IMG SRC="stream_0.jpg" ALIGN="CENTER" VALIGN="TOP"BORDER="0" HEIGHT="288"
 WIDTH="576"></IMG>
```

Similarly, when you stream the DHTML file, only the textual component is sent to the browser. Charts are handled in the `RPTImageGenerator` servlet. Instead of pointing the IMG tag to the image file, the tag is set to the `RPTImageGenerator` servlet and appends the ID of the chart image as a parameters.

```
<IMG SRC="http://localhost:8080/servlet/RPTImageGenerator?
ID=1166125348416_0" ALIGN="CENTER" VALIGN="TOP"BORDER="0" HEIGHT="288"
 WIDTH="576"></IMG>
```

Reports containing images that are stored locally do not require the servlet. The IMG source will point directly to the image file. However, images retrieved from the datasource (BLOBs) are treated similar to charts and does require the aid of the `RPTImageGenerator` servlet.

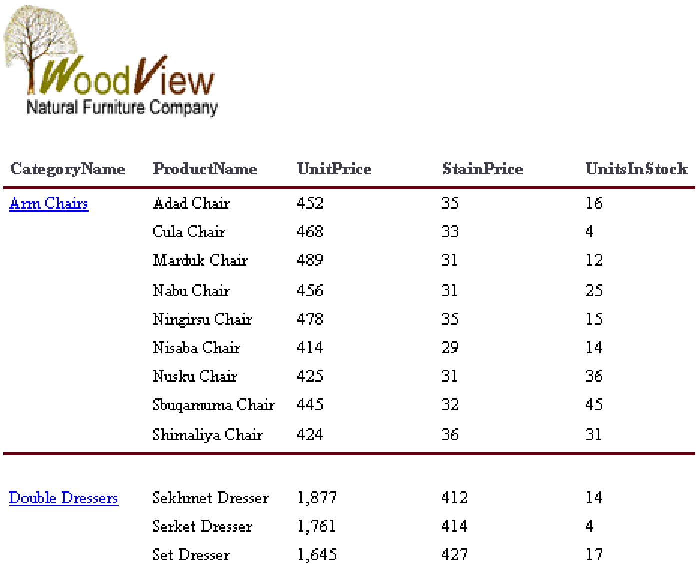
2.3.5.7.10.2. DrillDownReportServlet

Regardless of if you are exporting the report to a byte stream or to a file, reports with drill-downs require that the `DrillDownReportServlet` be accessible in order to function properly. When you export drill-down reports to either DHTML or PDF format, you must have `DrillDownReportServlet` in the servlet directory of your servlet runner. This servlet provides the link to the next level.

```
<a href="http://localhost:8080/servlet/DrillDownReportServlet?
 FILENAME=DrillDown%2Fstream_lvl11.rpt&FORMAT=4&PARAM0=112+Ishtar+Chair"
 title="tt" alt="tt">
Ishtar Chair<BR></a>
```

Since drill-down levels must contain a parameterized query, you need to make sure that the `ParamReportGeneratorServlet` is in the servlet directory as well. More information on the `ParamReportGeneratorServlet` are available in the next several sections.

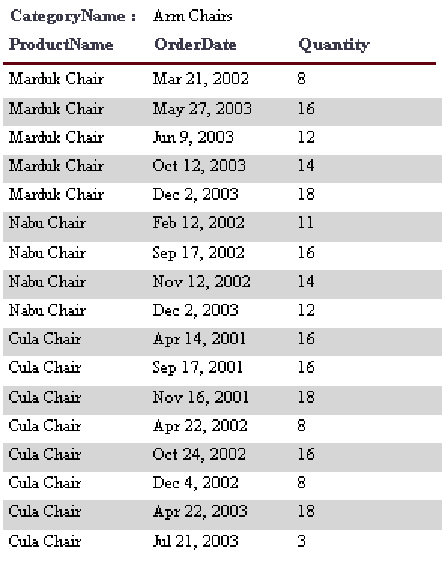
If you export the report to HTML or DHTML, the column linked to the drill-down report will display as a hypertext link. In PDF format, the linked column will not look any different from the others; however, when you hover the mouse over that column, the cursor will change to a hand.



CategoryName	ProductName	UnitPrice	StainPrice	UnitsInStock
Arm Chairs	Adad Chair	452	35	16
	Cula Chair	468	33	4
	Marduk Chair	489	31	12
	Nabu Chair	456	31	25
	Ningirsu Chair	478	35	15
	Nisaba Chair	414	29	14
	Nusku Chair	425	31	36
	Shuqamuna Chair	445	32	45
	Shimaliya Chair	424	36	31
Double Dressers	Sekhmet Dresser	1,877	412	14
	Serket Dresser	1,761	414	4
	Set Dresser	1,645	427	17

Root Level

Depending on the link clicked, the servlet delivers the appropriate next level report.



CategoryName : Arm Chairs

ProductName	OrderDate	Quantity
Marduk Chair	Mar 21, 2002	8
Marduk Chair	May 27, 2003	16
Marduk Chair	Jun 9, 2003	12
Marduk Chair	Oct 12, 2003	14
Marduk Chair	Dec 2, 2003	18
Nabu Chair	Feb 12, 2002	11
Nabu Chair	Sep 17, 2002	16
Nabu Chair	Nov 12, 2002	14
Nabu Chair	Dec 2, 2003	12
Cula Chair	Apr 14, 2001	16
Cula Chair	Sep 17, 2001	16
Cula Chair	Nov 16, 2001	18
Cula Chair	Apr 22, 2002	8
Cula Chair	Oct 24, 2002	16
Cula Chair	Dec 4, 2002	8
Cula Chair	Apr 22, 2003	18
Cula Chair	Jul 21, 2003	3

The Drill-Down Level

Note that drill-down reports exported to formats other than DHTML or PDF will only show the current level.

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/StreamingDrillDown.zip]

2.3.5.7.10.3. Generating HTML Parameter Page

When streaming the report, a powerful alternative to setting parameter values using the API (discussed in Section 2.3.5.1.4 - Parameterized Reports) is to use an automatically generated HTML form that contains text or selection inputs. After the user submits the form, the parameter values are passed to the `ParamReportGeneratorServlet` that processes them and streams back a report initialized with these parameter values.

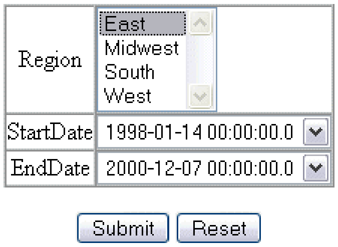
The main classes for this purpose are `ParameterPage`, `ParameterPageWriter`, `HtmlParameterPageWriter`, and `CssHtmlParameterPageWriter`. The typical method of generating the HTML Parameter Page involves the following code:

```
OutputStream out = ...; //response.getOutputStream();
ParameterPage paramPage = QbReport.getParameterPage();
Writer writer = new PrintWriter(out);
HtmlParameterPageWriter paramPageWriter = new
 HtmlParameterPageWriter(paramPage, writer);
paramPageWriter.writePage();
writer.flush();
writer.close();
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/HTMLParamPage.zip]

When you use the above functionality, you must have the `ParamReportGeneratorServlet` in the servlet directory of your servlet runner. This servlet generates the report in the specified format using the parameters specified in the HTML page.

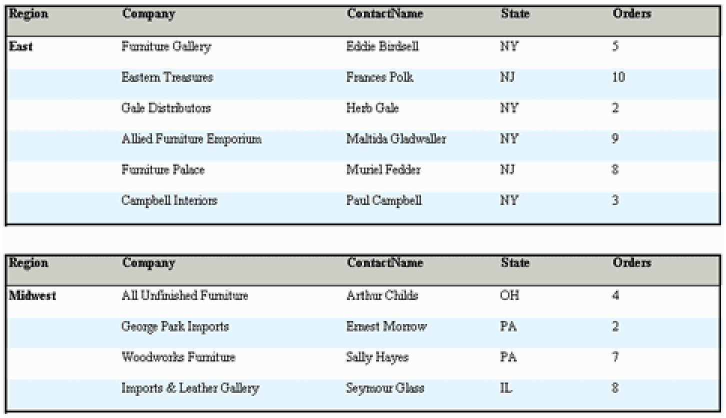
When you run the example, the following page is automatically generated:



Region	East Midwest South West
StartDate	1998-01-14 00:00:00.0
EndDate	2000-12-07 00:00:00.0

Generated Parameter Page

Depending on the values you select, clicking on the submit button may result in the following report:



Region	Company	ContactName	State	Orders
East	Furniture Gallery	Eddie Birdsell	NY	5
	Eastern Treasures	Frances Polk	NJ	10
	Gale Distributors	Herb Gale	NY	2
	Allied Furniture Emporium	Maltida Gladwaller	NY	9
	Furniture Palace	Muriel Fedder	NJ	8
	Campbell Interiors	Paul Campbell	NY	3

Region	Company	ContactName	State	Orders
Midwest	All Unfinished Furniture	Arthur Childs	OH	4
	George Park Imports	Ernest Morrow	PA	2
	Woodworks Furniture	Sally Hayes	PA	7
	Imports & Leather Gallery	Seymour Glass	IL	8

Parameterized Report

Although it is convenient to automatically generate the entire parameter page, there are times when you want more control over the appearance of the page. One way to achieve this is to only generate the main form, allowing you to design the contents around the main form to provide your own look and feel. Another way is to use CSS to set the desired format. A third way is to generate each component within the form individually giving you control over every element on the page. These various methods are discussed in greater detail in the next two sections.

2.3.5.7.10.3.1. Utilizing Cascading Style Sheets (CSS)

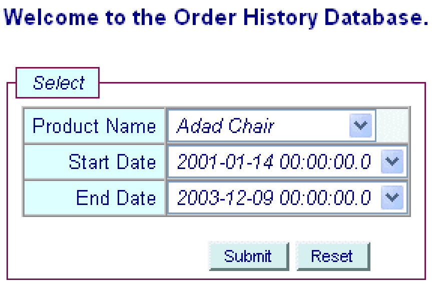
Two powerful ways to alter the HTML parameter page are to add content around the form and to utilize cascading style sheets. These methods allow you to maintain a consistent look and feel to your web application, without the necessity to write an excessive amount of code.

The example below shows how these methods can be used to present the parameter page using your predefined colors, fonts, style, and format. In order to run this example, you need to modify the template path. Keep in mind that if you use a relative path, it will be relative to the working directory of your application server. You will also be required to copy the CSS file to the root directory of your web application so that it can be accessible by the CSS link in the code (alternatively, you can position the CSS file elsewhere and modify the link). The examples require that both `ParamReportGeneratorServlet` and `DrillDownReportServlet` be available in the servlet directory since the example includes a Drill-Down report.

```
CssHtmlParameterPageWriter cssParamPageWriter =
                    new CssHtmlParameterPageWriter(paramPage,
 tempStringWriter);

// Specify the css file to be used
cssParamPageWriter.setCssFile("http://localhost:8080/CssExample.css");

// Create head of html page, adding title and css specification
cssParamPageWriter.write("<HEAD>\n" +
                 "<link REL=\"stylesheet\" TYPE=\"text/css\" " +
                 "href=\"" + cssParamPageWriter.getCssFile() + "\">" +
                 "\n<TITLE>Welcome to Woodview</TITLE>\n" +
                 "</HEAD>");

// Body section - Most of the formatting is done in the css file,
// very little is needed here
cssParamPageWriter.write("\n<body><p>
");

// Add title and fieldset
cssParamPageWriter.write("<TABLE class=\"outer\"><TR><TD>" +
                 "<p class=\"heading2\">Welcome to the Order History
 Database." +
                 "</TD></TR><TR><TD>&nbsp;</TD></TR>" +
                 "<TR><TD><FIELDSET><LEGEND>Select</LEGEND>");
// Add the parameter forms
cssParamPageWriter.writeBodyBody();

// Finish it off
cssParamPageWriter.write("</FIELDSET></TD></TR>" +
                 "</TABLE></body>\n</html>");
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/CSSParamPage.zip]

In the above segment of code, the <HEAD> is written manually so that a title can be inserted. In the body section, only the form is generated. Using this method, users can add titles, images, and other components to maintain their own look and feel. Depending on the CSS file provided, the result might look like this:



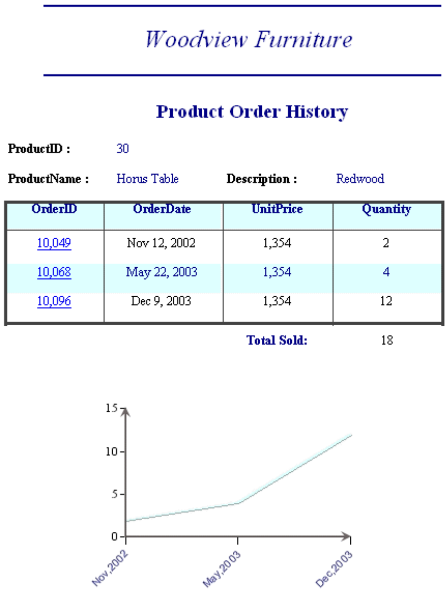
Welcome to the Order History Database.

Select

Product Name	Adad Chair
Start Date	2001-01-14 00:00:00.0
End Date	2003-12-09 00:00:00.0

Parameter Page using CSS

Depending on the parameter options you select, the result may look like the following:



Woodview Furniture

Product Order History

ProductID : 30

ProductName : Horus Table   Description : Redwood

OrderID	OrderDate	UnitPrice	Quantity
10,049	Nov 12, 2002	1,354	2
10,068	May 22, 2003	1,354	4
10,096	Dec 9, 2003	1,354	12

Total Sold: 18

Example Results

2.3.5.7.10.3.2. Customizing the Parameter Page

Another way to customize the parameter page is to write a java class that extends HtmlParameterPageWriter [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/common/param/HtmlParameterPageWriter.html] or CssHtmlParameterPageWriter [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/common/param/CssHtmlParameterPageWriter.html].This approach gives you access to the protected methods available in these classes. For information on the full set of methods available in these classes, please see the APIDocs [https://data.quadbase.com/Docs71/er/help/apidocs/index.html]

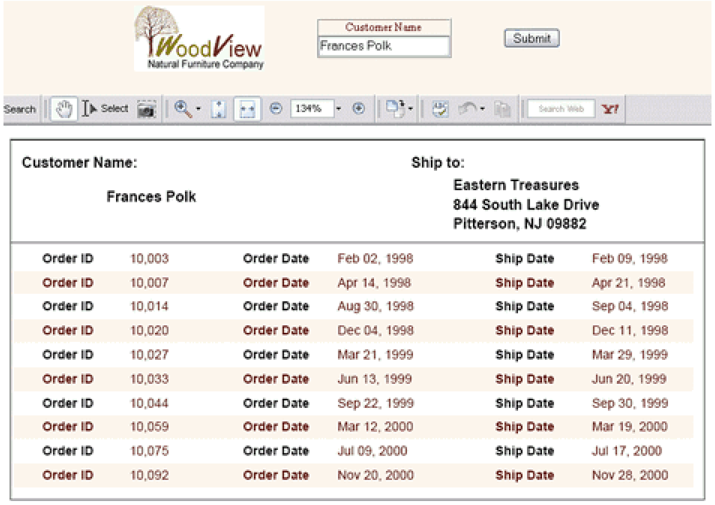
The following is an example extending the `HtmlParameterPageWriter` class. To run this example, you will need to copy the two html pages to the root directory of your application server. The template and java class files must be copied to the working directory and servlet directory of your application server respectively. The example also requires that `ParamReportGeneratorServlet` and `DrillDownReportServlet` be placed in the servlet directory.

```
private class MyHtmlParameterPageWriter extends HtmlParameterPageWriter{

    public MyHtmlParameterPageWriter(ParameterPage pp, StringWriter sw) {
        super(pp,sw); }

    public void printCustom() {
        try {
            write("<BODY bgcolor=#FDFAED>");
            write("<FORM action = \"" + servletName +"\" target=\"main\"
 method = GET>\n");
            write("<TABLE ALIGN=\"CENTER\"><TR><TD><IMG src=http://
www.quadbase.com/FurnitureImages/Woodview.gif></TD>");
            write("<TD WIDTH=50></TD><TD>");
            writeParamTable();

            write("</TD><TD WIDTH=50></TD><TD>");
            writeSubmitButton();
            write("</TD>");
            write("</TR></TABLE>");

            writeHiddenParamValue("ReportFilePath", reportLocation);
            write("\n");

            writeHiddenParamValue("ReportExportFormat", ""+format);
            write("\n");

            writeHiddenParamValue("ServerName", servletAddress);
            write("\n");
            writeHiddenParamValue("ServletRunnerPort", ""+servletPort);

            write("\n");
            writeHiddenParamValue("ServletDirectory", servletDirectory);

            write("\n</FORM>");
            html.bodyEnd();
            html.htmlEnd(); } catch (Exception e) {
            e.printStackTrace(); } }

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/CustomParamPage.zip]

Notice that by using the protected methods, you are required to add a number of details in your code, such as FORM tags and hidden parameters. However, this also grants you the ability to add other elements such as company logos into the form section of the page and arrange the elements to your preference. This approach also allows you to eliminate certain elements from the page such as the *reset* button.

Once the files have been copied to the correct locations, you can run the example by entering the following URL in your browser: `http://<IP Address/localhost>:<Port>/CustomParamPage.html`. You will see a page with two frames. The top frame is the parameter page and the bottom frame initially displays a simple message. To view the report, enter the full name of a customer (e.g. Francis Polk).



Customer Name: Frances Polk

Customer Name:

Frances Polk

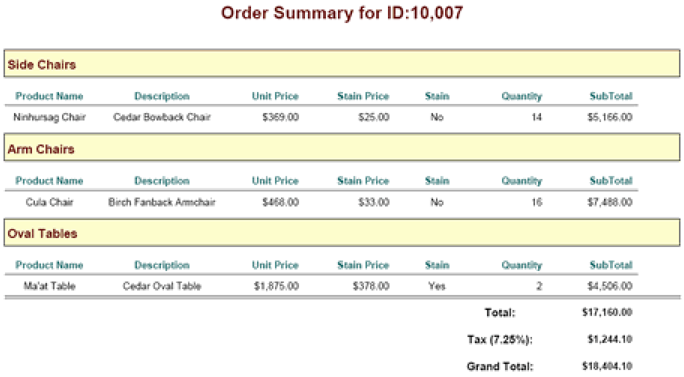
Ship to:

Eastern Treasures
844 South Lake Drive
Pitterson, NJ 09882

Order ID	10,003	Order Date	Feb 02, 1998	Ship Date	Feb 09, 1998
Order ID	10,007	Order Date	Apr 14, 1998	Ship Date	Apr 21, 1998
Order ID	10,014	Order Date	Aug 30, 1998	Ship Date	Sep 04, 1998
Order ID	10,020	Order Date	Dec 04, 1998	Ship Date	Dec 11, 1998
Order ID	10,027	Order Date	Mar 21, 1999	Ship Date	Mar 29, 1999
Order ID	10,033	Order Date	Jun 13, 1999	Ship Date	Jun 20, 1999
Order ID	10,044	Order Date	Sep 22, 1999	Ship Date	Sep 30, 1999
Order ID	10,059	Order Date	Mar 12, 2000	Ship Date	Mar 19, 2000
Order ID	10,075	Order Date	Jul 09, 2000	Ship Date	Jul 17, 2000
Order ID	10,092	Order Date	Nov 20, 2000	Ship Date	Nov 28, 2000

Order History Report

This example exports to PDF instead of DHTML. Although there are no links displayed in PDF format, if you hover the mouse over the Order ID column, you will see that the mouse pointer turns into a hand alerting you that you can drill-down from this column. Depending on the Order ID you selected, the follow report may be shown.



Order Summary for ID:10,007

Side Chairs

Product Name	Description	Unit Price	Stain Price	Stain	Quantity	SubTotal
Ninhursag Chair	Cedar Bowback Chair	$369.00	$25.00	No	14	$5,166.00

Arm Chairs

Product Name	Description	Unit Price	Stain Price	Stain	Quantity	SubTotal
Cula Chair	Birch Fanback Armchair	$468.00	$33.00	No	16	$7,488.00

Oval Tables

Product Name	Description	Unit Price	Stain Price	Stain	Quantity	SubTotal
Ma'at Table	Cedar Oval Table	$1,875.00	$378.00	Yes	2	$4,506.00

Total:	$17,160.00
Tax (7.25%):	$1,244.10
Grand Total:	$18,404.10

Order History Detail (Drill-Down)

2.3.5.7.11. Calling Report Designer from Report API

Report Designer can be called from Report API in either an application or an applet. Depending on the code, you can pass in parameters to open up Report Designer with a specified .rpt file or a specified data source or other parameters.

To call Report Designer from Report API, you must:

make sure that EspressManager is up and running;

add `ReportAPIWithChart.jar`, `EspressManager.jar` and `qblicense.jar` to the `CLASSPATH`;

make sure that the information to connect to EspressManager is specified using the relevant API calls (This is especially important if the EspressManager is on a different machine and/or if it started with a port number other than 22071);

copy the images, reportimages and backgroundimages directories to the working directory or use `QbReportDesigner.setUseSysResourceImages(true)` in your code to use the images from the jar files;

Depending on the `-RequireLogin` and `-QbDesignerPassword` flags for EspressManager, you may need to pass in a userid and/or password to connect to EspressManager. If the `-RequireLogin` flag is set for EspressManager, you need to add the following line of code before calling `setVisible()` or any `getDesigner` methods:

```
public login(String userName, String password); // Method found in
 QbReportDesigner class
```

If the `-QbDesignerPassword` is specified for EspressManager, you will need to add the following line of code before calling `setVisible()` or any `getDesigner` methods:

```
public login(String password); // Method found in QbReportDesigner class
```

You can also specify a look and feel to Report Designer (Report Designer will use the system's look and feel by default). This is done by using the following method in `QbReportDesigner`:

```
public static void setLookAndFeel(javax.swing.LookAndFeel newLookAndFeel);
```

Given below are the different ways Report Designer can be called via Report API. Note that if you are running the example code as an applet, you need to change the EspressManager machine from **`127.0.0.1`** (or **`localhost`**) to the EspressManager machine's name or IP address.

2.3.5.7.11.1. Specify a Report Template

You can open Report Designer with a specified report template file. This lets the end users create their own custom reports in Report Designer GUI and then view the finished report.

The following constructor is used:

```
QbReportDesigner(Object parent, String templateFileName);
```

Given below is an example of calling Report Designer with a specified .rpt file:

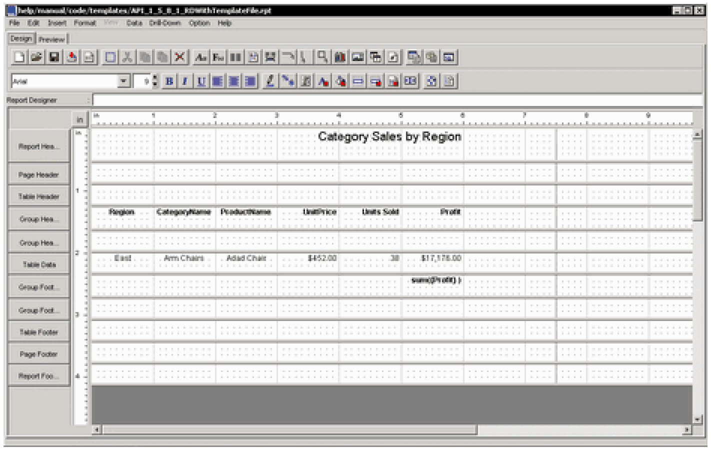
```
public void doReportDesignerApplet(Applet applet) throws Exception {

    // Connect to EspressManager
    QbReportDesigner.setServerAddress("127.0.0.1");
    QbReportDesigner.setServerPortNumber(22071);

    // Use images from jar file
    QbReportDesigner.setUseSysResourceImages(true);

    // Create a new QbReportDesigner instance
    designer = new QbReportDesigner(applet,
           "RDWithTemplateFile.rpt");

    // Overwrites default saveFile method
    designer.setReportIO(this);

    // Start Designer
    designer.setVisible(true);

}

// Save the file to a temp directory
public void saveFile(byte[] data, String fileLocation) {

    System.out.println("OLD LOCATION: " + fileLocation);
    String newLoc = fileLocation.replace('/', '\\');
    int idx = newLoc.lastIndexOf("\\");
    newLoc = "temp/" + newLoc.substring(idx+1);
    try {
        designer.writeFile(newLoc, data);
        System.out.println("NEW LOCATION: " + newLoc);

    } catch (Exception ex) {
        ex.printStackTrace();
    }

    designer.getDesigner().setTitle(newLoc);
    designer.getDesigner().repaint();

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/RDWithTemplateFile.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/RDWithTemplateFileEndUser.pdf]

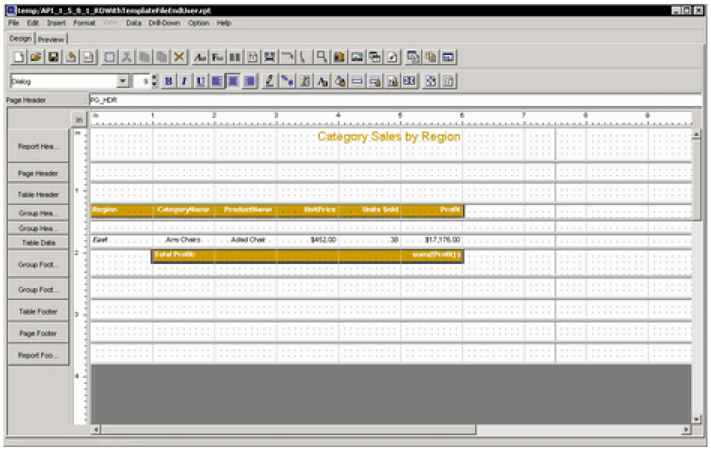
Note that the above code can be used as both an application and an applet. The above code also changes the save functionality in Report Designer so that the .rpt file is always saved in the temp directory.

When the user runs this code, the Report Designer is launched with the report template you specified.


help/manual/code/templates/API_1_5_8_1_RDWithTemplateFile.rpt
File Edit Insert Format View Data Drill-Down Option Help
Design Preview
Report Designer
Report Hea...
Page Header
Table Header
Group Hea...
Group Hea...
Table Data
Group Foot...
Group Foot...
Table Footer
Page Footer
Report Foo...
Category Sales by Region
Region CategoryName ProductName UnitPrice Units Sold Profit
East Arm Chairs Adad Chair $452.00 38 $17,176.00
sum(Profit)

Report Designer with Template

The user can then customize the report and save the results.


temp/API_1_5_8_1_RDWithTemplateFileEndUser.rpt
File Edit Insert Format View Data Drill-Down Option Help
Design Preview
Page Header
PG_HDR
Report Hea...
Page Header
Table Header
Group Hea...
Group Hea...
Table Data
Group Foot...
Group Foot...
Table Footer
Page Footer
Report Foo...
Category Sales by Region
Region CategoryName ProductName UnitPrice Units Sold Profit
East Arm Chairs Adad Chair $452.00 38 $17,176.00
Total Profit: sum(Profit)

End User Customization

You can also specify the registry to use when running the above code. This is achieved by adding in the following line of code before setting the designer visible:

```
designer.setDataRegistry("DataRegistry/Data_Registry.xml");
```

The above line specifies the Report Designer to use `Data_Registry.xml` as the Data Registry when choosing the data sources for a new report.

2.3.5.7.11.2. Specify a Data Registry

You can open Report Designer and have it starting with a Data Source Manager (with a specified .xml file for the Data Registry). This allows the end users to choose the data source and create their own custom reports in Report Designer GUI and then view the finished report.

The following constructor is used:

```
QbReportDesigner(Object parent, String nameOfXMLFile, boolean
 doNotStartWithOpenNewReportDialogWizard);
```

Given below is an example of calling Report Designer with a specified .xml file:

```
public void doReportDesignerWithDataRegApplet(Applet applet) throws
 Exception {

    // Connect to EspressManager
    QbReportDesigner.setServerAddress("127.0.0.1");
    QbReportDesigner.setServerPortNumber(22071);

    // Use images from jar file
    QbReportDesigner.setUseSysResourceImages(true);

    // QbReportDesigner (component, name of XML file, boolean);
    // true = start from Data Source Manager false = start from Create a new
 report
    // or open existing report choice before going to Data Source Manager
    designer = new QbReportDesigner(applet, "DataRegistry/Sample.xml",
 true);

    designer.setVisible(true);

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/RDWithDataRegistry.zip]

Note that the above code can be used as both an application and an applet.

You can also control what the user does at the Data Registry by enabling or disabling the options available. For example, you can remove complete or parts of data sources in the Data Registry and you can disable the options for the adding, copying, editing, and deleting of data sources. Note that the nodes are not hidden, they are removed. Therefore, when the Data Registry is opened (without any restrictions), the content of the nodes that were removed is lost. You can create a backup of the Data Registry .xml file and use the backup to enable/disable options.

Given below is an example of calling Report Designer with a specified data registry and setting it up so that only the database data sources are available and no queries or data views can be added.

```
public void doReportDesignerWithDataRegOptionApplet(Applet applet) throws
 Exception {

    // Connect to EspressManager
    QbReportDesigner.setServerAddress("127.0.0.1");
    QbReportDesigner.setServerPortNumber(22071);

    // Use images from jar file
    QbReportDesigner.setUseSysResourceImages(true);

    // QbReportDesigner (component, name of XML file, true = start from
    // Data Source Manager false = start from Create a new report or open
    // existing report choice before going to Data Source Manager
    designer = new QbReportDesigner(applet,
          "DataRegistry/Sample.xml", true);
    designer.addDataSourceManagerListener(this);
    designer.setVisible(true);

}

public JTree modifyDataSourceTree(JTree tree) {

    // The following code will remove all nodes from the tree except for
    // the database nodes.
    DefaultDataSourceNode root =
 (DefaultDataSourceNode)tree.getModel().getRoot();
    for (int i=root.getChildCount()-1; i>=1; i--)
    {
        // System.out.println("removing");
        root.remove(i);

    }

    // The following code will prevent the user from adding any query or
 dataview.
    DefaultDataSourceNode databaseHeading =
 (DefaultDataSourceNode)root.getChildAt(0);
    for (int i=0; i<databaseHeading.getChildCount(); i++)
    {
        ((DefaultDataSourceNode)databaseHeading.getChildAt(i)
               .getChildAt(0)).setAddEnabled(false);
        ((DefaultDataSourceNode)databaseHeading.getChildAt(i)
               .getChildAt(1)).setAddEnabled(false);

    }

    return tree;

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/RDWithDataRegistry2.zip]

Note that the above code can be used as both an application and an applet.

2.3.5.7.11.3. Specify a DBInfo Object (Database Information)

You can open Report Designer and have it starting at the *Select Report Type* Wizard window (with a specified DBInfo object). This allows the end users to create their own custom reports in Report Designer GUI and then view the finished report.

The following constructor is used:

```
QbReportDesigner(Object parent, DBInfo databaseInformation, boolean
 doNotStartWithOpenNewReportDialogWizard);
```

Given below is an example of calling Report Designer with a specified `DBInfo` object:

```
public QbReportDesigner designer;
String URL = "jdbc:hsqldb:help/examples/DataSources/database/woodview";
String driver = "org.hsqldb.jdbcDriver";
String username = "sa";
String password = "";
String query = "SELECT * FROM Order_Details";

public void doReportDesignerWithDBInfoApplet(Applet applet) throws Exception
 {
    // Connect to EspressManager
    QbReportDesigner.setServerAddress("127.0.0.1");
    QbReportDesigner.setServerPortNumber(22071);

    // Use images from jar file
    QbReportDesigner.setUseSysResourceImages(true);

    // QbReportDesigner (component, Database Information, true = start from
 Data Source Manager
    // false = start from Create a new report or open existing report choice
 before
    // going to Data Source Manager
    DBInfo dbInfo = new DBInfo(URL, driver, username, password, query);
    designer = new QbReportDesigner(applet, dbInfo, true);

    designer.setVisible(true);

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/RDWithDBInfo.zip]

Note that the above code can be used as both an application and an applet.

2.3.5.7.11.4. Open Query Builder with a Specified DBInfo Object (Database Information)

You can open Query Builder with a specified `DBInfo` object. This allows the end users to create and save their own custom SQL queries in Query Builder GUI.

Given below is an example of calling Query Builder with a specified `DBInfo` object:

```
public void doQueryBuilderApplet(Applet applet, String sqlName) {

    // Connect to EspressManager
    QbReportDesigner.setServerAddress("127.0.0.1");
    QbReportDesigner.setServerPortNumber(22071);

    // Use images from jar file
    QbReportDesigner.setUseSysResourceImages(true);

    queryMain = new QbQueryBuilder(applet, this);
    if (sqlName != null) {
        // sqlName .qry file name (without extension)
        // System.out.println("OPEN QUERY - " + sqlName);
        queryMain.openQuery(sqlName, false, null, null, driver, URL,
 username, password);

    } else {
        // System.out.println("NEW QUERY ");
        queryMain.newQuery("Setup Query", false, null, null, driver, URL,
 username, password);

    }
    frame = queryMain.getBuilder();
    frame.setVisible(true);
    try { queryMain.showTablesWindow();
    } catch (Exception ex) {};

}

public void back() {};
public void cancel() {};

public void next() {

    queryMain.getBuilder().setVisible(false);
    DBInfo dbInfo = new DBInfo(URL, driver, username, password,
 queryMain.getSQL());
    designer = new QbReportDesigner(applet, dbInfo, queryMain.getInSet(),
 true);
    designer.setVisible(true);

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/QBWithDBInfo.zip]

Note that the above code can be used as both an application and an applet.

You can also open Query Builder by passing in the database connection (schema) as a java object, rather than as a `DBInfo` object. Given below is an example of calling Query Builder by passing in the schema information:

```
public QBWithDBInfo2() {

    // Connect to EspressManager
    QbReportDesigner.setServerAddress("127.0.0.1");
    QbReportDesigner.setServerPortNumber(22071);

    // Use images from jar file
    QbReportDesigner.setUseSysResourceImages(true);

    try {
        Class.forName(driver);
        conn = DriverManager.getConnection(URL, username, password);
        metaData = conn.getMetaData();

    } catch (Exception ex) { ex.printStackTrace(); }

    queryMain = new QbQueryBuilder(null, this);
    queryMain.newQuery("Setup Query", this);
    queryMain.setVisible(true);

    try { queryMain.showTablesWindow();
    } catch (Exception ex) {};

}

public void back() {};
public void cancel() {};

public void next() {

    // System.out.println("EXIT");

}

public String getDatabaseProductName() throws Exception {

    return metaData.getDatabaseProductName();

}
```

```
public ResultSet getTables(String catalog, String schemPattern, String
 tableNamePattern,
String[] types) throws Exception {

    return metaData.getTables(catalog, schemPattern, tableNamePattern,
 types);

}

public String getNumericFunctions() throws Exception {

    return metaData.getNumericFunctions();

}

public String getStringFunctions() throws Exception {

    return metaData.getStringFunctions();

}

public String getTimeDateFunctions() throws Exception {

    return metaData.getTimeDateFunctions();

}

public String getSystemFunctions() throws Exception {

    return metaData.getSystemFunctions();

}

public ResultSet executeQuery(String sql) throws Exception {

    Statement stmt = conn.createStatement();
    return stmt.executeQuery(sql);

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/QBWithDBInfo2.zip]

Note that the above code can be used as both an application and an applet.

2.3.5.7.11.5. Open Report Designer with a Specific Class File Data Source

You can open Report Designer with a specified class file data source. This allows the end users to create their own custom reports in Report Designer GUI and then view the finished reports. The class file data source can be non-parameterized or parameterized.

The following constructor is used:

```
QbReportDesigner(Object parent, String classFile,
 quadbase.common.paramquery.QueryInParamSet inset, boolean newReport, int
 displayRow, String[] imagesPath);
```

Given below is an example of calling Report Designer with a specified class file data source:

```
public void doReportDesignerWithQueryApplet(Applet applet) {
    // Connect to EspressManager
    QbReportDesigner.setServerAddress("127.0.0.1");
    QbReportDesigner.setServerPortNumber(22071);

    // Use images from jar file
    QbReportDesigner.setUseSysResourceImages(true);

    // QbReportDesigner (component, class file location, parameter
 information,
    // true = start from Data Source Manager false = start from Create a new
    // report or open existing report choice before going to Data Source
    // Manager, number of rows to display (-1 means show all rows), path to
    // Report Designer images
    designer = new QbReportDesigner(applet, "ParamClassFile",
         null, true, -1, null); // Parameterized class file

    designer.setVisible(true);

    JFrame frame = designer.getDesigner();

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/RDWithClassData.zip]

For more information on creating your custom class file data source, see Appendix 2.B.5 - Data passed in a Custom Implementation.

2.3.5.7.11.6. Open Report Designer with Custom Functions

You can also include your own functions with the Formula and Scripts dialog boxes that users can use when calling Report Designer from the API. You can in effect create functions that handle complex computations and allow any user to use that same function in Report Designer.

Given below is an example code that shows you how to customize Report Designer by assigning custom functions to the Formulas and Scripts dialog boxes. The code takes in a number and converts it to an IP address by appending the number at the end of **192.168.0**.

```
public void doRDWithCustomFunctions(Applet applet) throws Exception {
 QbReportDesigner.setServerAddress("127.0.0.1");
 QbReportDesigner.setServerPortNumber(22071);

 QbReportDesigner.setUseSysResourceImages(true);

 designer = new QbReportDesigner(applet, "RDWithCustomFunctions.rpt");
 designer.setCustomDefinedFunctions(this);
 designer.setVisible(true);
}

public String[] getAllFunctionNames() {
 return new String[] { "inet_ntoa" };
}

public int getReturnType(String functionName) {
 if (functionName.equals("inet_ntoa")) {
  return STRING;
 }
 return -1;
}

public int[] getParamTypes(String functionName) {
 if (functionName.equals("inet_ntoa")) {
  return new int[] { NUMERIC };
 }
 return null;
}

public Object getValue(String functionName, Object[] args) throws Exception
 {
 if (functionName.equals("inet_ntoa")) {
  return "192.168.0." + ((Double) args[0]).intValue();
 }
 return null;
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/RDWithCustomFunctions.zip]

Note that the above code can be used as both an application and an applet.

The cell containing the function will initially display as `Null` since the function is not part of the core set. However, the function will compute correctly when previewing the data. The value will then be shown correctly thereafter.

2.3.5.7.11.7. Open Report Designer with Pre-set Directories

You can also set the directories that reports load from and save to. This feature will restrict users from navigating to any level above the specified directory, giving you the ability to control which files the user is able to see.

Given below is an example that shows you how to customize Report Designer by specifying different directories for the load folder and the save folder:

```
public void doCustomizeDesignerMenuApplet(Applet applet) throws Exception {
 QbReportDesigner.setServerAddress("127.0.0.1");
 QbReportDesigner.setServerPortNumber(22071);

 QbReportDesigner.setUseSysResourceImages(true);

 designer = new QbReportDesigner(applet, "RDWithModifiedBars.rpt");

 // Add a new menu item
 JMenuBar menuBar = designer.getReportMenuBar();
 JMenu fileMenu = menuBar.getMenu(0);
 newItem = new JMenuItem("Hello World");
 newItem.addActionListener(this);
 fileMenu.insert(newItem, 7);

 // Remove toolbar buttons
 JToolBar designBar = designer.getDesignerToolBar();
 designBar.remove(5); // Remove Separator
 designBar.remove(4); // Remove Apply Template Button
 designer.setSaveOnExitEnabled(false); // Do not prompt to save the report
 if unsaved on exiting Designer

 designer.setVisible(true);

}

public void actionPerformed(ActionEvent e) {

 /*** save report **********/
 designer.save("RDWithModifiedBars_Temp.rpt");
 /**** create new testing frame *********/
 JPanel contentPane = new JPanel();
 contentPane.setLayout(new BorderLayout());
 contentPane.add(new JLabel("Hello World!"), "Center");
 JFrame frame = new JFrame();
 frame.setContentPane(contentPane);
 frame.pack();
 frame.setVisible(true);

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/RDWithPreSetDirectories.zip]

Note that the above code can be used as both an application and an applet. Also, in the above example, you can choose Save As and specify a directory to save the .rpt file. However, the .rpt file will always be saved in the temp directory.

In addition to the above approach, you can also use the following method to get the default directories used by Report Designer:

```
public BrowseDirectories getBrowseDirectories();
```

You can then use the methods in BrowseDirectories [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/common/util/BrowseDirectories.html] to get the default location of the directories for the different browse dialogs.

2.3.5.7.11.8. Open Report Designer with Modified Menubar and Toolbar

You can also add items to the menu and remove items from the toolbar. Given below is an example illustrating that:

```
public void doCustomizeDesignerMenuApplet(Applet applet) {

    // Connect to EspressManager
    QbReportDesigner.setServerAddress("127.0.0.1");
    QbReportDesigner.setServerPortNumber(22071);

    // Use images from jar file
    QbReportDesigner.setUseSysResourceImages(true);

    designer = new QbReportDesigner(applet, "help/manual/code/templates/
RDWithModifiedBars.rpt");

    // Add a new menu item
    JMenuBar menuBar = designer.getReportMenuBar();
    JMenu fileMenu = menuBar.getMenu(0);
    newItem = new JMenuItem("Hello World");
    newItem.addActionListener(this);
    fileMenu.insert(newItem, 7);

    // Remove toolbar buttons
    JToolBar designBar = designer.getDesignerToolBar();
    designBar.remove(5); // Remove Separator
    designBar.remove(4); // Remove Apply Template Button
    // Do not prompt to save the report if unsaved on exiting Designer
    designer.setSaveOnExitEnabled(false);

    designer.setVisible(true);

}

public void actionPerformed(ActionEvent e) {

    /*** save report **********/
    designer.save("RDWithModifiedBars_Temp.rpt");
```

```
        /**** create new testing frame *********/
        JPanel contentPane = new JPanel();
        contentPane.setLayout(new BorderLayout());
        contentPane.add(new JLabel("Hello World!"), "Center");
        JFrame frame = new JFrame();
        frame.setContentPane(contentPane);
        frame.pack();
        frame.setVisible(true);
    }
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/RDWithModifiedBars.zip]

Note that the above code can be used as both an application and an applet.

2.3.5.7.11.9. Open Report Designer with Skipped Wizard Steps

You can open Report Designer with some of the Wizard steps skipped. Simply call the following functions to skip the `query result`, `multiple data source`, and `pre-defined templates` steps:

```
qbReportDesigner.setSkipQueryResultStep(true);
qbReportDesigner.setSkipMultiDataSourceStep(true);
qbReportDesigner.setSkipPredefinedTemplatesStep(true);
```

2.3.5.7.11.10. Open Report Designer with Customized Messages and Chart Designer

You can open Report Designer with customized warnings when navigating between sub-reports and drill-down report. In addition, you can also override the default Save As behavior for the Report Designer. The following example demonstrates ways to alter these various components. The example has been broken down into fragments to make it easier to read. The full source code can be found at the end of this section.

```
public class RDWithCustomMessage {

    public static void main(String[] args) {
        try {
            startDesigner(); } catch (Exception ex) {
            ex.printStackTrace(); } }

    static void startDesigner() {
        // Connect to EspressManager
        QbReportDesigner.setServerAddress("127.0.0.1");
        QbReportDesigner.setServerPortNumber(22071);

        // Use images from jar file
        QbReportDesigner.setUseSysResourceImages(true);

        QbReportDesigner designer = new QbReportDesigner((Frame)null);
        // modify warning message before adding drill-down layer

 designer.modifyWarningMessage(QbReportDesigner.SAVE_RPT_BEFORE_DRILLDOWN,
            "The report designer will save the current report for you,
 continue?");

        // modify warning message before inserting Sub-Report

 designer.modifyWarningMessage(QbReportDesigner.SAVE_RPT_BEFORE_SUBREPORT,
            "The report designer will save the current report for you,
 continue?");

        // by pass save as dialog when submitting save location
        designer.setReportIO(new ReportIO(designer));
        // by pass save as dialog when user try to navigate to next level
        designer.setByPassSaveAsIO(new ByPassSaveAsForReport());

        // modify chart designer from report
        designer.setChartDesignerHandle(new ChartDesignerHandle());

        // REMOVE "SAVE AS" option for Report Designer
        JMenu fileMenu = designer.getReportMenuBar().getMenu(0);
        fileMenu.remove(6);
        designer.setVisible(true); }

}
```

The above portion of the code modifies the warning message that is shown when adding a sub-report or a drill-down level. It also calls `setReportIO` , `setByPassSaveAsIO`, and `setChartDesignerHandle` to by pass the Save As dialogs (details are discussed below). The code also removes the Save As option from the menu bar in Report Designer.

```
public static class ReportIO implements IReportIO {

    String fileName = null;
    QbReportDesigner designer = null;

    public ReportIO(QbReportDesigner designer) { this.designer = designer; }

    public void saveFile(byte[] data, String fileName) {
        System.out.println("SAVE REPORT FILE - " + fileName);
        try {
            FileOutputStream fout = new FileOutputStream(fileName);
            fout.write(data, 0, data.length); } catch (Exception ex) {
            ex.printStackTrace(); }
        this.fileName = fileName; }

}
```

The above fragment implements the `IReportIO` interface. By setting the `ReportIO` in the previous code fragment, Report Designer will call the `saveFile` method here when the user tries to save the report. The file is passed to this method as a byte array and the filename as a string. This method creates an output stream using the filename and writes the report byte array to that stream.

```
public static class ByPassSaveAsForReport implements IByPassSaveAsForReport
 {

    public ByPassSaveAsForReport() {};

    public String getFileName(String originalFileName) {
        System.out.println("BY PASS REPORT SAVE AS OPTION...");
        if (originalFileName == null) return "TEMP_REPORT_FILE.rpt";
        else return originalFileName; }

    public Properties getSaveAsProperties(String originalFileName) {
        return new Properties(); }

}
```

When you by pass the Save As option, the `saveFile` method from the previous code fragment will obtain the filename from the `getFileName` method in the above class. Here, we are simplifying the process by hardcoding the name of the file. Typically, you will want to manipulate the filenames so that users do not overwrite each other's files. The `getSaveAsProperties` method returns a Properties instance containing any save options. Possible options include: CREATE_STL, SAVE_ALL_DATA, CREATE_HTML, USE_SWINGVIEWER, CREATE_XML, CREATE_PACK, and USE_PAGEVIEWER. Here, we choose to not use any.

```
public static class ChartDesignerHandle implements IChartHandle {

    public ChartDesignerHandle() {};

    public void processDesigner(QbChartDesigner designer) {
        System.out.println("PROCESS CHART....");
        // REMOVE "SAVE AS" option for Chart Designer
        JMenu fileMenu = designer.getChartMenuBar().getMenu(0);
        fileMenu.remove(6);
        designer.setChartIO(new ChartIO());
        designer.setByPassSaveAsIO(new ByPassSaveAsForChart()); }

}

public static class ChartIO implements IChartIO {

    public ChartIO() {};

    public String saveChartFile(byte[] data, String fileName) {
        System.out.println("SAVE CHART FILE - " + fileName);
        try {
            FileOutputStream fout = new FileOutputStream(fileName);
            fout.write(data, 0, data.length);
            fout.close(); } catch (Exception ex) {
            ex.printStackTrace(); }
        return fileName; }

}

public static class ByPassSaveAsForChart implements IByPassSaveAsForChart {

    public ByPassSaveAsForChart() {};

    public String getFileName(String originalFileName) {
        System.out.println("BY PASS CHART SAVE AS OPTION...");
        if (originalFileName == null) return "TEMP_CHART_FILE.cht";
        else return originalFileName; }

    public Properties getSaveAsProperties(String originalFileName) {
        return new Properties(); }

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/RDWithCustomMessage.zip]

The above three classes makes the same changes to the Chart Designer side. The ChartDesignerHandler class removes the Save As option from the menu bar. The ChartIO class saves the file given the filename and byte array. And the ByPassSaveAsForChart class sets a fixed filename for the chart.

2.3.5.7.12. Changing Report/Page Viewer Options

At times, you may want to configure what users can or cannot do when they are viewing the report using the Report Viewer or Page Viewer.

When the user is using the Viewers to view the report, right clicking on the report causes a menu to pop up. Using this menu, the user can navigate through different pages of the report, as well as perforM other tasks on the report, such as exporting it as a DHTML or PDF file. This can easily be done using the Report Viewer and Page Viewer APIs. However, these powerful features of the Report Viewer can be too complicated for the average user.

Therefore, a few API methods have been introduced to control what options are available in the pop-up menu of the Report/Page Viewer.

These API calls are:

```
viewer.setMenuVisible(boolean b);
viewer.setPageMenuVisible(boolean b);
viewer.setPageMenuItemVisible(String[] items, boolean b);
viewer.setOutputMenuVisible(boolean b);
viewer.setOutputMenuItemVisible(String[] items, boolean b);
viewer.setRefreshMenuItemVisible(boolean b);
viewer.setGoToMenuItemVisible(boolean b);
viewer.setSortMenuVisible(boolean b);
```

For more detailed information on these API methods, please consult the APIDocs [https://data.quadbase.com/Docs71/er/help/apidocs/index.html]

2.3.5.8. Javadoc

Javadoc for the entire API is provided along with EspressReport. The API covers both Report and Charting API. It is located at Quadbase website [https://data.quadbase.com/Docs71/er/help/apidocs/index.html].

2.3.5.9. Swing Version

1.1 JFC/Swing versions of Report API are also available. For more details, please refer to quadbase.reportdesigner.ReportViewer.swing [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/reportdesigner/ReportViewer/swing/package-summary.html]

2.3.5.10. Summary

The EspressReport API provides an easy-to-use, yet powerful application programming interface for business applications. Combined with Report Designer, programming is as simple as adding one line of code to your applet. All of the attributes of a report may be set in a template file, which can be created with the Report Designer. The EspressReport API has been tested with Netscape's Communicator (4.06 and above), Microsoft's Internet Explorer (4.x and above), and Sun's Appletviewer (1.2 and above) on the Windows 95, Windows NT/2000/XP, Solaris, Linux, AIX, and HP platforms.

2.4. Scheduler

2.4.1. Introduction

EspressReport has a Scheduler interface through which reports can be exported at a specific time or specific intervals. This allows EspressReport to handle mundane tasks such as exporting reports than having a user log in at a specific time and explicitly perform the operation. For more information on how to use Scheduler, please see Section 1.13 - Scheduling.

When writing Scheduler API code, two main classes are used; `ScheduleObject`, which is used to give details on the schedule job, and `ScheduleModifier`, which is used to alter Scheduler.

Both `ReportAPIWithChart.jar` and `Scheduler.jar` (in the `EspressReport/lib` directory) need to be added to the `CLASSPATH` for any code using the Scheduler.

All examples and code given in the manual assume that EspressManager is running locally (i.e., on your local machine) and default port number (22071). You can change this by going to the source code (you can download the source code by clicking on the *Full Source Code* link in the corresponding chapter), editing the code to enter the new connection information and recompiling the code.

2.4.2. Connecting to EspressManager

You cannot run Scheduler (or any Scheduler code) without connecting to EspressManager first. Depending on whether EspressManager is running as an application or a servlet, you can use API methods to specify connection information.

2.4.2.1. EspressManager Running as Application

If EspressManager is running as an application, you can use API methods to specify the IP address/machine name where EspressManager is located and the port number EspressManager is listening on.

You use the following two API methods to set the connection information:

```
static void setServerAddress(java.lang.String address);
static void setServerPortNumber(int port);
```

For example, the following lines of code:

```
QbScheduler.setServerAddress("someMachine");
QbScheduler.setServerPortNumber(somePortNumber);
```

will connect to EspressManager running on `someMachine` and listening on `somePortNumber`.

Please note that if the EspressManager connection information is not specified, the code will attempt to connect to EspressManager on the local machine and listening to the default port number (22071).

Please note that these methods exist in the `QbScheduler` and `ScheduleModifier` classes.

2.4.2.2. EspressManager running as servlet

If EspressManager is running as a servlet, you can use the following methods:

```
public static void useServlet(boolean b);
public static void setServletRunner(String comm_URL);
public static void setServletContext(String context);
```

For example, the following lines of code:

```
QbScheduler.useServlet(true);
QbScheduler.setServletRunner("http://someMachine:somePortNumber");
QbScheduler.setServletContext("EspressReport/servlet");
```

will connect to EspressManager running at `http://someMachine:somePortNumber/EspressReport/servlet`.

Please note that these methods exist in the `QbScheduler` and `ScheduleModifier` classes.

2.4.3. Invoking Scheduler Graphical Interface

Scheduler can be called from Report API in either an application or an applet. The following example, which can run as an applet or application, invokes Scheduler by constructing a `QbScheduler` Object:

```
QbScheduler qbScheduler = new QbScheduler((Applet)null, frame);
((JPanel)frame.getContentPane()).add("Center", qbScheduler.getScheduler());
frame.pack();
frame.setVisible(true);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/InvokingScheduler.zip]

A few “set” methods are also available if the programmer wishes to configure the Scheduler Interface to open with specific features turned on or off. Here are two methods that can be used to set the root directory for browsing and to set the `insert command` feature on of off:

```
qbScheduler.setRootDirectoryForBrowse(String root);
qbScheduler.setInsertCommandEnabled(boolean state);
```

2.4.4. Scheduling an Export

You can schedule a report to be exported using Scheduler. The following code, given below, shows how to do this:

```
   ScheduleObject sObj = new ScheduleObject("Report_OneTime",
ScheduleObject.REPORTOBJ);
sObj.setFileLocation("help/manual/code/templates/Account.rpt");
sObj.setReportType(IExportConstants.DHTML);
sObj.setTaskOption(ScheduleObject.ONE_TIME);
String exportLocation = sObj.pickDefaultExportLocation();
Calendar calendar = Calendar.getInstance();
calendar.add(Calendar.MINUTE, 1);
sObj.setStartDate(calendar.getTimeInMillis());
sObj.setSendEmail(false);

ScheduleModifier.addScheduleTask(sObj);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SchedulingExportReportOneTime.zip]

Note that any references to a data source or to a report file using the relative URL reference (e.g. `help/manual/code/templates/Account.rpt`) are relative to the directory from where EspressManager is running in.

The above code schedules a report (Account.rpt) to be exported to DHTML, in the default export directory and sets the export to take place once and in one minute.

The examples below show how to set up a periodic schedule:

```
   sObj.setFileLocation("Account.rpt");
sObj.setReportType(IExportConstants.DHTML);
String exportLoc = sObj.pickDefaultExportLocation();

sObj.setTaskOption(ScheduleObject.TIME_INTERVAL);
/*** every 5 mins ****/
sObj.setIntervalType(ScheduleObject.TIME);
sObj.setTimeInterval(5); // export every 5 mins

/*** every day ****/
//   sObj.setIntervalType(ScheduleObject.DAYS);
//   sObj.setDayInterval(1); // export everyday

Calendar calendar = Calendar.getInstance();
calendar.add(Calendar.MINUTE, 5);
sObj.setStartDate(calendar.getTimeInMillis());
Calendar calendar2 = Calendar.getInstance();
calendar2.add(Calendar.MINUTE, 26);
sObj.setEndDate(calendar2.getTimeInMillis());
sObj.setSendEmail(false);

ScheduleModifier.addScheduleTask(sObj);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SchedulingExportReportPeriodically.zip]

The above code snippet shows how to schedule a job to run every five minutes (or every day, depending on which section of the code you comment).

The following code snippet shows to schedule a job to run at certain time periods:

```
   ScheduleObject sObj = new ScheduleObject("Report_Time",
ScheduleObject.REPORTOBJ);
sObj.setFileLocation("Account.rpt");
String exportLoc = sObj.pickDefaultExportLocation();
sObj.setReportType(IExportConstants.PDF);
sObj.setTaskOption(ScheduleObject.FIXED_DAYS);
/*** export every 2 hrs. monday through friday ****/
sObj.setSpecifyDays(new int[] { 1, 2, 3, 4, 5 });
sObj.setStartTime(9 * 60); // 9:00AM in minutes
sObj.setEndTime(22 * 60); // 10:00PM in minutes
sObj.setTimeInterval(2 * 60); // export every 2 hours
/*** export at specific times on specific dates ***/
// sObj.setSpecifyDates(new int[]{26,27,28,29}); // export on specific
dates
// sObj.setSpecifyTime(new int[]{16 * 60, 17 * 60, 18 * 60}); // 4PM, 5PM
and 6PM in minute
Calendar calendar = Calendar.getInstance();
calendar.add(Calendar.MINUTE, 5);
sObj.setStartDate(calendar.getTimeInMillis());
sObj.setEndDate(-1); // run indefinitely
sObj.setSendEmail(false);

ScheduleModifier.addScheduleTask(sObj);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SchedulingExportReportTime.zip]

You can also specify the parameters when scheduling a parameterized chart or report to be exported. The following code shows how:

```
  ScheduleObject sObj = new ScheduleObject("ParamReport_OneTime",
ScheduleObject.REPORTOBJ);
//   Single Param Report
sObj.setFileLocation("EmployeeDetails.pak");
String exportLoc = sObj.pickDefaultExportLocation();
sObj.setReportType(IExportConstants.DHTML);
sObj.setTaskOption(ScheduleObject.ONE_TIME);
//   1st param set {"Denise Carron"}
Object tmp1[] = { "Denise Carron" };
//   2st param set {"Frank Carnody"}
Object tmp2[] = { "Frank Carnody" };
Vector paramList = new Vector();
paramList.addElement(tmp1);
paramList.addElement(tmp2);
sObj.setParamList(paramList);
Calendar calendar = Calendar.getInstance();
calendar.add(Calendar.MINUTE, 5);
sObj.setStartDate(calendar.getTimeInMillis());
sObj.setSendEmail(false);

ScheduleModifier.addScheduleTask(sObj);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SchedulingExportParamReportOneTime.zip]

A security level for the report can also be specified. The following code shows how to specify a security level (for report cells and/or query parameters):

```
  ScheduleObject sObj = new ScheduleObject("Report_Security",
ScheduleObject.REPORTOBJ);
sObj.setFileLocation("OrderFormByRegion.rpt");
sObj.setReportType(IExportConstants.DHTML);
sObj.setTaskOption(ScheduleObject.ONE_TIME);
sObj.setSecurityLevel("ParamSet");
String exportLocation = sObj.pickDefaultExportLocation();
```

```
Calendar calendar = Calendar.getInstance();
calendar.add(Calendar.MINUTE, 10);
sObj.setStartDate(calendar.getTimeInMillis());
sObj.setSendEmail(false);

ScheduleModifier.addScheduleTask(sObj);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SchedulingExportReportSecurity.zip]

You can also specify the schedules to be sent to one or more email addresses, either as a link or an attachment. The following code shows how to send a scheduled job which contains a non-parameterized report to multiple recipients:

```
   ScheduleObject sObj = new ScheduleObject("Report_Email",
ScheduleObject.REPORTOBJ);
sObj.setFileLocation("Account.rpt");
sObj.setReportType(IExportConstants.DHTML);
sObj.setTaskOption(ScheduleObject.ONE_TIME);
String exportLocation = sObj.pickDefaultExportLocation();
Calendar calendar = Calendar.getInstance();
calendar.add(Calendar.MINUTE, 10);
sObj.setStartDate(calendar.getTimeInMillis());
sObj.setSendEmail(true);
sObj.setFromAddress("user1@quadbase.com");
sObj.setSubject("Report Email Account Report");
sObj.setBodyText("Scheduled Export of Account from user1");
sObj.setEmailType(ScheduleObject.ASATTACHMENT);
sObj.setToAddresses(new String[]
{ "user2@quadbase.com", "user3@quadbase.com" });

ScheduleModifier.addScheduleTask(sObj);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SchedulingExportReportEmail.zip]

Similarly, emails can be sent for each parameter set chosen for a parameterized report or chart when scheduling an export. Note that at least one email address must be specified for each parameter set and the same recipient can be used for multiple parameter sets.

```
  ScheduleObject sObj = new ScheduleObject("ParamReport_Email",
ScheduleObject.REPORTOBJ);
//  Single Param Report
sObj.setFileLocation("EmployeeDetails.pak");
String exportLoc = sObj.pickDefaultExportLocation();
sObj.setReportType(IExportConstants.DHTML);
sObj.setTaskOption(ScheduleObject.ONE_TIME);
//  1st param set {"Denise Carron"}
Object tmp1[] = { "Denise Carron" };
//  2nd param set {"Frank Carnody"}
Object tmp2[] = { "Frank Carnody" };
Vector paramList = new Vector();
paramList.addElement(tmp1);
paramList.addElement(tmp2);
sObj.setParamList(paramList);
Vector paramNameList = new Vector();
paramNameList.addElement("Denise");
paramNameList.addElement("Frank");
sObj.setParamList(paramNameList, paramList);
Calendar calendar = Calendar.getInstance();
```

```
calendar.add(Calendar.MINUTE, 5);
sObj.setStartDate(calendar.getTimeInMillis());
sObj.setSendEmail(true);
sObj.setFromAddress("user1@quadbase.com");
sObj.setSubject("Param Report Email Account Report");
sObj.setBodyText("Scheduled Export of Employee Details from user1");
sObj.setEmailType(ScheduleObject.ASATTACHMENT);
Hashtable toAddr = new Hashtable();
toAddr.put("Denise", new String[] { "user2@quadbase.com" });
toAddr.put("Frank", new String[] { "user3@quadbase.com" });
sObj.setParamAddresses(toAddr);

ScheduleModifier.addScheduleTask(sObj);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SchedulingExportParamReportEmail.zip]

The scheduled reports can also be stored in a web-accessible location and the URL sent via email to various recipients. The following code shows how:

```
   ScheduleObject sObj = new ScheduleObject("Report_Email",
ScheduleObject.REPORTOBJ);
sObj.setFileLocation("Account.rpt");
sObj.setReportType(IExportConstants.DHTML);
sObj.setTaskOption(ScheduleObject.ONE_TIME);
String exportLocation = sObj.pickDefaultExportLocation();
Calendar calendar = Calendar.getInstance();
calendar.add(Calendar.MINUTE, 10);
sObj.setStartDate(calendar.getTimeInMillis());
sObj.setSendEmail(true);
sObj.setFromAddress("user1@quadbase.com");
sObj.setSubject("Report Email Account Report");
sObj.setBodyText("Scheduled Export of Account from user1");
sObj.setEmailType(ScheduleObject.ASLINK);
sObj.setURLMapping("http://someMachine:somePort/ScheduledReports/");
sObj.setToAddresses(new String[]
{ "user2@quadbase.com", "user3@quadbase.com" });

ScheduleModifier.addScheduleTask(sObj);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SchedulingExportReportEmailLink.zip]

2.4.5. Scheduling a Command

In addition to scheduling exports, you can also schedule commands to be run at a specific time (or time interval). The code remains the same as the one for scheduling exports except the command to be executed is specified (rather than the report location, export format and export location).

```
ScheduleObject sObj = new ScheduleObject("Command_OneTime",
 ScheduleObject.COMMANDOBJ);
 // Note that someClass does not exist
 sObj.setCommand("java someClass");
 sObj.setTaskOption(ScheduleObject.ONE_TIME);
 Calendar calendar = Calendar.getInstance();
 calendar.add(Calendar.MINUTE, 5);
 sObj.setStartDate(calendar.getTimeInMillis());
 sObj.setSendEmail(false);
```

```
ScheduleModifier.addScheduleTask(sObj);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SchedulingCommandOneTime.zip]

2.4.6. Removing a Schedule

In addition to adding schedules, you can also use the API to remove any schedules. This is done by getting a list of all the jobs scheduled and then delete a particular job.

The following code shows how to delete a particular job:

```
Vector schedList = ScheduleModifier.getScheduler().getScheduleList();

ScheduleObject obj = (ScheduleObject)schedList.elementAt(0);
ScheduleModifier.removeScheduleTask(obj);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/RemovingSchedule.zip]

Note that the above code removes the first job in the schedule list. The job may not be visible if it has been already marked for deletion. The recommended approach is to go through the vector and search each `ScheduleObject` by the name and then deleting the correct one.

2.4.7. Getting Details of a Failed Schedule

Sometimes, a scheduled job may fail for some reason or another. EspressManager keeps a track of all the failed jobs so that its details can be obtained later. The following code shows how to obtain the details:

```
   Vector vec = ScheduleLog.getFailedScheduledJob();
   for (int i = 0; i < vec.size(); i++) {

       FailedScheduledJob obj = (FailedScheduledJob) vec.elementAt(i);
       System.out.println("FAIL # " + i + ": Name = " +
obj.getScheduledJobName());
       System.out.println("File Location = " + obj.getFileLocation());
       System.out.println("Time = " + obj.getExportTime());
       System.out.println("StackTrace = " + obj.getStackTrace());

   }
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/FailedSchedule.zip]

You can also set up an email to be sent in case the schedule fails. The following code illustrates how:

```
ScheduleObject sObj = new ScheduleObject("Report_FailedEmail",
ScheduleObject.REPORTOBJ);
sObj.setFileLocation("Account.rpt");
sObj.setReportType(IExportConstants.DHTML);
sObj.setTaskOption(ScheduleObject.ONE_TIME);
String exportLocation = sObj.pickDefaultExportLocation();
Calendar calendar = Calendar.getInstance();
calendar.add(Calendar.MINUTE, 10);
sObj.setStartDate(calendar.getTimeInMillis());
sObj.setSendEmail(true);
sObj.setFromAddress("user1@quadbase.com");
sObj.setSubject("Report Email Account Report");
sObj.setBodyText("Scheduled Export of Account from user1");
sObj.setEmailType(ScheduleObject.ASATTACHMENT);
sObj.setToAddresses(new String[]
{ "user2@quadbase.com", "user3@quadbase.com" });
```

```
sObj.setFailSubject("Report_FailedEmail schedule failed");
sObj.setFailBodyText("Cannot send Account report from Report_FailedEmail");
sObj.setFailToAddress("user1@quadbase.com");

ScheduleModifier.addScheduleTask(sObj);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ScheduleReportFailedEmail.zip]

2.4.8. ICallBackScheduler Interface

You can also create additional code that performs certain actions based on the Scheduler performing a job (whether successful or not). This code implements the `ICallBackScheduler` interface and specifies what to do when a job has been successfully completed or not.

Please note that after creating the code you need to change `EspressManager.bat/.sh` (in the `<EspressReport-installation-directory>` directory) and add the following:

```
-SchedulerCallBackClass:<name of class file implementing ICallBackScheduler>
```

The following code shows how to use the `ICallBackScheduler` interface:

```
    import quadbase.scheduler.ICallBackScheduler;
    import quadbase.scheduler.ScheduleObject;

    public class CallBackScheduler implements ICallBackScheduler {

        public CallBackScheduler() {};

        public void exportSucceeded(ScheduleObject obj, String path) {

            System.out.println(obj.getName() + "- EXPORT SUCCEEDED");
            System.out.println("PATH = " + path);

        }

        public void exportFailed(ScheduleObject obj, String path, Throwable
 e) {

            System.out.println(obj.getName() + "- EXPORT FAILED");
            System.out.println("PATH = " + path);
            System.out.println("ERROR = " + e.toString());

        }

    }
```

To use the above code, add the following to `EspressManager.bat/.sh`:

```
-SchedulerCallbackClass:CallBackScheduler
```

Make sure that `CallBackScheduler` is in the `CLASSPATH`. The code will then print messages at the end of each job saying whether the job was run successfully or not.

2.4.9. Scheduler Listener

EspressReport allows scheduled reports to be modified via custom code using server extensions. These custom classes will intercept reports before they are exported and allow users to implement additional business logic to the scheduling process.

To use the Scheduler Listener, you will have to write your own custom class that implements `SchedulerListener`. Given below is an example:

```
    import quadbase.reportdesigner.ReportAPI.QbReport;
    import quadbase.scheduler.ScheduleObject;
    import quadbase.ext.SchedulerListener;

    public class MyEspressReportSchedulerListener implements
 SchedulerListener {

        public QbReport modifyBeforeExport(QbReport report, ScheduleObject
 so, String exportPath) {

            System.out.println("modifyBeforeExport(" + report + "," + so
 + "," + exportPath + ")");
            return report;

        }

    }
```

The above example prints a simple `System.out.println` statement before exporting either the report.

To use any custom class implementing `SchedulerListener`, you will have to implement another custom class implementing `DefaultListenerManager`. For example:

```
    import quadbase.ext.DefaultListenerManager;
    import quadbase.ext.SchedulerListener;

    public class MyEspressReportListenerManager extends
 DefaultListenerManager {

        public MyEspressReportListenerManager() {}

        public EspressReportSchedulerListener getSchedulerListener() {

            return new MyEspressReportSchedulerListener();

        }

    }
```

To use the above code, add the following to `EspressManager.bat/.sh`:

```
-ListenerManagerClass:MyEspressReportListenerManager
```

Make sure that both `MyEspressReportListenerManager` and `MyEspressReportSchedulerListener` are in the `CLASSPATH`.

2.4.10. Report Bursting

You can also use the Report Bursting feature in the Scheduler by using the API. Note that the report must use a database as the datasource and must be grouped in order for the report to be bursted.

Given below is an example on how to use report bursting:

```
ScheduleObject sObj = new ScheduleObject("ReportBursting_All",
 ScheduleObject.REPORTOBJ);
sObj.setFileLocation("InventoryInformation.rpt");
String exportLoc = sObj.pickDefaultExportLocation();
sObj.setReportType(IExportConstants.DHTML);
sObj.setTaskOption(ScheduleObject.ONE_TIME);
Calendar calendar = Calendar.getInstance();
```

```
calendar.add(Calendar.MINUTE, 5);
sObj.setStartDate(calendar.getTimeInMillis());
sObj.setSendEmail(true);
sObj.setFromAddress("user1@quadbase.com");
sObj.setSubject("Test report bursting");
sObj.setBodyText("You should find a report of one group");
sObj.setEmailType(ScheduleObject.ASATTACHMENT);
// export every group as a single report and email using report column
 values
sObj.setBurstReport(ScheduleObject.ALLBURSTING);
// get email address from 16th column in template
sObj.setEmailColumnIndex(16);

ScheduleModifier.addScheduleTask(sObj);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ReportBurstingAll.zip]

The above code bursts all groups and sends out an email using a report column as the source for the addresses.

Note that in the above code, column 16 does not exist in the template. The method is provided in the example to show how to pass in the email column for bursting.

2.4.11. Javadoc

Javadoc for the entire API is provided along with EspressReport. It is located at Quadbase website [https://data.quadbase.com/Docs71/er/help/apidocs/index.html].

2.4.12. Summary

EspressReport API provides an easy-to-use and powerful API to query the Scheduler interface, as well as to add and remove schedules. You can also write code to perform your own action when a job has been completed successfully (or not) in Scheduler.

Please note that the API requires a JDK 1.8 or above. The EspressReport API has been tested on Windows, Solaris, Linux, AIX, and HP platforms.

2.5. Servlets and Java Server Pages

2.5.1. Servlets

2.5.1.1. Introduction

As described on the Sun Microsystems®web site (`http://java.sun.com`), Java servlets provide web developers with a simple, consistent mechanism for extending the functionality of a web server and for accessing existing business systems. A servlet can almost be thought of as an applet that runs on the server side -- without a face. Java Servlets have made many web applications possible.

Today, servlets are a popular choice for building interactive web applications. Servlet containers are available for Apache Web Server, iPlanet Web Server (formerly Netscape Enterprise Server), Microsoft IIS, and others. Servlet containers can also be integrated with web-enabled application servers, such as BEA WebLogic Application Server, IBM WebSphere, Netscape Application Server, and others.

The sections below describe how to set up an example servlet provided with EspressReport (located in `EspressReport/help/examples/servlet/StreamingReport` directory). Please note that each section deals specifically with the example servlet. However, they can be used as a guide for setting up and running your own servlets in addition to the other example servlets given.

In addition to this servlet, other servlet examples have also been provided. They range from the simple showing of a report (either in HTML or DHTML format) to the streaming of a report containing charts directly to the browser. Setup and running of these examples can be done using the following sections as a guide:

2.5.1.2. Setup

- Make sure that the following files:-

  - `Summary_1.rpt`
  - `Summary_2.rpt`
  - `CrossTab.rpt`

  exist in the EspressReport`/help/examples/templates` directory.

- Make sure the EspressManager is running.

- Next, depending on what servlet server (runner) that you are using, follow the guidelines below. Then run `index.html` (located in the `EspressReport/help/examples/servlet/StreamingReport` directory) from a browser.

Note that even though the instructions refer to the Windows platforms, they can be used for Unix/Linux platforms as well. Changing the file names and path to conform to the Unix/Linux standard is the only necessary step.

The examples all use single threaded model for simplicity's sake. However, EspressReport API can also be used in a multi-threaded environment also.

The following uses the StreamingReport servlet example under `EspressReport/help/examples/servlet`.

2.5.1.3. Running Under

2.5.1.3.1. Apache Tomcat 5.5.20 or 6.0.10

1. Please, be sure, that you have Tomcat server in version 5.5.20 or 6.0.10 installed However, this guide is probably also applicable for previous versions of Tomcat server.

   There are two installation file versions for Windows operating system (for every version) of ApacheTomcat server:

   zip file - uses `startup.bat`

   exe file - installs Windows service and uses “Apache Tomcat Monitor” application

2. Classpath – You should include `ReportDesigner.jar` and `servlet-api.jar` files in the classpath.

   ReportDesigner.jar file can be found in `EspressReport/lib` directory

   servlet-api.jar can be found in Tomcat installation directory (`apache-tomcat-5.5.20\common\lib\` directory if you use **zip installation file;** `C:\Program Files\Apache Software Foundation\Tomcat 5.5\common\lib\` directory as the default Tomcat installation directory when using **exe installation file**)

   Here is the example classpath setting (using command line console):

   ```
   set classpath=%classpath%;C:\Program Files\Apache Software Foundation
   \Tomcat 5.5\common\lib\servlet-api.jar;C:\EspressReport\lib
   \ReportDesigner.jar
   ```

   (when using Tomcat 5.5 zip installation file)

   ```
   set classpath=%classpath%;C:\apache-tomcat-5.5.20\common\lib\servlet-
   api.jar;C:\EspressReport\lib\ReportDesigner.jar
   ```

   (when using Tomcat 5.5 installer - exe installation file)

Note

For Tomcat 6.0 version, please use following:

```
set classpath=%classpath%;C:\Program Files\Apache Software
 Foundation\Tomcat 6.0\lib\servlet-api.jar;C:\EspressReport\lib
\ReportDesigner.jar
```

(when using Tomcat 6.0 zip installation file)

```
set classpath=%classpath%;C:\apache-tomcat-6.0.10\lib\servlet-
api.jar;C:\EspressReport\lib\ReportDesigner.jar
```

(when using Tomcat 6.0 installer - exe installation file)

Note

Classpath does not necessarily need to be set, but it simplifies `.java` files compilation (you don't need to add classpath there).

Note

You can set Classpath variable also using MyComputer → Administrate → Details tab window.

3. Compiling the example file (`streamingReportServlet.java`)
   Compile command:

   ```
   javac -classpath "C:\Program Files\Apache Software Foundation\Tomcat
    5.5\common\lib\servlet-api.jar;C:\EspressReport\lib\ReportDesigner.jar"
    streamingReportServlet.java
   ```

   (for Tomcat 5.5 version)

   ```
   javac -classpath "C:\Program Files\Apache Software Foundation\Tomcat
    6.0\lib\servlet-api.jar;C:\EspressReport\lib\ReportDesigner.jar"
    streamingReportServlet.java
   ```

   (for Tomcat 6.0 version)

   (if classpath environment variable was not set in step 2 of this guide)

   or simply:

   ```
   javac streamingReportServlet.java
   ```

   (if classpath was set properly)

4. Copying to Tomcat's `webapps\ROOT\WEB-INF\classes` directory
   Copy `streamingReportServlet.class` file to Tomcat's `webapps\ROOT\WEB-INF\classes` directory.

Note

Please note that `webapps\ROOT\WEB-INF\classes` directory does not need to exist by default and you will have to create it.

5. Adding necessary libraries for running the servlet to lib dir

In order to run the example, you also need to add libraries to Apache Tomcat's `webapps\ROOT\WEB-INF\lib` directory. Please copy `ReportDesigner.jar` file placed in `EspressReport\lib directory to the webapps\ROOT\WEB-INF\lib` directory.

Note

Please note that the `webapps\ROOT\WEB-INF\lib` directory does not need to exist by default and you will have to create it.

6. Registering your servlet appplication in `web.xml` file of your Tomcat server.

   Before you can run your servlet application, you need to modify `web.xml` file that is placed in `webapps\ROOT\WEB-INF\` directory of your Tomcat server.

   For registering `streamingReportServlet`, please add the following code:

   ```
   <servlet>
     <servlet-name>streamingReportServlet</servlet-name>
     <servlet-class>streamingReportServlet</servlet-class>
   </servlet>

   <servlet-mapping>
     <servlet-name>streamingReportServlet</servlet-name>
     <URL-pattern>/servlet/streamingReportServlet</URL-pattern>
   </servlet-mapping>
   ```

   between `<web-app>` and `</web-app>` xml tags.

Note

When using Tomcat in version 5.5.20, you can also enable `automatic invoking` of all servlets placed in webapp directory. This way you do not need to register your servlet applications.

This is intended only for test purposes. Using the invoker servlet in a production environment is not recommended and is unsupported.

7. Start (restart) the Tomcat Server

   If you used zip installation file, please run `startup.bat`(`startup.sh` for Linux) file from bin directory of Tomcat installation directory.

   For exe installation files you need to startup the Tomcat service. You can use Tomcat's “Monitor Tomcat ”application for this purpose.

Note

If your Tomcat server is already running, you probably need to restart it(shutdown and startup again) to make the example runnable.

8. Running the example

   The last step is running the example. To run the example, please run `index.html` file placed in `EspressReport\help\examples\servlet\StreamingReport` directory in your browser.

   After clicking the *GetReport* button, the `streamingReportServlet` is called and appropriate report is generated.

Note

Please be sure, that EspressManager is running before you run the example. If it is not running, empty report is generated.

Note

Please note that the servlet as designed and given will only work if both the servlet runner and the client browser are on the same machine. To allow the client to be on a different machine, modify the line in `Servlet.html` file `(placed in EspressReport\help\examples\servlet\StreamingReport` directory)

from:

```
<form action="http://127.0.0.1:8080/servlet/streamingReportServlet"
 method="post">
```

to:

```
<form action="http://<machine_name>:8080/servlet/
streamingReportServlet" method="post">
```

Note

If you use Automatic settings discovery in your web browser, there can be problem reaching `127.0.0.1` machine depending on your network settings.

Please use `localhost` instead.

```
<form action="http://localhost:8080/servlet/streamingReportServlet"
 method="post">
```

2.5.1.3.2. JRun 3.1

- Replace the machine name and port number in `Servlet.html` file to

  ```
  <FORM ACTION=http://<machine name>:8100/servlet/streamingReportServlet
   method=POST>
  ```

- Log into the JRun Default Server Administrator.
- Click on *Java Settings* and then *Classpath*.
- Add the path to `ReportAPIWithChart.jar` and click on *Update*.
- Restart the JRun Default Server.
- Compile the servlet and copy it to `<jrun_installation_directory>/servers/default/default-app/WEB-INF/classes`.

2.5.1.3.3. ColdFusion Server 4.5

- Replace all html files with.cfm extensions (For example, `Servlet.html` becomes `Servlet.cfm`) and replace the machine name and port number in the cfm file to

  ```
  <FORM ACTION=http://<machine name>:8100/servlet/streamingReportServlet
   method=POST>
  ```

- Log into the JRun Default Server Administrator.
- Click on *Java Settings* and then *Classpath*.
- Add the path to `ReportAPIWithChart.jar` and click on *Update*.
- Restart the JRun Default Server.

- Compile the servlet and copy it to `<jrun_installation_directory>/servers/default/default-app/WEB-INF/classes`.

2.5.1.3.4. WebLogic 6.0

To set up and run the servlet example, first, go to the `wlserver6.0\config\examples` directory. Modify both `setExamplesEnv.cmd` and `startExamplesServer.cmd` files according to the following instructions:

- Add `set ES_REPORT=<directory where EspressReport is installed>` in the `@rem Set user-defined variables` section. The `ES_REPORT` variable contains the path to the EspressReport Home directory in your machine. Modify the path value to correspond to the path in your machine. Also, please make sure the `WL_HOME` and `JAVA_HOME` variables correspond to the correct paths on your computer.

- Add `%ES_REPORT%\lib\ReportAPIWithChart.jar;`. to the *set CLASSPATH* field of the same file.

- Put the html files in the `wlserver6.0\config\examples\applications\examplesWebApp` directory.

- Change the machine name in the code from `127.0.0.1:8080` to `<machineName>:7001` in the `Servlet.html` file.

- Put the `streamingReportServlet.java` in the `wlserver6.0\samples\examples\servlets` directory.

- Insert the following code fragments in the `web.xml` file located in the `wlserver6.0\config\examples\applications\examplesWebApp\WEB-INF\` directory. The `<servlet>` code would go in the `<servlet>` code section in the file and in the `<servlet-mapping>` section.

```
<servlet>

    <servlet-name>streamingReportServlet</servlet-name>
    <servlet-class>streamingReportServlet</servlet-class>
    <init-param>

        <param-name>reportType</param-name>
        <param-value>Summary</param-value>
        <param-name>rowBreaks</param-name>
        <param-value>ProductType</param-value>
        <param-name>browser</param-name>
        <param-value>Netscape</param-value>

    </init-param>

</servlet>
<servlet-mapping>

    <servlet-name>streamingReportServlet</servlet-name>
    <URL-pattern>/streamingReportServlet/*</URL-pattern>

</servlet-mapping>
```

- In a command prompt window, go to the `wlserver6.0\config\examples` directory and run `setExamplesEnv`.

- Then, go to `wlserver6.0\samples\examples\servlets` directory and compile `streamingReportServlet.java` separately using the command lines listed below.

  ```
  javac –d %EX_WEBAPP_CLASSES% streamingReportServlet.java
  ```

- In the same command prompt window, go back to `wlserver6.0\config\examples` directory and start the WebLogic server by typing `startExamplesServer` and pressing the Enter key.

- Open your web browser and go to `http://yourMachineName:7001/ examplesWebApp/index.html` to view the servlet example.

2.5.1.3.4.1. WebLogic 9.2

The following instructions show how to set up and run the servlet example under WebLogic 9.2. The instructions assume that you have WebLogic 9.2 server installed on the system. The location of the WebLogic installation will be referenced as `<WL_INSTALL_DIR>` and the location of the EspressReport installation will be referenced as `<ER_INSTALL_DIR>`.

To set up and run the servlet example, first, go to the `<WL_INSTALL_DIR>/samples/domains/wl_server/bin` directory. Modify the `setExamplesEnv.cmd` file according to the following instructions.

- Add `set ES_REPORT=<ER_INSTALL_DIR>`. The `ES_REPORT` variable contains the path to the EspressReport Home directory in your machine. Modify the path value to correspond to the path in your machine. Also, please make sure the `WL_HOME` and `JAVA_HOME` variables correspond to the correct paths on your computer.

- Add `%ES_REPORT%\lib\ReportAPIWithChart.jar;%ES_REPORT%\lib\qblicense.jar;` to the *set CLASSPATH* field of the same file.

Next, follow the steps below (note that the files are under `<ER_INSTALL_DIR>\help\examples\servlet\StreamingReport` directory):

- Put the html files (`columnar.html`, `servlet.html`, `index.html`) to the `<WL_INSTALL_DIR>\samples\server\examples\build\examplesWebApp` directory.

- Edit the `servlet.html` file and change the machine name in code from `127.0.0.1:8080/servlet` to `yourmachineName:7001/examplesWebApp`.

- Put the `streamingReportServlet.java` file in the `<WL_INSTALL_DIR>\samples\server\examples\build\examplesWebApp\WEB-INF\classes` directory.

- Insert the following code fragments in the `web.xml` file located in the `<WL_INSTALL_DIR>\samples\server\examples\build\examplesWebApp\WEB-INF` directory. The `<servlet>` code would go in the `<servlet>` code section in the file and `<servlet-mapping>` in the `<servlet-mapping>` section.

```
<servlet>

    <servlet-name>streamingReportServlet</servlet-name>
    <servlet-class>streamingReportServlet</servlet-class>

</servlet>

<servlet-mapping>

    <servlet-name>streamingReportServlet</servlet-name>
    <URL-pattern>/streamingReportServlet/*</URL-pattern>

</servlet-mapping>
```

- Then, go to `<WL_INSTALL_DIR>\samples\server\examples\build\examplesWebApp\WEB-INF\classes` directory and compile `streamingReportServlet.java`. Please include `ReportAPIWithChart.jar` and `javax.servlet.jar` in the classpath (Note: `javax.servlet.jar` is located under `bea\jrockit90_150_06\mercuryprofiler\lib` directory).

- In a command prompt window, go back to `<WL_INSTALL_DIR>/samples/domains/wl_server` directory and start the WebLogic server by typing `startWebLogic.cmd` and pressing the Enter key.

- Open your web browser and go to `http://yourMachineName:7001/examplesWebApp/index.html` to view the servlet example.

Note

For troubleshooting, please check for typing errors.

2.5.1.3.5. WebSphere 3.5

- Open the WebSphere Administrator's Console and under the node (usually the machine name) → Default Servlet Engine → Default Host → Default App, click on the *Advanced* tab and add `ReportAPIWithChart.jar` to the Classpath. Then click on *Apply*.
- Modify the `FORM ACTION` tag in `Servlet.html` to the following:

  ```
  <FORM ACTION=http://<machine name>/servlet/streamingReportServlet
   method=POST>
  ```

- Compile `streamingReportServlet.java` and move the class file to the `WebSphere/AppServer/servlets` directory.
- Start/Restart the Default Server under the node.

2.5.1.3.5.1. WebSphere 6.1

The following instructions show how to set up and run the servlet example under WebSphere 6.1. The instructions assume that you have WebSphere 6.1 server installed on the system. The location of the WebSphere installation will be referenced as `<WS_INSTALL_DIR>` and the location of the EspressReport installation will be referenced as `<ER_INSTALL_DIR>`.

To set up and run the servlet example, first, copy `<ER_INSTALL_DIR>\lib\ReportAPIWithChart.jar` and `<ER_INSTALL_DIR>\lib\qblicense.jar` files to the `<WS_INSTALL_DIR>\AppServer\lib` directory.

Next, follow the steps below (note that the files are under `<ER_INSTALL_DIR>\help\examples\servlet\StreamingReport` directory):

- Put the `streamingReportServlet.java` file in the `<WS_INSTALL_DIR>\AppServer\profiles\AppSrv01\installedApps\<YourMachineName>Node01Cell\ivtApp.ear\ivt_app.war\WEB-INF\classes` directory.
- Edit the `servlet.html` file and change the machine name in code from `127.0.0.1:8080/servlet` to `yourmachineName:9080/ivt/servlet`.
- Put the html files (`columnar.html`, `servlet.html`, `index.html`) to the `<WS_INSTALL_DIR>\AppServer\profiles\AppSrv01\installedApps\<YourMachineName>Node01Cell\ivtApp.ear\ivt_app.war` directory.
- Insert the following code fragments in the `web.xml` file located in the `<WS_INSTALL_DIR>\AppServer\profiles\AppSrv01\installedApps\<YourMachineName>Node01Cell\ivtApp.ear\ivt_app.war\WEB-INF` directory. The `<servlet>` code would go in the `<servlet>` code section in the file and `<servlet-mapping>` in the `<servlet-mapping>` section.

  ```
  <servlet>

      <servlet-name>streamingReportServlet</servlet-name>
      <servlet-class>streamingReportServlet</servlet-class>

  </servlet>

  <servlet-mapping>

      <servlet-name>streamingReportServlet</servlet-name>
  ```

```
        <URL-pattern>streamingReportServlet</URL-pattern>

    </servlet-mapping>
```

- Then, go to `<WS_INSTALL_DIR>\AppServer\profiles\AppSrv01\installedApps\<YourMachineName>Node01Cell\DefaultApplication.ear\DefaultWebApplication.war\WEB-INF\classesdirectory` and compile `streamingReportServlet.java`. Please include `ReportAPIWithChart.jar` and `j2ee.jar` in the classpath (Note: j2ee.jar is located under `<WS_INSTALL_DIR>\AppServer\lib` directory).
- Start the WebSphere server. There are several ways to do this. The easiest way for Windows users is to launch the *First Steps* tool from Start → Programs → IBMWebSphere → Application Server v6.1 → Profiles → AppSrv01 → First Steps. You can also launch the administration console from the First Steps tool.
- Open your web browser and go to `http://yourMachineName:9080/ivt/index.html` to view the servlet example.

Note

For troubleshooting please check for typing errors.

2.5.1.3.6. WebSphere Express Server 5.0

- Go to File → New → Web Project.
- Enter in the name of the project in the *Project Name* text-field. For our example, we will use **`EspressReport`**.
- Click on the *Finish* button.
- Go to the *EspressReport* project folder in the *J2EE Navigator* panel and expand the project folder.
- Right click on the *Java Source* sub-folder and choose the *Import* option. Choose the *File System* type import.
- Import the Java source files for your servlets and java beans here. For this example, import `streamingReportServlet.java` from the `EspressReport/help/examples/servlet/StreamingReport` directory.
- Go back to the *EspressReport* project folder.
- Go to the *Web Content* sub-folder. Import JSP and HTML files here. For this example, choose the *File System* type import and import `index.html`, `Columnar.html`" and `Servlet.html` files from `EspressReport/help/examples/servlet/StreamingReport` directory.
- Go to the `WEB-INF/lib` sub-folder and right click to the *Import* option and import `ExportLib.jar` and `ReportAPIWithChart.jar` files from the `EspressReport/lib` directory.
- Double-click and open the `Servlet.html` file and hit the *Source* tab to edit.
- Modify the `FORM ACTION` tag in `Servlet.html` to the following:

  ```
  <FORM ACTION=http://<machine name>:7080/EspressReport/servlet/
  streamingReportServlet method=POST>
  ```

  where `7080` is the HTTP port number. If `7080` is not the port number, substitue the appropriate value.
- Save this HTML file.
- Right-click on the *EspressReport* web project folder and choose *Rebuild Project*.
- Right-click on the *EspressReport* web project folder and choose *Run on Server...* .

- Choose the server and click on the *Finish* button. It is recommended that you choose a server that has already been configured.
- The results will be showin in a *Web Browser* panel.

2.5.1.3.7. JBoss 4.0.5

The following instructions show how to set up and run the servlet example under JBoss 4.0.5.The instructions assume that you have JBoss 4.0.5 server installed on the system. The location of the JBoss 4.0.5 installation will be referenced as `<JB_INSTALL_DIR>` and the location of the EspressReport installation will be referenced as `<ER_INSTALL_DIR>`.

To set up and run the servlet example, first, go to the `<JB_INSTALL_DIR>\bin` directory. Modify the `run.bat` file according to the following instructions:

- Add `set ES_REPORT=<ER_INSTALL_DIR>`. The `ES_REPORT` variable contains the path to the EspressReport Home directory in your machine.
- Add `%ES_REPORT%\lib\ReportAPIWithChart.jar;%ES_REPORT%\lib\qblicense.jar;` to the `set JBOSS_CLASSPATH` field of the same file.

Next, follow the steps below (note that the files are under `<ER_INSTALL_DIR>\help\examples\servlet\StreamingReport` directory):

- Put the `streamingReportServlet.java` file in the `<JB_INSTALL_DIR>\server\default\deploy\jmx-console.war\WEB-INF\classes` directory.y
- Edit the `servlet.html` file and change the machine name in code from `127.0.0.1:8080/servlet` to `yourmachineName:8080/jmx-console` in the `servlet.html` file.
- Put the html files (`columnar.html`, `servlet.html` and `index.html`) to the `<JB_INSTALL_DIR>\server\default\deploy\jmx-console.war` directory.
- Insert the following code fragments in the `web.xml` file located in the `<JB_INSTALL_DIR>\server\default\deploy\jmx-console.war\WEB-INF` directory. The `<servlet>` code would go in the `<servlet>` code section in the file and `<servlet-mapping>` in the `<servlet-mapping>` section.

```
<servlet>

    <servlet-name>streamingReportServlet</servlet-name>
    <servlet-class>streamingReportServlet</servlet-class>

</servlet>

<servlet-mapping>

    <servlet-name>streamingReportServlet</servlet-name>
    <URL-pattern>/streamingReportServlet</URL-pattern>

</servlet-mapping>
```

- Then, go to `<JB_INSTALL_DIR>\server\default\deploy\jmx-console.war\WEB-INF\classes` directory and compile `ExportServlet2.java`. Please include `ReportAPIWithChart.jar` and `javax.servlet.jar` in the classpath (Note: `javax.servlet.jar` is located under `<JB_INSTALL_DIR>\server\all\lib` directory).
- In a command prompt window, go to `<JB_INSTALL_DIR>\bin` directory and start the JBoss server by typing `run.bat` and pressing the Enter key.
- Open your web browser and go to `http://yourMachineName:8080/jmx-console/index.html` to view the servlet example.

Note

For troubleshooting, please check for typing errors.

2.5.1.3.8. Orion 2.0.7

The following instructions show how to set up and run the servlet example under Orion 2.0.7. The instructions assume that you have Orion 2.0.7 server installed on the system. The location of the Orion installation will be referenced as `<OR_INSTALL_DIR>` and the location of the EspressReport installation will be referenced as `<ER_INSTALL_DIR>`.

Follow the steps below (note that the files are under `<ER_INSTALL_DIR>\help\examples\servlet\StreamingReport` directory):

- Put the `streamingReportServlet.java` file in the `<OR_INSTALL_DIR>\default-web-app\WEB-INF\classes` directory.

- Edit the `servlet.html` file and change the part of URL in code from `127.0.0.1:8080/servlet` to `yourmachineName:80`.

- Rename the `index.html` file to `index2.html` and put the html files (`columnar.html`, `servlet.html` and `index2.html`) to the `<OR_INSTALL_DIR>\default-web-app` directory.

- Insert the following code fragments in the `web.xml` file located in the `<OR_INSTALL_DIR>\default-web-app\WEB-INF` directory. The `<servlet>` code would go in the `<servlet>` code section in the file and `<servlet-mapping>` in the `<servlet-mapping>` section.

```
<servlet>

    <servlet-name>streamingReportServlet</servlet-name>
    <servlet-class>streamingReportServlet</servlet-class>

</servlet>

<servlet-mapping>

    <servlet-name>streamingReportServlet</servlet-name>
    <URL-pattern>streamingReportServlet</URL-pattern>

</servlet-mapping>
```

- Then, go to `<OR_INSTALL_DIR>default-web-app\WEB-INF\classes` directory and compile `streamingReportServlet.java`. Please include `ReportAPIWithChart.jar` and `j2ee.jar` in the classpath (Note: You can use j2ee.jar from the J2eeSDK installation located under `<J2EE_SDK_INSTALL_DIR>\lib` directory).

- Copy the `ReportAPIWithChart.jar` and `qblicense.jar` files to the `<OR_INSTALL_DIR>\bin` directory.

- In a command prompt window, go to `<OR_INSTALL_DIR>` directory and start the Orion server by typing `java -jar orion.jar` and pressing the Enter key.

- Open your web browser and go to `http://yourMachineName:80/index2.html` to view the servlet example.

Note

For troubleshooting, please check for typing errors.

2.5.1.3.9. JRun 4(with Update 6)

The following instructions show how to set up and run the servlet example under JRun 4 (with Update 6). The instructions assume that you have JRun server installed on the system. The location of the JRun installation will be referenced as `<JR_INSTALL_DIR>` and the location of the EspressReport installation will be referenced as `<ER_INSTALL_DIR>`.

Follow the steps below (note that the files are under `<ER_INSTALL_DIR>\help\examples\servlet\StreamingReport` directory):

- Edit the `servlet.html` file and change the part of URL in code from `127.0.0.1:8080` to `yourmachineName:8100`.
- Put the html files (`columnar.html`, `servlet.html` and `index.html`) in the `<JR_INSTALL_DIR>\servers\default\default-ear\default-war` directory.
- Put the `streamingReportServlet.java` file in the `<JR_INSTALL_DIR>\servers\default\default-ear\default-war\WEB-INF\classes` directory.
- Insert the following code fragments in the `web.xml` file located in the `<JR_INSTALL_DIR>\servers\default\default-ear\default-war\WEB-INF` directory. The `<servlet>` code would go in the `<servlet>` code section in the file and `<servlet-mapping>` in the `<servlet-mapping>` section.

```
<servlet>

    <servlet-name>streamingReportServlet</servlet-name>
    <servlet-class>streamingReportServlet</servlet-class>

</servlet>

<servlet-mapping>

    <servlet-name>streamingReportServlet</servlet-name>
    <URL-pattern>/servlet/streamingReportServlet</URL-pattern>

</servlet-mapping>
```

- Then, go to `<JR_INSTALL_DIR>\servers\default\default-ear\default-war\WEB-INF\classes` directory and compile `streamingReportServlet.java`. Please include `ReportAPIWithChart.jar` and `j2ee.jar` in the classpath (Note: You can use `j2ee.jar` from the J2eeSDK installation located under `<J2EE_SDK_INSTALL_DIR>\lib` directory).
- Copy the `ReportAPIWithChart.jar` and `qblicense.jar` files to the `<JR_INSTALL_DIR>\servers\lib` directory.
- Start the JRun 4 application server by executing `<JR_INSTALL_DIR>\bin\jrun.exe`. This will launch the *JRun Launcher* window.From JRun Launcher, start *default* server.
- Open your web browser and go to `http://yourMachineName:8100/index.html` to view the servlet example.

Note

For troubleshooting, please check for typing errors.

2.5.1.3.10. Oracle 10g (10.1.3.1.0)

The following instructions show how to set up and run the servlet example under Oracle 10g (10.1.3.1.0).The instructions assume that you have Oracle 10g (10.1.3.1.0) server installed on the system. The location of the Oracle

10g installation will be referenced as <ORA_INSTALL_DIR> and the location of the EspressReport installation will be referenced as <ER_INSTALL_DIR>.

Follow the steps below (note that the files are under <ER_INSTALL_DIR>\help\examples\servlet\StreamingReport directory):

- Edit the servlet.html file and change the part of URL in code from 127.0.0.1:8080 to yourMachineName:8888.
- Rename the index.html file to index2.html and put the html files (columnar.html, servlet.html and index2.html) in the <ORA_INSTALL_DIR>\j2ee\home\default-web-app directory.
- Put the streamingReportServlet.java file in the <ORA_INSTALL_DIR>\j2ee\home\default-web-app\WEB-INF\classes directory.
- Insert the following code fragments in the web.xml file located in the <ORA_INSTALL_DIR>\j2ee\home\default-web-app\WEB-INF directory. The <servlet> code would go in the <servlet> code section in the file and <servlet-mapping> in the <servlet-mapping> section.

```
<servlet>

    <servlet-name>streamingReportServlet</servlet-name>
    <servlet-class>streamingReportServlet</servlet-class>

</servlet>

<servlet-mapping>

    <servlet-name>streamingReportServlet</servlet-name>
    <URL-pattern>/servlet/streamingReportServlet</URL-pattern>

</servlet-mapping>
```

- Then, go to <ORA_INSTALL_DIR>\j2ee\home\default-web-app\WEB-INF\classes directory and compile streamingReportServlet.java. Please include ReportAPIWithChart.jar and servlet.jar in the classpath (Note: The servlet.jar is located under <ORA_INSTALL_DIR>\j2ee\home\lib directory).
- Copy the ReportAPIWithChart.jar and qblicense.jar files to the <ORA_INSTALL_DIR>\j2ee\home\applib directory.
- Start the Oracle server. The easiest way for Windows users is to launch the *Start Soa suite* from Start → Programs → Oracle → .... You can also launch the Oracle server by executing runstartupconsole.bat under the <ORA_INSTALL_DIR>\j2ee\home\bin directory.
- Open your web browser and go to http://yourMachineName:8888/index2.html to view the servlet example.

Note

For troubleshooting, please check for typing errors.

2.5.1.3.11. Sun Java System Application PE (9.0, 8.2)

The following instructions show how to set up and run the servlet example under Sun Java System Application PE (9.0 or 8.2). The instructions assume that you have Sun Java System Application PE server installed on the system. The location of the Sun App Server installation will be referenced as <SAP_INSTALL_DIR> and the location of the EspressReport installation will be referenced as <ER_INSTALL_DIR>.

- First, create your own directory. The location of the directory will be referenced as <USER_DIR>.

- Then, go to the `<SAP_INSTALL_DIR>\samples\quickstart` directory and copy the `hello.war` file to the `<USER_DIR>` directory. Next unpack the hello.war file. Note: In order to unpack the war file, you can use jar or unzip. For unpacking using jar, first, make sure that the `java\bin` directory is in your path and then execute the following command `jar -xf hello.war` in the `<USER_DIR>` directory.

Follow the steps below (note that the files are under `<ER_INSTALL_DIR>\help\examples\servlet\StreamingReport` directory):

- Edit the `servlet.html` file and change the part of URL in code from `127.0.0.1:8080/servlet`" to `yourMachineName:8080/hello/servlet`.

- Put the html files (`columnar.html`, `servlet.html` and `index.html`) in the `<USER_DIR>` directory.

- Put the `streamingReportServlet.java` file in the `<USER_DIR>\WEB-INF\classes` directory.

- Insert the following code fragments in the `web.xml` file located in the `<USER_DIR>\WEB-INF` directory.

```
<servlet>

    <servlet-name>streamingReportServlet</servlet-name>
    <servlet-class>streamingReportServlet</servlet-class>

</servlet>

<servlet-mapping>

    <servlet-name>streamingReportServlet</servlet-name>
    <URL-pattern>/servlet/streamingReportServlet</URL-pattern>

</servlet-mapping>
```

- Then, go to `<USER_DIR>\WEB-INF\classes` directory and compile `streamingReportServlet.java`. Please include `ReportAPIWithChart.jar` and `j2ee.jar` in the classpath (Note: The `j2ee.jar` file is located under `<SAP_INSTALL_DIR>\lib` directory).

- Navigate to the `<USER_DIR>` directory in command window and then create a war file by executing the following command: `jar -cvf hello.war *`.

- Move the `hello.war` file to the `<SAP_INSTALL_DIR>\domains\domain1\autodeploy` directory.

- Copy the `ReportAPIWithChart.jar` and `qblicense.jar` files to the `<SAP_INSTALL_DIR>\domains\domain1\lib\applibs directory`.

- Start the Sun App server by executing `<SAP_INSTALL_DIR>\bin\asadmin start-domain domain1`. The easiest way for Windows users is to launch the *Start Default Server* from Start → Programs → Sun Microsystems → Application Server PE 9.

- Open your web browser and go to `http://yourMachineName:8080/hello/index.html` to view the servlet example.

> **Note**
>
> Note: For troubleshooting, please check for typing errors.

2.5.1.3.12. Sun Java System WebServer 7.0

The following instructions show how to set up and run the servlet example under Sun Java System WebServer 7.0. The instructions assume that you have Sun Java System WebServer 7.0 installed on the system. The location of the

Sun WebServer installation will be referenced as <SWS_INSTALL_DIR> and the location of the EspressReport installation will be referenced as <ER_INSTALL_DIR>.

- First, create your own directory. The location of the directory will be referenced as <USER_DIR>.
- Then, go to the <SWS_INSTALL_DIR>\samples\java\webapps\simple directory and copy the webapps-simple.war file to the <USER_DIR> directory. Next unpack the webapps-simple.war file.

> **Note**
>
> In order to unpack the war file, you can use jar or unzip. For unpacking using jar, firstly, make sure that the java\bin directory is in your path and then execute the following command jar -xf webapps-simple.war in the <USER_DIR> directory.

Follow the steps below (note that the files are under <ER_INSTALL_DIR>\help\examples\servlet\StreamingReport directory):

- Edit the servlet.html file and change the part of URL in code from 127.0.0.1:8080/servlet to yourMachineName:81/webapps-simple/servlet.
- Rename the index.html file to index2.html and put the html files (columnar.html, servlet.html and index.html) in the <USER_DIR> directory.
- Put the streamingReportServlet.java file in the <USER_DIR>\WEB-INF\classes directory.
- Insert the following code fragments in the web.xml file located in the <USER_DIR>\WEB-INF directory. The <servlet> code would go in the <servlet> code section in the file and <servlet-mapping> in the <servlet-mapping> section.

```
<servlet>

    <servlet-name>streamingReportServlet</servlet-name>
    <servlet-class>streamingReportServlet</servlet-class>

</servlet>

<servlet-mapping>

    <servlet-name>streamingReportServlet</servlet-name>
    <URL-pattern>/servlet/streamingReportServlet</URL-pattern>

</servlet-mapping>
```

- Then, go to <USER_DIR>\WEB-INF\classes directory and compile ExportServlet2.java. Please include ReportAPIWithChart.jar and j2ee.jar in the classpath (Note: You can use j2ee.jar from the J2eeSDK installation located under <J2EE_SDK_INSTALL_DIR>\lib directory).
- Navigate to the <USER_DIR> directory in command window and create a war file by executing the following command: jar -cvf webapps-simple.war *.
- Copy the ReportAPIWithChart.jar and qblicense.jar files to the <SWS_INSTALL_DIR>\https-yourMachineName\lib directory.
- Start the administration server by executing <SWS_INSTALL_DIR>\admin-server\bin\startserv.bat. After that start your web browser and go to the Admin Console page http://yourMachineName:8989. From the Admin Console page, log in the administration server and click the *Add web application* link in the Virtual Server tasks. This will bring you to the Add Web Application window.
- From the Add Web Application window, click the *Browse* button and navigate to the <USER_DIR>/webapps-simple.war file. Specify the URI (by default: /webapps-simple) that represents application's context root and is relative to server host. Next, click the *OK* button.

- From the next screen you should see the webapps-simple application enabled. Click the *Save* button. After that you will see the *Deployment Pending* warning link in the upper right corner of the screen. Click the link and push the *Deploy* button.
- If the deployment was successful you will see the Results window that will inform you: "The configuration has been deployed successfully to all available nodes".
- Open your web browser and go to `http://yourMachineName:81/webapps-simple/index2.html` to view the servlet example.

> **Note**
>
> For troubleshooting, please check for typing errors.

2.5.1.3.13. Resin 3.1.0

The following instructions show how to set up and run the servlet example under Resin 3.1.0. The instructions assume that you have Resin 3.1.0 installed on the system. The location of the Resin installation will be referenced as `<RES_INSTALL_DIR>` and the location of the EspressReport installation will be referenced as `<ER_INSTALL_DIR>`.

- First, create your own directory. The location of the directory will be referenced as `<USER_DIR>`.
- Then, go to the `<RES_INSTALL_DIR>\webapps` directory and copy the `resin-doc.war` file to the `<USER_DIR>` directory. Next, unpack the `resin-doc.war` file.

> **Note**
>
> In order to unpack the war file, you can use jar or unzip. For unpacking using jar, make sure that the `java\bin` directory is in your path and then execute the following command `jar -xf resin-doc.war in the <USER_DIR>` directory.

- Navigate to the `<USER_DIR>\tutorial` directory and copy the `servlet-hello` directory to the `<RES_INSTALL_DIR>\webapps`. Next rename the `servlet-hello` directory to `test`.

Follow the steps below (note that the files are under `<EC_INSTALL_DIR>\help\examples\servlet\DataFile` directory):

- Edit the `servlet.html` file and change the part of URL in code from `127.0.0.1:8080/servlet` to `yourMachineName:8080/test/servlet`.
- Put the html files (`columnar.html`, `servlet.html` and `index.html`) in the `<RES_INSTALL_DIR>\webapps\test` directory.
- Put the `streamingReportServlet.java` file in the `<RES_INSTALL_DIR>\webapps\test\WEB-INF\classes` directory.
- Insert the following code fragments in the `resin-web.xml` file located in the `<RES_INSTALL_DIR>\webapps\test\WEB-INF` directory.

  ```
  <servlet-name="streamingReportServlet"
   servlet-class="streamingReportServlet"/>

  <servlet-mapping URL-pattern="/servlet/streamingReportServlet" servlet-
  name="streamingReportServlet"/>
  ```

- Then, go to `<RES_INSTALL_DIR>\webapps\test\WEB-INF\classes` directory and compile `streamingReportServlet.java`. Please include `ReportAPIWithChart.jar` and `j2ee.jar` in the classpath (Note: You can use `j2ee.jar` from the J2eeSDK installation located under `<J2EE_SDK_INSTALL_DIR>\lib` directory).

- Copy the `ReportAPIWithChart.jar` and `qblicense.jar` files to the `<RES_INSTALL_DIR>\lib` directory.
- Start the Resin server by executing `httpd.exe` under the `<RES_INSTALL_DIR>` directory.
- Open your web browser and go to `http://yourMachineName:8080/test/index.html` to view the servlet example.

> **Note**
>
> For troubleshooting, please check for typing errors.

2.5.1.3.14. ColdFusion MX 7.02

The following instructions show how to set up and run the servlet example under ColdFusion MX 7.02. The instructions assume that you have ColdFusion MX 7 application server installed on the system. The location of the ColdFusion MX 7.02 installation will be referenced as `<CF_INSTALL_DIR>` and the location of the EspressReport installation will be referenced as `<ER_INSTALL_DIR>`.

Follow the steps below (note that the files are under `<ER_INSTALL_DIR>\help\examples\servlet\StreamingReport` directory):

- Edit the `servlet.html` file and change the part of URL in code from `127.0.0.1:8080` to `yourmachineName:8500`.
- Put the html files (`columnar.html`, `servlet.html` and `index.html`) in the `<CF_INSTALL_DIR>\wwwroot` directory.
- Put the `streamingReportServlet.java` file in the `<CF_INSTALL_DIR>\wwwroot\WEB-INF\classes` directory.
- Insert the following code fragments in the `web.xml` file located in the `<CF_INSTALL_DIR>\wwwroot\WEB-INF` directory. The `<servlet>` code would go in the `<servlet>` code section in the file and `<servlet-mapping>` in the `<servlet-mapping>` section.

```
<servlet>

        <servlet-name>streamingReportServlet</servlet-name>
        <servlet-class>streamingReportServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>streamingReportServlet</servlet-name>
        <URL-pattern>/servlet/streamingReportServlet</URL-pattern>

    </servlet-mapping>
```

- Then, go to `<CF_INSTALL_DIR>\wwwroot\WEB-INF\classes` directory and compile `streamingReportServlet.java`. Please include `ReportAPIWithChart.jar` and `j2ee.jar` in the classpath

> **Note**
>
> You can use `j2ee.jar` from the J2eeSDK installation located under `<J2EE_SDK_INSTALL_DIR>\lib` directory

- Copy the `ReportAPIWithChart.jar` and `qblicense.jar` files to the `<CF_INSTALL_DIR>\runtime\lib` directory.
- Start ColdFusion application server. For Windows platforms,the ColdFusion application server is installed to run as a service. It should be started automatically. If the server is not running, navigate to the Windows Services and start the `ColdFusion MX 7 Application server` service. Re-start the application server, if necessary.
- Open your web browser and go to `http://yourMachineName:8500/index.html` to view the servlet example.

Note

For troubleshooting, please check for typing errors.

2.5.1.4. Running the Servlet

After setting up the servlet per the instructions given above, open `index.html` (located in the `EspressReport/help/examples/servlet/StreamingReport` directory) from a browser. Select from the parameters given and then click on *Get Report*.

2.5.2. Java Server Pages (JSP)

2.5.2.1. Introduction

Java Server Pages allows the separation of the dynamic content from the static HTML. JSPs allow HTML to be written in the traditional manner and then the code for the dynamic content is enclosed within the HTML within special tags, which usually start with "<%" and end with `%>`

The JSPs can be placed along with regular HTML files and they look like HTML files in many ways. However, at run time, JSPs get converted into normal servlets while the static HTML pages are printed.

Java Server Pages are basically an extension of the Servlet technology. However, there is one very big advantage in using JSPs. Java Server Page technology separates the user interface from content generation enabling designers to change the overall page layout without altering the underlying dynamic content.

EspressReport provides a few JSP examples. The example we will be looking at `EspressReport/help/examples/jsp/StreamingReport`. This example can also be used as a guide to run the other JSP examples as well as your own JSP code.

2.5.2.2. Running Under

2.5.2.2.1. Apache Tomcat 4.0

- Please include `ReportAPIWithChart.jar` and `servlet.jar` in the classpath. For instance,

  ```
  set classpath=c:\EspressReport\lib\ReportAPIWithChart.jar;c:\Apache Tomcat
   4.0\common\lib\servlet.jar;
  ```

- Create a subdirectory called report under `webapps\ROOT\` . Therefore, if Tomcat is installed under `c:\Apache Tomcat 4.0`, you now have a new directory `c:\Apache Tomcat 4.0\webapps\ROOT\report` .
- Copy `StreamingReport.jsp`, and `streamerror.jsp` to the `webapps/ROOT/report` directory.
- Create another directory, also called report under the `webapps/ROOT/WEB-INF/classes` directory. Thus, you now have `c:\Apache Tomcat 4.0\webapps\ROOT\WEB-INF\classes\report` .
- Place the file `CreateStreamingReport.java` in the `c:\Apache Tomcat 4.0\webapps\ROOT\WEB-INF\classes\report` directory.
- Compile `CreateStreamingReport.java`. `CreateStreamingReport.class` should now exist in the same directory.

- Start the Tomcat server. (Run `startup.bat`) and start EspressManager.
- Open a web browser and go to `http://yourMachineName:8080/report/StreamingReport.jsp` (just substitute your actual machine name for yourMachineName).
- Specify the options that you want and then submit to get a report in return.

2.5.2.2.2. WebSphere 3.5

- Add `ReportAPIWithChart.jar` to the Dependent Classpath under the node (usually called by the machine name).
- Click on the *Wizards* button on the toolbar (the last button) and select *Create a Web Application*.
- Enter **`Quadbase`** (without the quotes) as the name of the web application. Make sure to check the Enable File Servlet option and check Server Servlet by classname and click on *Next*.
- Choose *Default Servlet Engine* as the node (keep expanding the tree untill you can select the *Default Servlet Engine*) and click on *Next*.
- Change the Web Application Web Path to `/Quadbase` and click on *Next*.
- Add `ReportAPIWithChart.jar` to the Classpath.
- Note the document root and add it to the Classpath (you will need to create the directory for this path if it does not exist) and click on *Finish*.
- Restart the Default Server.
- Create a directory under your web server's document root called **`Quadbase`** (make sure the path matches the Web Applications document root) and move the .jsp files and the Java file there.
- Modify `CreateStreamingReport.java` to remove (or comment out) the package and compile the Java file. The class file should be in the document root.
- Modify the .jsp files to replace report.CreateStreamingReport with CreateStreamingReport.
- Start your browser and open `http://<machine name>/Quadbase/StreamingReport.jsp` . Input the parameters and click on the button to get the report. Make sure you replace the `<machine name>` with a host alias setting found in WebSphere admin console.

2.5.3. Saving a Report to a File versus Sending a Report to a Browser

You can use both servlets and JSPs to return a report back to the browser. The report can either be shown in a JNLP (More about Applets in JNLP: Section 1.2.7 - Run Applets in WebStart with JNLP file) (i.e., it can be interactive) or be shown in a static format (i.e., HTML, DHTML, PDF). Net traffic and bandwidth availability should be the key factors in deciding whether to use JNLP (More about Applets in JNLP: Section 1.2.7 - Run Applets in WebStart with JNLP file) or static formats.

The reports returned to the browsers can be done in two ways: they can be saved to a file or sent directly to the browser. In the first scenario, the report is saved on the server side and the file location is sent to the client browser which then shows the report.&bsp; In the second scenario, the report itself is sent as an output stream to the client browser.

2.5.3.1. Saving the Report to a File

Servlets and JSPs can save reports to files if the reports shown need to be re-used or a copy is needed. Here, the reports are saved to a file on the server side (whether the report is shown in an applet or in a static format is immaterial). However, these types of servlets and JSPs need to be constructed carefully. If multiple clients hit the page, it is possible for one client to see another client's report. And the browsers might sometimes cache a previous report (in the case of a static format) and will not show the new report unless refreshed/reloaded again. Also, since the files are saved on the server side, care must be taken that the files are not overridden before the client browser has seen the report. Reports should also be purged periodically to avoid taking up server space.

In both cases, a report is exported to the desired format.

```
QbReport report = new QbReport (..........);
....
....
....
int format = QbReport.RPT;    // Here set to an interactive format.  For a
 static format change to QbReport.HTML
report.export(format, "report");
```

A HTML file can then be returned to the browser which contains the report location within the HTML file so that the browser can view it.

```
    <applet code="quadbase.reportdesigner.ReportViewer.Viewer.class"
 width=100% height=100% archive="http://<machineName>/EspressReport/lib/
ReportViewer.jar">
        <PARAM name="filename" value="report.rpt">

    </applet>
```

Report Viewer has been used here. Please note that when using Report Viewer in the above code, EspressManager MUST be running.

In the case of a static format, calling the HTML file (or providing a link to the PDF file within a HTML file) is sufficient to display the report.

Using this method, you can separate the pages returning the report from your servlet or JSP and design them well in advance and outside of your servlet or JSP. You can use a database or a hashtable to keep track of the reports generated and show them again as needed, as well as have the reports periodically re-generated.

To see an example of a servlet saving the report as a HTML or DHTML file on the server side, please go to `EspressReport/help/examples/servlet/Report`.

When you export reports containing charts in HTML or DHTML format, you **MUST** use the method `setHTMLParameters` to set the directory the charts have to be exported to, the http URL mapped to that directory and to set the name of the export chart. Thus in the exported report, `<img src>` tags point to the URL and file name specified (with extensions).

For example, given a report with three charts in it and the following method call:

```
 reportChartObject.setHTMLParameters("D:/inetpub/wwwroot/EspressReport/
chartimages/",
"http://<machine name>/EspressReport/chartimages/",
"ReportWithChart");
```

When the report is exported, the above code sees to it that the chart gets exported to the `D:/inetpub/wwwroot/EspressReport/chartimages/` directory, with the charts being named `ReportWithChart_0`, `ReportWithChart_1` and `ReportWithChart_2` (the extension depends on the image type set). And in the HTML/DHTML report, the following lines of code will be present:

```
<img src = "http://<machine name>/EspressReport/chartimages/
ReportWithChart_0">
<img src = "http://<machine name>/EspressReport/chartimages/
ReportWithChart_1">
<img src = "http://<machine name>/EspressReport/chartimages/
ReportWithChart_2">
```

in place of the charts in the report.

To see an example of a servlet saving the report that contains charts as either a HTML or DHTML file on the server side, please go to EspressReport/help/examples/servlet/ReportWithChart.

You can also add charts that contain hyperlinks to the report. When you export these types of reports to a DHTML or HTML file, the map file is automatically generated within the DHTML/HTML file as long as the option for the map file is selected in the Designer, or by setting the option in the Report API. The option is set by getting a handle to the ReportChartObject and then using the method setExportMapFile.

```
ReportChartObject reportChartObject = .......;
reportChartObject.setExportMapFile(true);
```

2.5.3.2. Sending the Report Directly to a Browser

Servlets and JSPs can send reports directly to the browser so that they do not need to be saved as files on the server side. Thus, disk space on the server is not being filled up and maintenance of files need not be done. However, a permanent copy of the report cannot be made (unless printed from the browser). Therefore, the report would need to be reprinted if it is needed again..

Sending reports to the browser differs from saving them to files, as here reports are sent as an outputstream.

For a static format, on the servlet side, an outputstream is created and the report is exported to the outputstream.

```
     public class OutputStreamServlet extends HttpServlet  {

        public void doGet(HttpServletRequest request, HttpServletResponse
 response) throws ServletException, IOException {

            // first, set the "content type" header of the response
            response.setContentType("text/html");
            // where response is the response to the servlet

            OutputStream toClient = res.getOutputStream();
            QbReport report = new QbReport (..........);
            ....
            ....
            ....
            report.export(QbReport.HTML, toClient);

        }

    }
```

In the browser, the servlet URL can be used to view the streamed HTML page. For example, you can enter the URL http://machine_name:8080/servlet/OutputStreamServlet to generate the report and stream it to the browser as an HTML page.

For an interactive report, the report is exported as a string.

```
QbReport report = new QbReport (..........);
String buffer = report.exportReportToString();
....
....
....
```

On the HTML side, you can use Report Viewer to view the interactive report. Here, the HTML is embedded in the servlet code itself as shown :

```
toClient.println("<applet code=\"quadbase.reportdesigner.ReportViewer.Viewer
\" width=600 height=600 " + "archive=\"http://<machineName>/EspressReport/
lib/ReportViewer.jar\">");
toClient.println("<PARAM name=\"ReportData\" value=\"" +
 report.exportReportToString() +"\">");
```

```
toClient.println("</applet>");
```

To see an example of a servlet streaming the report as a HTML or DHTML file, please go to `EspressReport/help/examples/servlet/StreamingReport`.

On a JSP, you have to create a Java Bean object (`myStreamReport`), which creates a report and returns it as a String object. The JSP page has to be modified as shown :

```
<jsp:useBean id="myStreamReport" scope="page"
 class="streamingReport.CreateReport" />
<jsp:setProperty name="myStreamReport" property="*"/>
<applet code="quadbase.reportdesigner.ReportViewer.Viewer" width=600
 height=600 archive="http://<machineName>/EspressReport/lib/
ReportViewer.jar">
<PARAM name="ReportData" value="<%= myStreamReport.export()%>">
</applet>
```

where `myStreamReport` Java bean will contain the following code :

```
     public string export() throws Exception {

        QbReport report = new QbReport(....);
        return report.exportReportToString();

    }
```

To see an example of a JSP streaming the report as a HTML file, please go to `EspressReport/help/examples/jsp/StreamingHTML`.

When you stream reports (in HTML or DHTML format) containing charts, you can either use the method `setHTMLParameters` which saves the charts to a physical location on the server side, or you can stream the charts using the `RPTImageGenerator` servlet provided. To use this servlet, you must make it accessible from your servlet runner or application server. The URL to access this servlet **MUST** be `http://<machine name>:<port number>/servlet/RPTImageGenerator`. The source code for this servlet is in the `EspressReport/ImageGenerator` directory. Next, in your servlet/jsp code, you must use the `QbReport.setDynamicExport(boolean state, String serverName, int servletRunnerPort)` .

For example, given a report with three charts in it and the following method call:

```
report.setDynamicExport(true, "Aphrodite", 8080);
```

When the report is exported, the above code sees to it that the chart gets streamed using the machine Aphrodite and the port 8080. The `RPTImageGenerator` servlet gets called using the URL, `http://Aphrodite:8080/servlet/RPTImageGenerator` and thus streams the chart back to the browser.

You can also change the URL for this servlet by using the `setServletDirectory(String)` method in the `QbReport` class. For instance, given the example above, if the following line of code were added:

```
report.setServletDirectory("EspressReport/Test/");
```

then the links will show `http://Aphrodite:8080/EspressReport/Test/RPTImageGenerator` instead of the default `servlet/` in the link.

Further, if you are using SSL or wants to use a relative link to locate the `RPTImageGenerator`, you can use the following methods to configure the SRC attribute of the IMG tag that is used to locate the `RPTImageGenerator`:

```
     /** use a relative URL for the SRC tag. */
    report.setDynamicExport(boolean state, boolean
 relativeURLToImageGenerator);

    /** use https protocol along with specified server name and port number
 for the SRC tag. */
```

```
    public void setHttpsDynamicExport(boolean state, String serverName, int
 servletRunnerPort);
```

To see an example of a servlet streaming the report, which contains charts, as a HTML or DHTML file, please go to `EspressReport/help/examples/servlet/StreamingReportWithChart`.

You can also add in charts, that contain hyperlinks, to the report. When you export these types of reports to a DHTML or HTML file, the map file is automatically generated within the DHTML/HTML file as long as the option for the map file is selected in the Designer, or by setting the option in the Report API. The option is set by getting a handle to the `ReportChartObject` and then using the method `setExportMapFile`.

```
ReportChartObject reportChartObject = .......;
reportChartObject.setExportMapFile(true);
```

2.6. Working with WebObjects

2.6.1. Introduction

EspressReport for WebObjects provides seamless integration with WebObjects. It not only provides an easy way to integrate with standard EspressReportAPI by using `WOComponent` classes to return dynamic pages, but also supports generating report with WebObjects's non-java-standard collection class `NSArray`. This makes it very easy for users who use enterprise objects to retrieve data.

EspressReport for WebObjects also supports advanced reporting features, such as streaming report with charts, drill-down reports, and parameterized reports.

EspressReport for WebObjects is available as separate jar file, `EspressReportForWebObjects.jar`. Integrating it with your WebObjects project is very easy, just add it to your project frameworks. The source is available. You can change the source file and add them back to your project.

This chapter covers basic integration of EspressReport within a WebObjects application environment, as well as step-by-step examples for using some of the different features in EspressReport.

2.6.2. Basic API

This section explains how to integrate some of the basic Report API functionality within a WebObjects application.

2.6.2.1. Integrating with EspressReport Standard API

Step 1: Create a QbReport object

The following code shows how to create a `QbReport` object from scratch. For more detailed information about creating report from different data sources, please refer to Section 2.3 - EspressReport Report API.

```
// not using EspressManager, we will use a datasource from file
QbReport.setEspressManagerUsed(false);

// Set up column mapping
ColInfo colInfo[] = new ColInfo[14];
for (int i = 0; i < colInfo.length; i++)

    colInfo[i] = new ColInfo(i);

// allocate the QBReport Object using the specified parameters
QbReport report = new QbReport((Applet)null, QbReport.COLUMNAR, "/
EspressReport/help/examples/DataSources/text/sample.dat", colInfo, null);
```

Step 2: Then, add more code to the above code so that the method returns a `WOComponent`. The entire code for the method now looks like this:

```
public WOComponent returnReport() {
    // allocates a new WOComponent with static type WOReport
    // this is possible because WOReport extends WOComponent
    WOReport nextPage = (WOReport)pageWithName("WOReport");

    QbReport.setEspressManagerUsed(false);
    ColInfo colInfo[] = new ColInfo[14];
    for (int i = 0; i < colInfo.length; i++)
        colInfo[i] = new ColInfo(i);

    QbReport report = new QbReport((Applet)null, QbReport.COLUMNAR, "/
EspressReport/help/examples/DataSources/text/sample.dat", colInfo, null);

    // load in the WBReport into the WOReport
    nextPage.setReport(report);

    // configure to export the report in DHTML format
    nextPage.setExportFormat(QbReport.DHTML);

    // return the WOComponent (super-class of WOReport)
    return nextPage;

}
```

2.6.2.2. Integrating with WebObjects Collection Class NSArray

The class `WOQbReport` extends `QbReport` in the ReportAPI and is the main class used to support WebObjects. Its constructors take a `NSArray` of `EOGenericRecord` as data source and create reports based on it.

Before we can create a report, we need a data source to be used by the report. We will assume that we are using a Database with some EOModel mapped on top of it. The EOModel will contain a mapping of entities (tables), attributes (fields), and relationships (foreign keys) to the Database. For a review of concepts on how to setup the EOModel, please read the next section. You can safely skip this section if your understanding of the EOModel and EO Fetch Specification is strong.

2.6.2.2.1. EOModel and EO Fetch Specification

he EOModel is an abstraction that separates the database layer from business logic. Using the EOModel we are able to manipulate database data as if they were Objects. After the user has setup the proper EOModel using the EOModeler, he/she will be ready to fetch data based on attributes of the EOModel.

In the EOModel, there are entities (similar to tables of SQL) and attributes (similar to columns in the tables of SQL). Entities in the EOModel are mapped to tables and views in the database. Attributes in the EOModel are mapped to columns of the tables and views in the database. In addition, the EOModel also contains relationships, which are key based logical associations among entities. It is important that the user understands the concept of Entities, Relationships, and Attributes before setting up their own EOModel. Please review the relevant topics of an EOModel before setting up your EOModel.

After we have setup an EOModel for our database, we are ready to obtain data from it. Recall that the way to obtain data as a `NSArray` is by writing Fetch Specifications. An `EOFetchSpecification` is an Object that specifies what are the qualifications of data we want. In a way, writing Fetch Specification for an EOModel is similar to writing a Query for the database. In fact, it is true that if we have a Fetch Specification for an EOModel, we can always write a corresponding query for a database. However, the reverse is not true. Not all queries can be written as a Fetch Specification. In this way, a Fetch Specification serves as an abstraction that helps programmers deal with data in an object oriented way, but loses the power that queries have over the control of database data.

Having said that, let’s look at an example of translating a SQL Query for a database to a Fetch Specification. Remember this is not always possible but for our example, we have restricted our task to illustrate how it can be done.

Consider a database schema with four tables: `Orders`, `OrderDetails`, `Categories`, and `Products`.

Orders: stores information about a customer order.

- Attributes - `OrderID`, `ShipCity`, and `ShipState`.
- Primary Key - `OrderID`

OrderDetails: details about a particular order.

- Attributes - `OrderID`, `ProductID`, `Quantity`
- Primary Key - `ProductID`

Categories: categories of product.

- Attributes - `CategoryID`, `CategoryName`
- Primary Key - `CategoryID`

Products: information about a product.

- Attributes - `CategoryID`, `ProductID`, `ProductName`, `UnitsInStock`
- Primary Key - `ProductID`

Next, consider the query.

```
SELECT Orders.OrderID, Orders.ShipCity, Orders.ShipState,
 Categories.CategoryName,
Products.ProductName, OrderDetails.Quantity
FROM Orders, OrderDetails, Categories, Products
WHERE Orders.ShipState = ‘NY’
AND Products.CategoryID = Categories.CategoryID
AND OrderDetails.OrderID = Orders.OrderID
AND OrderDetails.ProductID = Products.ProductID
ORDER BY Orders.OrderID DESC, Orders.ShipCity ASC;
```

This query joins four tables and obtain orders that are only shipped to NY, and orders the records in descending order by the `OrderID` column and orders it again in ascending order by the quantity column before returning the data records.

Now, let’s look at how to convert this SQL query to a Fetch Specification.

To have a restriction on the query, we will need to pass in an `EOQualifier` object to the `EOFetchSpecification` constructor. The static method `qualifierWithQualifierFormat()` from `EOQualifier` returns an `EOQualifier`. If we simply pass in a String that is similar to a SQL `Where` clause in the first argument of the `qualifierWithQualifierFormat()` method, we can create a valid `EOQualifier` object that restricts the data being fetched. We construct the `EOQualifier` simply by including the following code:

```
EOQualifier qual = EOQualifier.qualifierWithQualifierFormat

    ("shipstate = NY", null); // we set the binding to null for simplicity
```

In the above code, the Where clause from the SQL Query is converted to an `EOQualifier` Object by passing in a String argument. It is also possible to include AND and OR to combine one or more Boolean operators in the argument of the `EOQualifier`. In addition, the `EOQualifier` class also has a collection of sub-classes that are designed for other restrictions on the query. Some of these sub-classes are Boolean Qualifiers, `EOAndQualifier`, `EOOrQualifier`, and `EONotQualifier`.

Now that we have a qualification, all we need is an ordering for the `Order ID` and `ShipCity`. By passing a `NSArray` of `EOSortOrdering` Objects to the `EOFetchSpecification` Constructor, we can accomplish this task. Here is the code:

```
EOSortOrdering shipdateOrdering = EOSortOrdering.sortOrderingWithKey
```

```
    ("shipcity", EOSortOrdering.CompareAscending);

EOSortOrdering orderidOrdering = EOSortOrdering.sortOrderingWithKey

    ("orderid", EOSortOrdering.CompareDescending);

NSArray sortOrderings = new NSArray

    (new Object[] {shipdateOrdering, orderidOrdering});
```

Finally, construct the `EOFetchSpecification` Object using an entity name, the previously constructed `EOQualifier`, and the `NSArray` of `EOSortOrdering` Objects:

```
EOFetchSpecification fetchSpec = new EOFetchSpecification

    ("Orders", qual, sortOrderings);
```

Note

Using the SQL Statement approach or the Fetch Specification approach to obtain data from the database should give the same results.

Also note that for the Fetch Specification to work correctly, you have to set up the EOModel with the Relationship correctly defined. That is, for the joining of the identities to work automatically, you have to have joined the identities in your EOModel for the database. For information on how to use the EOModeler, please consult online documentation for the EOModeler [https://developer.apple.com/library/archive/documentation/WebObjects/UsingEOModeler/Introduction/Introduction.html] at Apple's web site.

2.6.2.2.2. Creating a WOQbReport using a NSArray as a datasource

After an EOModel has been created, we are ready to write a fetch specification and obtain Data Objects from the Database using the EOModel. Lastly, we will use these Data Objects, in the form of a `NSArray` Object, as a parameter to the `WOQbReport` Constructor to create our report. The following code illustrates this concept:

```
EOQualifier qual = EOQualifier.qualifierWithQualifierFormat

    ("shipstate = 'NY'", null);

EOSortOrdering shipdateOrdering = EOSortOrdering.sortOrderingWithKey

    ("shipcity", EOSortOrdering.CompareAscending);

EOSortOrdering orderidOrdering = EOSortOrdering.sortOrderingWithKey

    ("orderid", EOSortOrdering.CompareDescending);

NSArray sortOrderings = new NSArray

    (new Object[] {shipdateOrdering, orderidOrdering});

EOFetchSpecification fetchSpec = new EOFetchSpecification

    ("Orders", qual, sortOrderings);

EOEditingContext myEditingContext = this.session().defaultEditingContext();
NSArray data = myEditingContext.objectsWithFetchSpecification(fetchSpec);

String[] entities = new String[]
{"Orders", "OrderDetails", "Products", "Categories"};
```

```
WOQbReport report = new WOQbReport(null, QbReport.SUMMARY, data, entities,
 colInfo, null);
```

In the above code, we first instantiated an `EOFetchSpecification` Object to retrieve data from the database via the EOModel. Then fetch the data using the default Editing Context. Finally, we call the `WOQbReport` constructor and pass in the `NSArray` as the data source to create our report. Note that we also pass in a String array of Entity names to the constructor of the `WOQbReport`. This is because we also want to display the attributes of the entities that relate to the `Orders` Entity. It is also important to note that the entities `OrderDetails`, `Products`, and `Categories` all are “related” to the `Orders` entity in some type of relationship setup in the EOModel.

For an example WebObjects Project that creates a `WOQbReport` using a `NSArray` as the data source, please use the project EspressReportExamples in the folder `<EspressReport installation dir>/help/examples/webobjects/`.

2.6.3. Deploying a WebObjects Application

This section explains step-by-step how to integrate EspressReport into a WebObjects application using Eclipse.

2.6.3.1. Install Xcode

Xcode is a suite of tools for developing software on Mac OS X, developed by Apple. It includes (among other things) the compiler and other necessary tools. Xcode comes with the OS X install CD (although it is not installed by default).

2.6.3.2. Install Eclipse

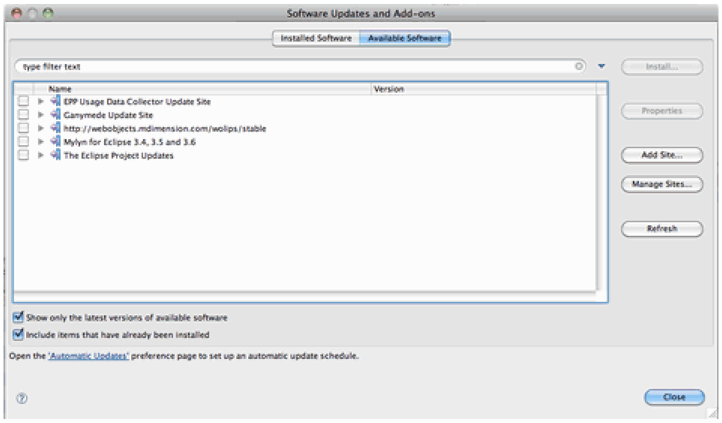
Eclipse is a Java IDE and the latest Eclipse application, for the Mac, can be downloaded from `http://www.eclipse.org/downloads`. Before launching Eclipse, please edit `<eclipse folder>/Eclipse.app/Contents/MacOS/eclipse.ini` and add the following parameters to the file in order to increase the memory and heap space of the Eclipse application:

- `-XX:MaxPermSize=128m`
- `-Xmx512m`

After saving the file, start Eclipse.

2.6.3.3. Install WOLips

WOLips is a set of Eclipse tools for WebObjects development. You can download and install the plug-in (through Eclipse) by going to Help → Software Updates and Add-ons. This opens the *Available Software* panel. The first time you go through this process, you will need to add the WOLips site to the list. Click on *Available Software Sites* and it should pop up a new dialog. Select *Add Site...* on the right-hand side from the dialog. In the *Add Site* panel that appears, enter **`WOLips`** in the name field and enter the URL for the stable version of WOLips, `http://webojects.mdimension.com/wolips/stable`, for the location field and click *OK*.



Name	Version
□ ▸ EPP Usage Data Collector Update Site	
□ ▸ Ganymede Update Site	
□ ▸ http://webobjects.mdimension.com/wolips/stable	
□ ▸ Mylyn for Eclipse 3.4, 3.5 and 3.6	
□ ▸ The Eclipse Project Updates	

☑ Show only the latest versions of available software

☑ Include items that have already been installed

Open the 'Automatic Updates' preference page to set up an automatic update schedule.

After the WOLips site is added to the list, fully expand the list to display all its offerings. Select only Standard Install. After installation, restart Eclipse.

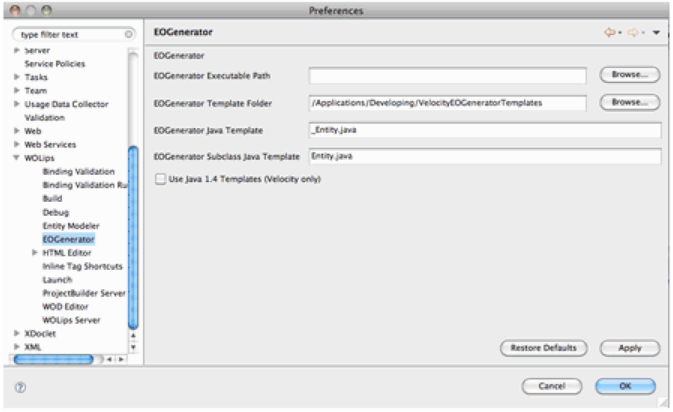
2.6.3.4. Install Velocity EOGenerator

This is an external Java program that helps programmers generate the java source for their EOs. Download the templates from `http://webobjects.mdimension.com/wolips/EOGenerator/Velocity%20EOGenerator%20Templates/` and place them in the `/Applications/Developing/VelocityEOGeneratorTemplates` folder.

Inside this template folder you will find two standard templates: `_Entity.java` and `Entity.java`. In Eclipse, select Preferences → WOLips → EOGenerator and enter the following information:

- EOGenerator Template Folder: /Applications/Developing/VelocityEOGeneratorTemplates
- EOGenerator Java Template: _Entity.java
- EOGenerator Subclass Java Template: Entity.java

You are now ready to create your first WebObjects example with EspressReport.



EOGenerator

EOGenerator Executable Path:

EOGenerator Template Folder: /Applications/Developing/VelocityEOGeneratorTemplates

EOGenerator Java Template: _Entity.java

EOGenerator Subclass Java Template: Entity.java

□ Use Java 1.4 Templates (Velocity only)

2.6.3.5. Create new WO Application

In Eclipse, select the WOLips perspective. This is indicated in the upper right corner of the window. If you cannot find the WOLips button, click on the double right arrow at the very edge of the window and choose *Other*, then select the WOLips perspective from the list and click *Ok*.

In WOLips perspective, **CTRL**+**Click** in the Package Explorer tab and choose New > Other. This will bring up a *Select a..* window. Choose *WedObjects Application* under the *WOLips* folder from the popup dialog. When the *New WebObjects Project* dialog opens, enter **`EspressReportApp`** as the name of the project, check the *Use default location* box to use the default workspace location, and click *Next* to create the project.

Clear the Base Package and Component Package fields in the next window and click *Finish*. You should see an *EspressReportApp* project created in your Package Explorer. Expand your project and verify that it contains the following: `Sources`, `JRE System Library`, `build`, `Components` (and under `Components`, `Main WO`) and `build.xml`.

Also, expand Sources and then (default package) and edit each of the java files present (`Application.java`, `DirectAction.java`, `Main.java` and `Session.java`) to remove the `package .` and `import .Main` lines. After editing, save the files.

2.6.3.6. Edit Main WO Component

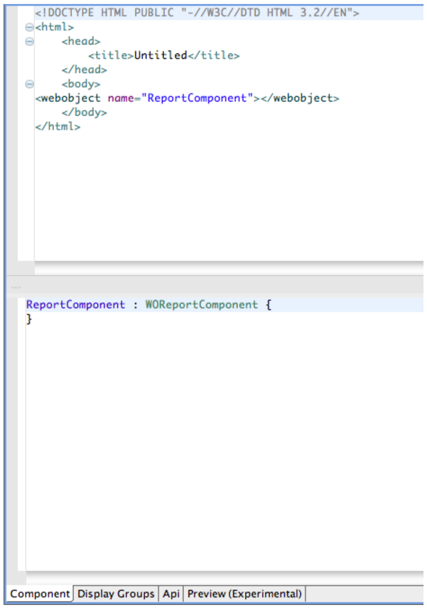
In the Package Explorer, select `Components/Main WO` from the tree. Click on the *Component* tab at the bottom of the main editor and in the HTML area (the top panel) add:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
```

```
    <html>
        <head>
            <title>Untitled</title> </head>

        <body>
            <webobject name = "ReportComponent"></webobject> </body>

    </html>
```

Click in the bottom panel of the Component editor to edit the WOD file and enter

```
ReportComponent : WOReportComponent {
}
```



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
    <head>
        <title>Untitled</title>
    </head>
    <body>
<webobject name="ReportComponent"></webobject>
    </body>
</html>
```

```
ReportComponent : WOReportComponent {
}
```

Save the modifications.

2.6.3.7. Create new WO Component

Select `EspressReportApp/Components` from Package Explorer and **CTRL**+**Click** to show a pop up dialog.

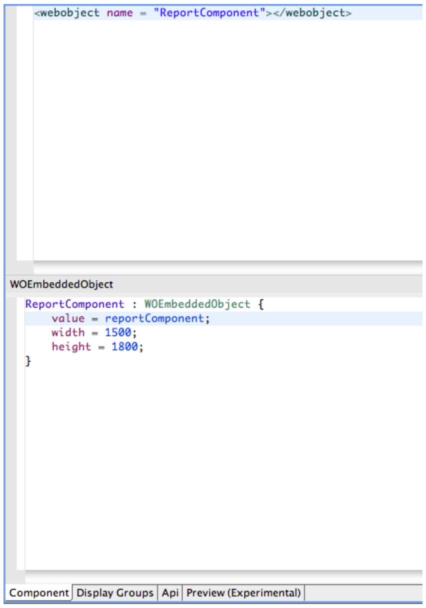
Select New → Other and then select *WOComponent* under *WOLips*. The *New WebObjects Component* dialog opens. Enter **`WOReportComponent`** as the component name. Deselect *Create HTML contents* and *Create api file* and click *Finish*.

You should see a new `WOReportComponent.java` in your source folder and a `WOReportComponent WO` folder at the Components folder. Select `WOReportComponent WO` from Package Explorer tree and switch to the *component* tab. Type in the following in *HTML* panel (UPPER PANEL):

```
<webobject name = "ReportComponent"></webobject>
```

Then, type the following in the *WOD definition* window (LOWER PANEL) and save:

```
ReportComponent : WOEmbeddedObject {

    value = reportComponent;
    width = 1500;
    height = 1800;

}
```



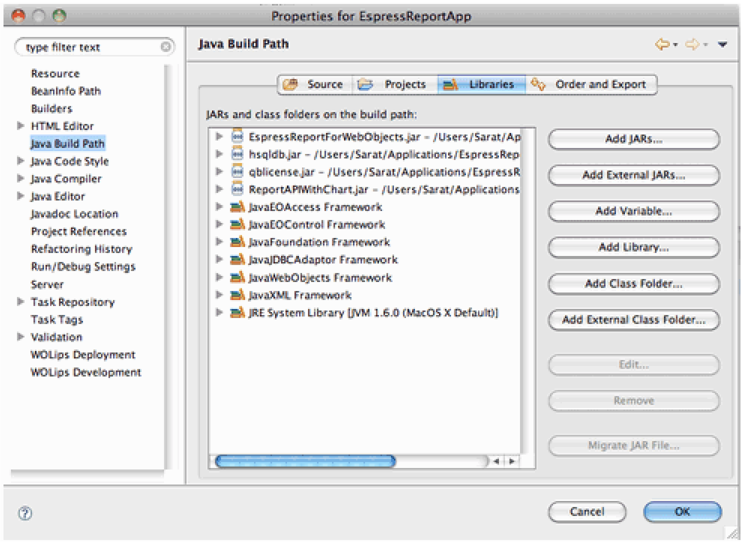
```
<webobject name = "ReportComponent"></webobject>
```

WOEmbeddedObject

```
ReportComponent : WOEmbeddedObject {
    value = reportComponent;
    width = 1500;
    height = 1800;
}
```

2.6.3.8. Modify build path

CTRL+**Click** on the `EspressReportApp` folder from Package Explorer tree and select *Properties* from the popup menu. Then click on *Java Build Path* from the left side in the new popup dialog. Select *Add External Jars* and browse and add `EspressReportForWebObjects.jar`, `ReportAPIWithChart.jar`, `hsqldb.jar` and `qblicense.jar` from the `<EspressReport Install Directory>/lib` folder.



Properties for EspressReportApp

Java Build Path

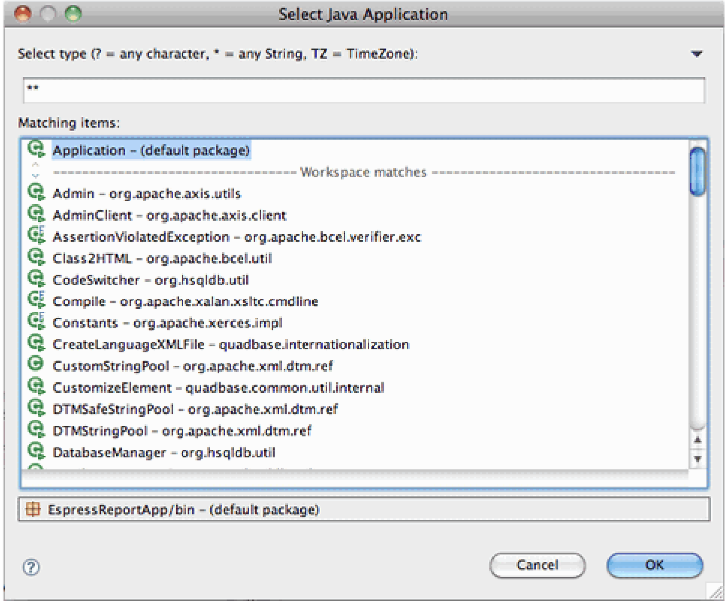
JARs and class folders on the build path:

- EspressReportForWebObjects.jar - /Users/Sarat/Ap
- hsqldb.jar - /Users/Sarat/Applications/EspressRep
- qblicense.jar - /Users/Sarat/Applications/EspressR
- ReportAPIWithChart.jar - /Users/Sarat/Applications
- JavaEOAccess Framework
- JavaEOControl Framework
- JavaFoundation Framework
- JavaJDBCAdaptor Framework
- JavaWebObjects Framework
- JavaXML Framework
- JRE System Library [JVM 1.6.0 (MacOS X Default)]

Drag `ParameterizedNSArray.java` and `WOReportComponent.java` from the `<EspressReport Install Directory>/help/examples/WebObjects` folder to `EspressReportApp/Sources/(default package)` in package explorer.

Click on the `WOReportComponent.java` file underneath the Sources folder and modify path to sample.dat to point to `<EspressReport Install Directory>/help/examples/DataSources/text/sample.dat` .

CTRL+**Click** on the project name and select Run As → WOApplication. Choose *Application - (default package)* in the *Select Java Application* dialog and click *OK*. Then you should see a report which displayed in your default web browser.



Select Java Application

Select type (? = any character, * = any String, TZ = TimeZone):

**

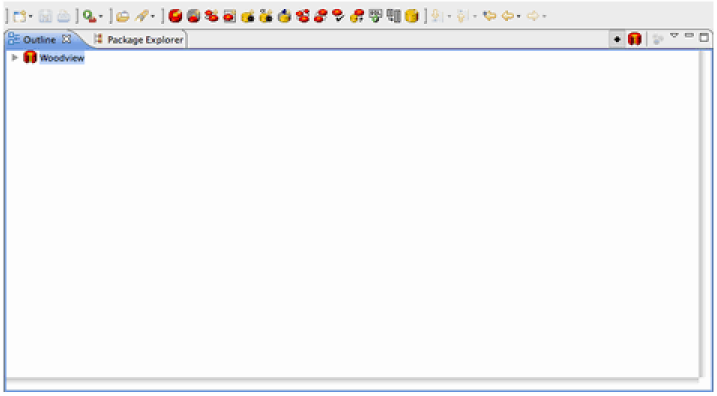
Matching items:

Application – (default package)
------------------------------- Workspace matches -------------------------------
Admin – org.apache.axis.utils
AdminClient – org.apache.axis.client
AssertionViolatedException – org.apache.bcel.verifier.exc
Class2HTML – org.apache.bcel.util
CodeSwitcher – org.hsqldb.util
Compile – org.apache.xalan.xsltc.cmdline
Constants – org.apache.xerces.impl
CreateLanguageXMLFile – quadbase.internationalization
CustomStringPool – org.apache.xml.dtm.ref
CustomizeElement – quadbase.common.util.internal
DTMSafeStringPool – org.apache.xml.dtm.ref
DTMStringPool – org.apache.xml.dtm.ref
DatabaseManager – org.hsqldb.util

EspressReportApp/bin – (default package)

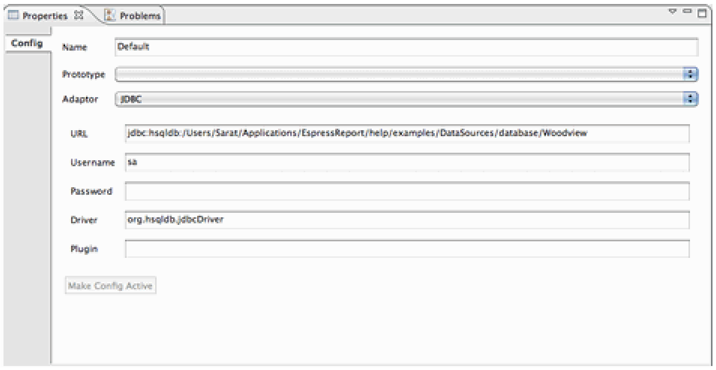
2.6.3.9. Create EOModel using Entity Modeler

CTRL+Click on `EspressReportApp/build/EspressReportApp.woa/Contents/Resources` and select *New: Other*. Select *EOModel* from *WOLips* folder in the popup dialog. Enter **`WoodView`** for the EOModel name and choose *JDBC* for Adaptor type. Please make sure *Use EOGenerator File* is checked and click *Finish*. Switch to the Entity Modeler Prespective.



2.6.3.10. Setup Database Information

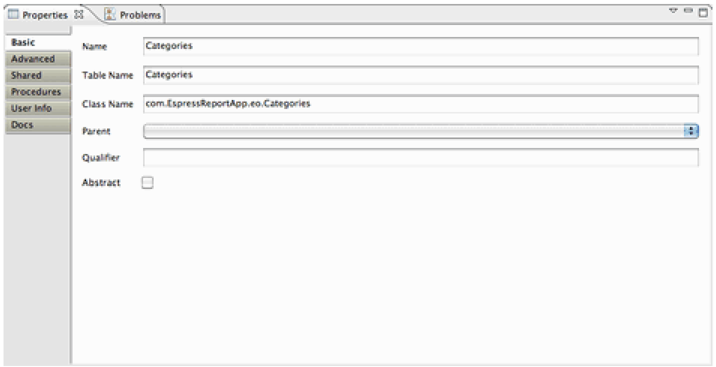
In the Entity Modeler, select *Outline* from the left tabbed panel, expand *WoodView* folder. Select *Default* and click on *Properties* in the bottom panel below. Fill in the username (**`sa`**) and password (leave it blank). For URL, enter *jdbc:hsqldb:<EspressReport Install Directory>help/examples/DataSources/database/woodview* and for Driver, enter `org.hsqldb.jdbcDriver`.



Name: Default
Prototype:
Adaptor: JDBC
URL: jdbc:hsqldb:/Users/Sarat/Applications/EspressReport/help/examples/DataSources/database/Woodview
Username: sa
Password:
Driver: org.hsqldb.jdbcDriver
Plugin:

2.6.3.11. Create Entities

Click on *New Entity* in the toolbar button to create a new entity. Click on the *Basic* Tab in the Properties pane and change the new entity's name to **`Orders`**, the table name to **`Orders`** and set the class name to **`com.EspressReportApp.eo.Orders`**.



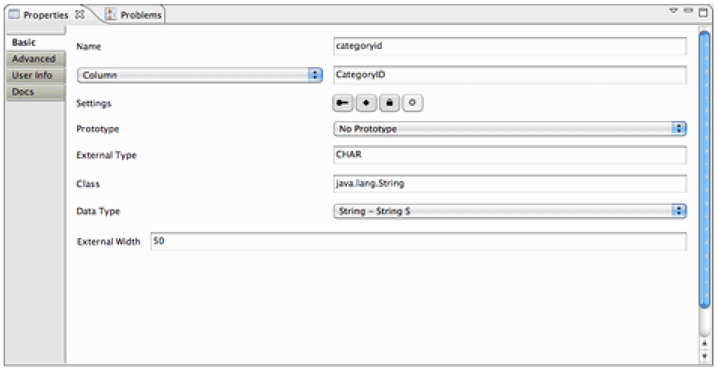
Name: Categories
Table Name: Categories
Class Name: com.EspressReportApp.eo.Categories
Parent:
Qualifier:
Abstract: □

```
Name: OrderDetails
Table Name: Order_Details
Class Name: com.EspressReportApp.eo.OrderDetails

Name: Categories
Table Name: Categories
Class Name: com.EspressReportApp.eo.Categories

Name: Products
Table Name: Products
Class Name: com.EspressReportApp.eo.Products
```

2.6.3.12. Create attribute definitions

Select the `Orders` entity and click on the *New Attribute* from the toolbar button. After a new attribute is created, select the Properties pane and enter the following:

```
Name: orderid
Column: OrderID
External Type: INTEGER
Data Type: Integer - Integer
Key icon (Primary Key): checked
Diamond icon (Class Property): checked
Lock icon (Used for Locking): checked
```



Name: categoryid

Column: CategoryID

Settings

Prototype: No Prototype

External Type: CHAR

Class: java.lang.String

Data Type: String – String S

External Width: 50

Repeat the above steps to create the following attributes for the `Orders` entity:

```
Name: shipcity
Column: ShipCity
External Type: CHAR
Data Type: String - String S
External Width: 50

Key icon (Primary Key): unchecked
Diamond icon (Class Property): checked
Lock icon (Used for Locking): checked

Name: shipstate
Column: ShipState
External Type: CHAR
Data Type: String - String S
External Width: 50
Key icon (Primary Key): unchecked
Diamond icon (Class Property): checked
Lock icon (Used for Locking): checked
```

Add the following attributes for the `OrdersDetails` entity:

```
Name: orderid
Column: OrderID
External Type: INTEGER

Data Type: Integer - Integer
Key icon (Primary Key): unchecked
Diamond icon (Class Property): checked
Lock icon (Used for Locking): checked

Name:  productid
Column: ProductID
External Type: INTEGER
Data Type: Integer - Integer
Key icon (Primary Key): checked
Diamond icon (Class Property): checked
Lock icon (Used for Locking): checked

Name: quantity
Column: Quantity
External Type: INTEGER
Data Type: Integer - Integer
Key icon (Primary Key): unchecked
```

```
Diamond icon (Class Property): checked

Lock icon (Used for Locking): checked
```

Select the entity `Categories` and create the following attributes:

```
Name: categoryid
Column: CategoryID
External Type: CHAR
Data Type: String - String S
External Width: 50
Key icon (Primary Key): checked
Diamond icon (Class Property): checked
Lock icon (Used for Locking): checked

Name: categoryname
Column: CategoryName
External Type: CHAR
Data Type: String - String S
External Width: 50 Key icon (Primary Key): unchecked

Diamond icon (Class Property): checked
Lock icon (Used for Locking): checked
```

Add the following attributes for the `Products` entity:

```
Name: categoryid
Column: CategoryID
External Type: CHAR
Data Type: String - String S
External Width: 50
Key icon (Primary Key):un checked
Diamond icon (Class Property): checked
Lock icon (Used for Locking): checked

Name: productid
Column: ProductID
External Type: INTEGER
Data Type: Integer - Integer

Key icon (Primary Key):checked
Diamond icon (Class Property): checked
Lock icon (Used for Locking): checked

Name: productname
Column: ProductName
External Type: CHAR
Data Type: String - String S
External Width: 50
Key icon (Primary Key):un checked
Diamond icon (Class Property): checked
Lock icon (Used for Locking): checked

Name: unitsinstock
Column: UnitsInStock
External Type: INTEGER
Data Type: Integer - Integer
Key icon (Primary Key):un checked
Diamond icon (Class Property): checked
```

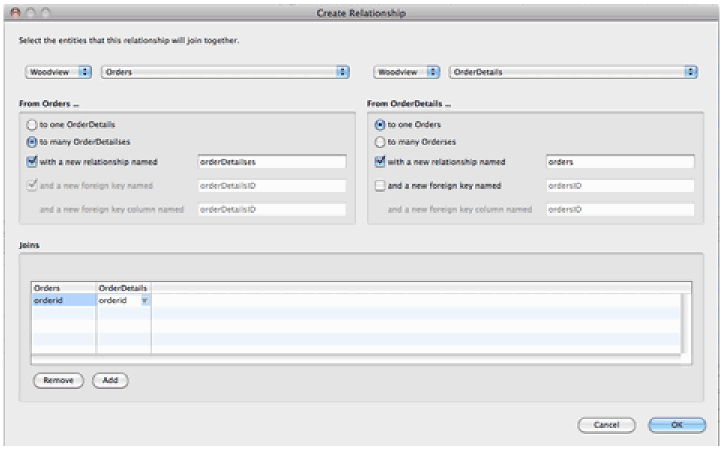
```
Lock icon (Used for Locking): checked
```

2.6.3.13. Create Relationships Between Entities

2.6.3.13.1. Create Relationship Between "Orders" and "OrderDetails"

CTRL+**Click** on `Orders` and select *New Relationship* from the popup menu. The Create New Relationships dialog opens. Map `Woodview.Orders` and `Woodview.OrderDetails` together.

In left panel *From Orders…*, select *to many OrderDetails*, check the checkbox for *write a new relationship named* and uncheck the checkbox for *foreign key*. In right panel *From OrderDetails…*, select *to one Orders*, check the checkbox for *write a new relationship named* and uncheck the checkbox for *foreign key*. At the *Joins* panel, map `Orders.orderid` and `OrderDetails.orderid` together and click *OK*.



Create Relationship

Select the entities that this relationship will join together.

Woodview | Orders

Woodview | OrderDetails

From Orders …

○ to one OrderDetails
● to many OrderDetailses
☑ with a new relationship named: orderDetailses
☑ and a new foreign key named: orderDetailsID
and a new foreign key column named: orderDetailsID

From OrderDetails …

● to one Orders
○ to many Orderses
☑ with a new relationship named: orders
□ and a new foreign key named: ordersID
and a new foreign key column named: ordersID

Joins

Orders	OrderDetails
orderid	orderid

2.6.3.13.2. Create Relationship Between "OrderDetails" and "Products"

CTRL+**Click** on *OrderDetails* and select *New Relationship* from the pop up menu. Map `Woodview.OrderDetails` and `Woodview.Products` together.

In left panel *From OrderDetails…*, select *to one Products*, check the checkbox for *write a new relationship named* and uncheck the checkbox for *foreign key*. In right panel *From Products…*, select *to many OrderDetails*, check the checkbox for *write a new relationship named*, and uncheck the checkbox for *foreign key*.

At the *Joins* panel, map `OrderDetails.productid` and `Products.productid` together and click *OK*

2.6.3.13.3. Create relationship between "Products" and "Categories"

CTRL+**Click** on *Products*" and select *New Relationship* from the popup menu. Map `Woodview.Products` and `Woodview.Categories` together.

In left panel *From Products…*, select *to one Categories*, check the checkbox for *write a new relationship named* and uncheck the checkbox for *foreign key*. In right panel *From Categories…*, select *to many Products*, check the checkbox for *write a new relationship named* and uncheck the checkbox for *foreign key*.

At the *Joins* panel, map `Products.categoryid` and `Products.categoryid` together and click *OK*

2.6.3.14. Verify and Save

After creating the entities, attributes and relationships, click the *Verify Model* button from the toolbar. The window should come up with no errors message.

Select the *Save* button from the toolbar to save the view model.

2.6.3.15. Create Java Code

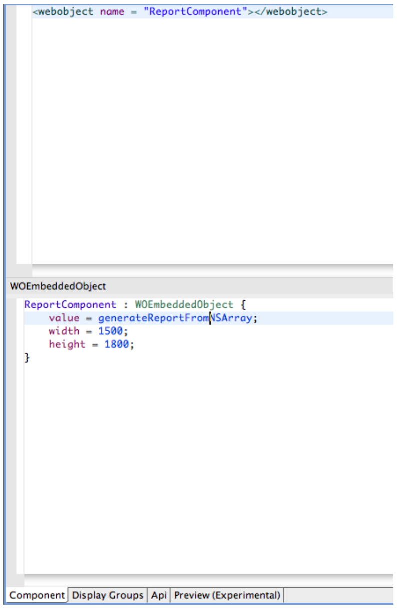
Delete the `woodview.ecogen` file (if it exists) from `EspressReportApp/build/EspressReportApp.woa/Contents/Resources`. **CTRL**+**Click** on the `WoodView` EOModel from the package explorer and select WOLips Tools → Create EOGenerater File… from the pop up menu. **CTRL+Click** on the generated `woodview.ecogen` and select *EOGenerate* from the popup menu. It should return without any errors and you should see new java files created under the Sources folder.

2.6.3.16. Running the Test Case with NSArray

Select *WOReportComponent WO* from *Package Explorer* tree and switch to the *Component* tab. In the *WOD definition* window (LOWER PANEL), replace the `value` attribute to `generateReportFromNSArray`

```
ReportComponent : WOEmbeddedObject {

    value = generateReportFromNSArray;
    width = 1500;
    height = 1800;

}
```

CTRL+Click the project, *EspressReportApp*, in the Package Explorer pane and select Run As → WOApplication from the popup menu. As before, choose *Application - (default package)* from the list. The report should pop up from your default web browser.



```
<webobject name = "ReportComponent"></webobject>
```

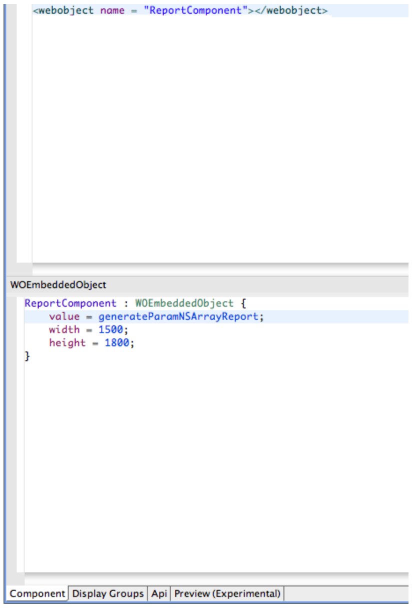
```
ReportComponent : WOEmbeddedObject {
    value = generateReportFromNSArray;
    width = 1500;
    height = 1800;
}
```

2.6.3.17. Running the Test Case for Parameterized Report using NSArray

Select *WOReportComponent WO* from *Package Explorer* tree and switch to the *Component* tab. In the *WOD definition* window (LOWER PANEL), replace the value attribute to `generateParamNSArrayReport`

```
ReportComponent : WOEmbeddedObject {

    value = generateParamNSArrayReport;
    width = 1500;
    height = 1800;

}
```

CTRL+Click the project, *EspressReportApp*, in the *Package Explorer* pane and select Run As → WOApplication from the popup menu. Again, choose *Application - (default package)* from the list. You will then get prompted by the parameter input dialog. Type in **`NY`** for the city and click *OK*. The report should popup from your default web browser.



```
<webobject name = "ReportComponent"></webobject>
```

WOEmbeddedObject

```
ReportComponent : WOEmbeddedObject {
    value = generateParamNSArrayReport;
    width = 1500;
    height = 1800;
}
```

2.6.4. Notes

All files (those in the `EspressReportsForWebObjects.jar` file and the two additional files) are located under `<EspressReport Install Directory>/help/examples/WebObjects` folder.

Note

By default, EspressManager is not used. If you want to have EspressManager up and running, you will need to modify the source code to suit your needs. If EspressManager is not running for drill-down reports, reports with a sub-report, or reporta with A chart, you will need to specify the path using the following methods:

- `report.setDrillDownPath(...);`
- `report.setChartPath(...);`
- `report.setSubReportPath(..);`

2.7. Deployment

2.7.1. Introduction

EspressReport consists of several components: Report Designer, Report Viewer, Page Viewer, Chart Viewer, Scheduler, EspressManager, and Report API. Report Designer is used to create reports in a GUI environment. Report Viewer is an applet used to view already created reports (saved in .rpt/.xml/.pak format). Page Viewer is an applet that is used to view a report page by page (saved in .page format). Chart Viewer is an applet that is used to view a chart (saved in .cht/.tpl format). Report API is used to create reports programmatically. Scheduler is used to schedule exports for reports. Finally, EspressManager serves as a user administrator and handles data and data buffering.

Note

`qblicense.jar` **must** be included in the CLASSPATH or in the HTML page as an archive (along with any other additional jars) in order to run any code.

While the Report Designer and Scheduler **must** be used in conjunction with EspressManager, an option is provided for Report Viewer, Page Viewer, Chart Viewer, and Report API to work without connecting to EspressManager.

You must specify the information about how to connect to EspressManager. If EspressManager is running as an application, you can use API methods to specify the IP address/machine name where EspressManager is located and the port number EspressManager is listening on.

You use the following two API methods to set the connection information:

```
static void setServerAddress(java.lang.String address);
static void setServerPortNumber(int port);
```

For example, the following lines of code:

```
QbReport.setServerAddress("someMachine");
QbReport.setServerPortNumber(somePortNumber);
```

will connect to EspressManager running on `someMachine` and listening on `somePortNumber`.

> **Note**
>
> If EspressManager connection information is not specified, the code will attempt to connect to EspressManager on the local machine and listening to the default port number (22071).

If EspressManager is running as a servlet, you can use the following methods:

```
public static void useServlet(boolean b);
public static void setServletRunner(String comm_URL);
public static void setServletContext(String context);
```

For example, the following lines of code:

```
QbReport.useServlet(true);
QbReport.setServletRunner("http://someMachine:somePortNumber");
QbReport.setServletContext("EspressReport/servlet");
```

will connect to EspressManager running at `http://someMachine:somePortNumber/EspressReport/servlet`.

Please note that these methods exist in the `QbChart`, `QbReportDesigner`, `QbChartDesigner`, `QbScheduler`, and `ScheduleModifier` classes as well.

For Page Viewer and Report Viewer (in an applet), you can set the connection information to the EspressManager by passing in the following parameters:

```
server_address (server address), server_port_number (port number)
```

The above parameters are set if EspressManager is running as an application. If EspressManager is running as a servlet, the following parameters are used:

```
comm_protocol (servlet), comm_URL (machine name/IP address and port number),
 servlet_context (servlet context)
```

Described, in sections below, are the various deployment scenarios that you may encounter and the consequences of each scenario.

2.7.2. Deploying with EspressManager

This section deals with deploying report components that work in conjunction with EspressManager.

To learn more about the interaction, please refer to the Interaction with EspressManager section under the EspressReport Report API Overview chapter.

> **Note**
>
> Any references to a data source or a report file, using the relative URL reference (i.e. `help\examples\data\sample.dat`) are relative to the directory from where EspressManager is running in.

2.7.2.1. Report Designer

As mentioned before, Report Designer must be used in conjunction with EspressManager. Report Designer can be called using the API. Note that in this scenario, you will have to add ReportDesigner.jar in your CLASSPATH and place a copy of the reportimages (`EspressReport/reportimages`), images (`EspressReport/images`), and the backgroundImages (`EspressReport/backgroundImages`) directories under the working directory of the .class file.

Instead of copying the directories relative to the working directory when invoking the Report Designer from API, an option to set the location `images/`, `reportimages/` and `backgroundImages/` directory is also available. Simply use a `QbReportDesigner` constructor with the parameter `imagesPath`. For more detail about the `QbReportDesigner` constructor or the `imagesPath` parameter, please refer to the EspressReport Java API Documentation.

To connect to EspressManager running as an application, you use the following two API methods in `QbReportDesigner` to set the connection information:

```
static void setServerAddress(java.lang.String address);
static void setServerPortNumber(int port);
```

To connect to the EspressManager running as a servlet, you use the following three API methods in `QbReportDesigner` to set the connection information:

```
static void useServlet(boolean b);
static void setServletRunner(String comm_URL);
static void setServletContext(String context);
```

Please note that specifying the connection information to EspressManager must be made before calling any `QbReportDesigner` constructors.

2.7.2.2. Report Viewer

Report Viewer can be used in an applet or application environment to show a report saved in a `.rpt/.xml/.pak` file (either generated by Report Designer or Report API).

To deploy Report Viewer in an applet environment, the `ReportViewerWithChart.jar` must be included in the HTML file as the archive file.

To use Report Viewer, construct an HTML page with the proper applet code (for more details, please refer to the Report Viewer chapter). Note that any relative references to the report location and/or data are relative to the directory from where EspressManager has been started.

To connect to EspressManager running as an application, you must pass the following parameters to the Report Viewer applet:

```
server_address (server address), server_port_number (port number)
```

To connect to EspressManager running as a servlet, you pass the following parameters to the Report Viewer applet:

```
comm_protocol (servlet), comm_URL (machine name/IP address and port number),
 servlet_context (servlet context)
```

2.7.2.3. Page Viewer

Page Viewer can be used in an applet or application environment to show a particular page of a report at a time. These PAGE files are of `.page` file extension and are generated by Report Designer or the Report API.

To deploy Page Viewer in an applet environment, the `PageViewer.jar` must be included in the HTML file as the archive.

To use Page Viewer, construct an HTML page with the proper applet code (for more details, please refer to the Page Viewer chapter). Note that any relative references to the report location and/or data are relative to the directory from where EspressManager has been started.

To connect to EspressManager running as an application, you pass the following parameters to the Report Viewer applet:

```
server_address (server address), server_port_number (port number)
```

To connect to EspressManager running as a servlet, you must pass the following parameters to the Report Viewer applet:

```
comm_protocol (servlet), comm_URL (machine name/IP address and port number),
 servlet_context (servlet context)
```

2.7.2.4. Chart Viewer

Deploying Chart Viewer is similar to deploying Report Viewer. Chart Viewer can be used in an applet or application environment to show a chart saved in a `.cht or .tpl` file (either generated by Designer or API).

To deploy Chart Viewer in an applet environment, `EspressViewer.jar` must be included in the HTML file as the archive.

To use Chart Viewer, construct an HTML page with the proper applet code (for more details, please refer to start the Chart Viewer chapter). Note that any relative references to the chart location and/or data are relative to the directory from where EspressManager has been started.

To connect to EspressManager running as an application, you must pass the following parameters to the Report Viewer applet:

```
server_address (server address), server_port_number (port number)
```

To connect to EspressManager running as a servlet, you must pass the following parameters to the Report Viewer applet:

```
comm_protocol (servlet), comm_URL (machine name/IP address and port number),
 servlet_context (servlet context)
```

2.7.2.5. Report API

Report API can be used in either an applet environment or as an application. Report API can also be used in a servlet or jsp environment to generate server-side reports.

To deploy Report API, `ReportAPIWithChart.jar`, and `ExportLib.jar` must be in the CLASSPATH (for application and servlet/jsp environment) or included in the HTML file (for an applet).

To connect to EspressManager running as an application, you would use the following methods in `QbReport` to set the connection information:

```
static void setServerAddress(java.lang.String address);
static void setServerPortNumber(int port);
```

To connect to EspressManager running as a servlet, you would use the following methods in `QbReport` to set the connection information:

```
static void useServlet(boolean b);
static void setServletRunner(String comm_URL);
static void setServletContext(String context);
```

Please note that specifying the connection information to EspressManager must be made before calling any `QbReport` constructors.

2.7.2.6. Scheduler

A scheduler has also been included with EspressReport. This Scheduler can be used to export reports from their source files at specific intervals. For more information on the Scheduler, please look in the Scheduler chapter.

Like Report Designer, Scheduler must also be used in conjunction with EspressManager. Scheduler can be called using the API. Note that in this scenario, you will have to add `Scheduler.jar` in your CLASSPATH.

To connect to EspressManager running as an application, you use the following two API methods in `QbScheduler`/`ScheduleModifier` to set the connection information:

```
static void setServerAddress(java.lang.String address);
static void setServerPortNumber(int port);
```

To connect to the EspressManager running as a servlet, you use the following three API methods in `QbScheduler`/`ScheduleModifier` to set the connection information:

```
static void useServlet(boolean b);
static void setServletRunner(String comm_URL);
static void setServletContext(String context);
```

Please note that specifying the connection information to EspressManager must be made before calling any `QbScheduler`/`ScheduleModifier` constructors.

2.7.3. Deploying without EspressManager

This section deals with deploying components that work independent of EspressReport. To learn more about the interaction, please refer to the Interaction with EspressManager section under the EspressReport API Overview chapter.

Generating reports/charts independently of EspressManager results in a slightly better performance as the interaction of EspressManager and the report/chart component is removed.

An important consideration is that any references to a data source or a report source, that is relative (i.e. for example `help\examples\data\sample.dat`), are relative to the working directory from where the HTML file (in the case of an applet) or the class file (in an application) is being executed. In a servlet/jsp environment, the working directory typically differs from the directory where the class resides. To find out the working directory, have the following line of code in your servlet/jsp:

```
System.out.println("The working directory is " +
 System.getProperty("user.dir") + ". ");
```

By having the above line of code, and executing the servlet/jsp, the working directory is ascertained and the data files and report templates can be moved to locations relative to the working directory, as dictated by the code.

2.7.3.1. Report Viewer

To deploy Report Viewer, `ReportViewerWithChart.jar` must be included in the HTML file as the archive. Depending on what functionality you are using, you have to include additional jars as necessary.

Report Viewer can be used independent of EspressManager with the addition of one more parameter in the applet code:

```
<param name="EspressManagerUsed" value="false">
```

Note that any relative references to the report location and/or data are relative to the directory where the HTML file is located. For example, if the data source specified in the .rpt file is `help\examples\data\sample.dat` and if the HTML file is located in `D:\EspressReport\TestApplet`, then for the report to be displayed successfully, `help\examples\data\sample.dat` must exist within `D:\EspressReport\TestApplet` (i.e., `D:\EspressReport\TestApplet\help\examples\data\sample.dat` must exist).

2.7.3.2. Page Viewer

To deploy Page Viewer, `PageViewer.jar` must be included in the HTML file as the archive. Depending on what functionality you are using, you have to include additional jars as necessary.

Page Viewer can be used independent of EspressManager with the addition of one more parameter in the applet code:

```
<param name="EspressManagerUsed" value="false">
```

Note that any relative references to the report location and/or data are relative to the directory where the HTML file is located. For example, if the data source specified in the .rpt file is `help\examples\data\sample.dat` and if the HTML file is located in `D:\EspressReport\TestApplet`, then for the report to be displayed successfully, `help\examples\data\sample.dat` must exist within `D:\EspressReport\TestApplet` (i.e., `D:\EspressReport\TestApplet\help\examples\data\sample.dat` must exist).

Also note that when using Page Viewer without connectng to EspressManager, you cannot pass in a template directly. You must pass in the page file instead.

2.7.3.3. Chart Viewer

To deploy Chart Viewer, `EspressViewer.jar` must be included in the HTML file as the archive. Depending on what functionality you are using, you have to include additional jars as necessary.

Chart Viewer can be used independent of EspressManager with the addition of one more parameter in the applet code:

```
<param name="EspressManagerUsed" value="false">
```

Note that any relative references to the report location and/or data are relative to the directory where the HTML file is located. For example, if the data source specified in the .rpt file is `help\examples\data\sample.dat` and if the HTML file is located in `D:\EspressReport\TestApplet`, then for the report to be displayed successfully, `help\examples\data\sample.dat` must exist within `D:\EspressReport\TestApplet` (i.e., `D:\EspressReport\TestApplet\help\examples\data\sample.dat` must exist).

2.7.3.4. Report API

To deploy Report API, `ReportAPIWithChart.jar` and `ExportLib.jar` must be in the CLASSPATH (for application and servlet/jsp environment) or included in the HTML file (for an applet).

Report API can also be used without connecting to EspressManager by adding the line of code below:

```
QbReport.setEspressManagerUsed(false);
```

This method **must** be called prior to any `QbReport` instantiation.

As with Report Viewer, any relative references to the report location and/or data are relative to the working directory from where class file is being executed. For example, if the data source specified is `help\examples\data\sample.dat` and if the class file is located in `D:\EspressReport\TestApplication`, then for the report to be displayed successfully, `help\examples\data\sample.dat` must exist within `D:\EspressReport\TestApplication` (i.e., `D:\EspressReport\TestApplicationhelp\examples\data\sample.dat` must exist).

Note that any charts created in Report Designer will contain a **relative** path to the template. Therefore, the directory structure must be mirrored under the working directory (i.e., create a `chart` sub-directory with the template under the working directory), or open the template and change the path with an absolute path using the following code:

```
// Get the Chart from the footer
ReportChartObject obj =
 (ReportChartObject)report.getTable().getFooter().getData(0);
String originalLocation = obj.getText();
String newLocation = new String("http://128.0.0.1/EspressEnterprise/
EspressReport/test/chart/chart/EspressReport1.1_0.tpl");
obj.setText(newLocation);
```

In a servlet/jsp environment, the location of the directory with the class file and the working directory might differ. In that case, any relative references are with respect to the working directory. For example, if the servlet class file was in `D:\<some servlet engine>\webapp\examples\quadbase\someservlet.class` and the data source was `help\examples\data\sample.dat` and the working directory was `D:\<some servlet engine>\webapp`, then `help\examples\data\sample.dat` must exist under `D:\<some servlet engine>\webapp` (i.e., `D:\<some servlet engine>\webapp\help\examples\data\sample.dat` must exist) for the report to be generated successfully.

Note that in the API, methods are also available that allow you to specify the location of the chart, drilldown, image, and subreport directories where certain files may be located. For instance, given the existence of a template, `EspressReport1.1_0.tpl`, in the report, you can specify the location this template by using the following code:

```
report.setChartPath("d:/EspressReport/Chart/");        // where report is an
 object of type QbReport
```

The above code sets the location of the template file thus avoiding the need to have a chart sub-directory under the working directory of your class file.

The following methods work similarly:

```
QbReport.setDrillDownPath(String directory);
QbReport.setImagePath(String directory);
QbReport.setSubReportPath(String directory);
```

Alternative to deploying your report in RPT format, you can also pack your report in PAK format and deploy it as a PAK file. This alternative simplifies the deployment procedures by including all the chart, sub-report, and drill-down files that are associated with your RPT file. To do this in the API, simply use the following method to pack your report file:

```
pack(java.lang.String filename, java.lang.String subReportPath,
 java.lang.String drillDownPath, java.lang.String chartPath);
```

During deployment onto your destination machine, simply open the report using the `QbReport` constructor with your PAK file name in the template parameter.

2.7.4. Deploying in a Non-Windows Environment

EspressReport is a Pure Java tool for generating reports. As such, it does not contain graphical libraries for generating colors and fonts and other AWT information. For that, Java relies on the system's (on which the reports are being generated) libraries for providing that information. Thus an environment capable of providing AWT information and a graphics card (for exporting to static formats) are required.

In a Windows environment, nothing extra needs to be done to set up such an environment as it already exists. A GUI interface is already running and a graphics card already exists.

For non-Windows environments (such as Unix and Linux), such is usually not the case. You need to have X or some form of X running on such systems and point the display to the machine running X (such as running the command export DISPLAY=192.168.0.16:0.0 in a shell). For best performance, Quadbase recommends running X on the machine (or setting the DISPLAY to point to another machine running X). However, if that is not an acceptable solution, there are alternative solutions available.

2.7.4.1. Xvfb (X Virtual Frame Buffer)

Xvfb is an X server that can run on machines with no display hardware and no physical input devices. It emulates a dumb terminal framebuffer using virtual memory.

The primary use of this server is intended to be server testing. Xvfb is also limited in the number of colors and fonts it can handle.

Xvfb can be downloaded (there are different version available depending on the system) and then run. After Xvfb is running, the `DISPLAY` variable needs to be set, and then passed to the application (or the servlet engine).

2.7.4.2. JVM 1.5+ in Headless Mode

You can use a 1.5+ JVM run-time parameter to run applications using EspressReport. The run-time parameter is `java.awt.headless` and setting it to true results in Java not making an X connection for any graphics information. For example, if you have an application called chartExport that generates charts as jpg's, ordinarily running it would involve making a connection to X. However, using the following command:

```
java -Djava.awt.headless=true exportChart
```

the same application is now run without a connection to X.

2.7.5. Platform Specific Issues

2.7.5.1. AS/400

2.7.5.1.1. Running under NAWT

To run any application using EspressReport in an AS/400 environment using NAWT, you must do the following:

- Set the DISPLAY environment variable to the system name and display number. The display number is the display number of the VNC server.
- Set the XAUTHORITY variable to `/home/VNCProfile/.Xauthority`, where `VNCProfile` is the profile that started the VNC server. Please note that XAUTHORITY need not be set if both the VNC server and the Java virtual machine is running under the same user profile.
- Set the Java system property before running java

  ```
  os400.awt.native=true
  ```

2.7.5.1.2. Running under Headless Mode

To run any application using EspressReport in an AS/400 environment using headless mode, you must do the following:

- Set the Java system property before running java

  ```
  java.awt.headless=true
  ```

- Make sure that your code include the following before calling any `QbChart` constructor:

  ```
  QbChart.setForExportOnly(true);
  ```

2.7.5.2. Linux/Unix

2.7.5.2.1. Running under X

To run any application using EspressReport in a Linux/Unix environment using X, you must do the following:

- Set the DISPLAY environment variable to a X client system name and display number.

2.7.5.2.2. Running under Headless Mode

To run any application using EspressReport in a Linux/Unix environment using headless mode, you must do the following:

- Set the Java system property before running java

  ```
  java.awt.headless=true
  ```

- Make sure that your code include the following before calling any `QbChart` constructor:

  ```
  QbChart.setForExportOnly(true);
  ```

2.A. Parameter Server

2.A.1. Writing a Parameter Server

Report Viewer supports push technology by means of interacting with a parameter server. A programmer can write a parameter server using the class `quadbase.reportdesigner.ReportViewer.ParamServer` to supply updated data continuously for Report Viewer to plot the report.

On the client side, the HTML syntax is:

```
<applet code = "quadbase.reportdesigner.ReportViewer.Viewer.class" width=640
 height=480>
<PARAM name="filename" value="example.rpt">
<PARAM name="ParameterServer" value="machine:portno">
</applet>
```

Once the browser has loaded Report Viewer class and the report data from the web server, Report Viewer attempts to connect to the specific machine and port number identified in the parameter. For security reasons untrusted applets are not allowed to open a connection to other machines. The parameter server can interact with Report Viewer in order to update and manipulate the records held by Report Viewer.

On the server side, a server program should be written to listen to the port number. This server can be written using the class `quadbase.reportdesigner.ReportViewer.ParamServer`.

Upon opening a connection, the server will supply data to Report Viewer.

The constructor for the server is:

```
public ParamServer(DataInputStream in, DataOutputStream out)
```

Where in and out are the socket input and output used to communicate with the server.

`ParamServer` has three basic methods to manipulate the records in Report Viewer:

```
int addRecord(Object record[]);
int deleteRecord(int recordNo);
int updateRecord(Object record[], int recordNo);
```

After a sequence of record updates a final call

```
void repaint();
```

will send a request to Report Viewer to repaint the report using the new data.

The following Java program provides an example of using the parameter server class to generate new reports. Report API for `ParamServer` is provided in the online API documentation. In this example, one field in the twelfth record is simply updated with an integer chosen at random. In a real application, a programmer would have to write the code which enables the parameter server to interact with the data source.

```
import java.io.*;
import java.net.*;
import java.util.*;
import quadbase.reportdesigner.ReportViewer.ParamServer;

// Sample Program to update the data in EspressReport Viewer
// using Parameter Server
public class pserver extends Thread {
protected int port = 1997; // port no for report viewer to connect

protected ServerSocket listen_socket;

public static void fail(Exception e, String msg) {
System.err.println(msg + ":" + e);
System.exit(1);
}

public pserver() {
try {
listen_socket = new ServerSocket(port);
}
catch (IOException e) {
fail(e, "Exception creating server socket");
}
this.start();
}
public void run() {
try {
while(true) {
// create a thread for each connection to handle the
// request
Socket client_socket = listen_socket.accept();
Connection c = new Connection(client_socket);
}
}
catch (IOException e) {
fail(e, "Exception while listening for connections");
}
}
public static void main(String[] args) {
new pserver();
}
}

class Connection extends Thread {
protected Socket client;
protected DataInputStream in;
protected DataOutputStream out;

public Connection(Socket client_socket) {
client = client_socket;
try { // get the input and output stream
in = new DataInputStream(client.getInputStream());
out = new
DataOutputStream(client.getOutputStream());
}
catch (IOException e) {
try {
client.close();
}
catch (IOException e2) {}
return;
}
this.start();
}

public void run() {
ParamServer paramserver;
Random random = new Random(System.currentTimeMillis());
try {
paramserver = new ParamServer(in, out);
System.out.println("Total no of record = " + paramserver.getRecordNo());

// get record 12 from the report viewer
Object rec[] = paramserver.getRecord(12);
for (int i=0; i < 100; i++) {
// update the third field of record 12
rec[3] = new Integer(random.nextInt() % 30);
paramserver.updateRecord(rec, 12);
// tell the report viewer to repaint //after every tenth record is updated
if (i % 10 == 0)
paramserver.repaint();
}
}
catch (IOException e) {}
finally {
try {
client.close();
}
catch (IOException e2) {}
}
}
}
```

Report Viewer would have read an HTML file of the form:

```
<html>
<title>Sample Report</title>
<applet code = "quadbase.reportdesigner.ReportViewer.Viewer.class" width=600
 height=450>
<PARAM name="filename" value="PServer.cht">
<PARAM name="ParameterServer" value=":1997">
</applet>
</html>
```

Here the Java class that Report Viewer reads is the name of a report file `Pserver.rpt` and the next line specifies the machine and port to which Report Viewer should open a connection. EspressManager will then provide information to Report Viewer at regular intervals such as updated records so that Report Viewer can refresh the report it displays.

2.A.1.1. Variables

UPDATE_RECORD	public final static int `UPDATE_RECORD`
INSERT_RECORD	public final static int `INSERT_RECORD`
DELETE_RECORD	public final static int `DELETE_RECORD`
GET_RECORD	public final static int `GET_RECORD`
GET_RECORDNO	public final static int `GET_RECORDNO`
GET_RECORD_-DATATYPE	public final static int `GET_RECORD_DATATYPE`
REPAINT	public final static int `REPAINT`
OK	public final static int `OK`
ERROR	public final static int `ERROR`

2.A.1.2. Constructor

`public ParamServer(DataInputStream in, DataOutputStream out) throws IOException`

Constructor for Parameter Server

2.A.1.3. Methods

updateRecord `public int updateRecord(String rec[], int recordNo) throws IOException`
Update a record

Parameters: `rec` - the record passed in as a string array, see class `quadbase.reportdesigner.ReportAPI.DbData` for the format of the string array.
`recordNo` - the record number to be updated

Returns: OK if success, returns ERROR if no such record number or record type mismatch.

updateRecord `public int updateRecord(Object rec[], int recordNo) throws IOException`
Update a record

Parameters: `rec` - the record passed in as an object array `recordNo` - the record number to be updated

Returns: OK if success, return ERROR if no such record number or record type mismatch.

addRecord `public int addRecord(String rec[]) throws IOException`

Add a record

Parameters: `rec` - the record passed in as string array

Returns: OK if success, returns ERROR if record type mismatch.

addRecord `public int addRecord(Object rec[]) throws IOException`

Add a record

Parameters: `rec` - the record passed in as an object array

Returns: OK if success, returns ERROR if record type mismatch.

deleteRecord `public int deleteRecord(int recordNo) throws IOException`

Delete a record

Parameters: `recordNo` - the record number to be deleted

Returns: OK if success, returns ERROR if record type mismatch.

repaint `public void repaint() throws IOException`

Inform the report viewer to repaint the report

checkRecord `public Boolean checkRecord(Object rec[])`

Auxiliary function to check whether the record type given matches the record type used in report viewer

Parameters: `rec` - the input record to be checked

Returns: true if record match, otherwise return false

getRecordNo `public int getRecordNo() throws IOException`

Get the number of records used

Returns: the number of record

getDataType `public int[] getDataType() throws IOException`

Get the data type of the record used

Returns: array of data types as defined in jdbc/Types.class, the size of the array is equal to the number of field in record

getRecordSize `public int getRecordSize()`

Get the number of fields in record

Returns: number of fields in record

getRecord `public Object[] getRecord(int recordNo) throws IOException`

Get the record

Parameters: recordNo - the record number to be retrieved

Returns: an object array for the record

convertRecord `public Object[] convertRecord(String rec[])`

Convert a record in string array format to Object array format

Returns: object array for the record, null if data type mismatch

2.B. Getting the Report Data

The two appendices in this Chapter (Appendix 2.B - Getting the Report Data and Appendix 2.C - Creating the Report) contain information to help you build a report from scratch (without using the Designer). Keep in mind that this method is typically not recommended as it will make the report more difficult to deploy and maintain. However, under certain circumstances, it may be necessary to construct reports in this manner.

The following is an example of a `QbReport` constructor. Although there are numerous variations available, the typical `QbReport` constructor requires at least three parameters. To create a new report, you must specify the report type, the input data source information, and a mapping of the data columns to the respective columns of the report.

```
QbReport(java.lang.Object parent, int reportType, IResultSet data, ColInfo[]
 mapping, java.lang.String template)
```

The report template is not required (can be null) and the parent object is only required if the report is used in an applet, but the other three parameters must always contain meaningful information. This appendix takes an in depth look at the data parameter and the methods for connecting to different data sources. Appendix 2.C - Creating the Report discusses the report type, the column mapping, and the actual creation of the report. Appendix 2.C - Creating the Report also contains working examples for you to try.

The first step to creating any report is to obtain the data. Data may be retrieved from one of several different types of sources. These sources include:

- Fetch the data from a local or remote database using a JDBC driver. All you need to do is to provide the information of how to connect to the database and the exact form of the query. The report would then fetch the result automatically.
- Read the data from a plain text file, which contains database records in plain text (ASCII) format or XML format. Files of this format can be created by most database programs.
- Pass your own dynamic data, either as an EJB or a class file, through API.
- Pass the dataset as an array in memory.

In the following sections, we will take a close look at each of these data sources and the methods used to obtain data from them.

2.B.1. Data from a Database

One of the powerful features of Report API is its ability to fetch data from a database directly through JDBC. With this approach, all you need to do is to specify the information necessary to connect to the database and the precise form of the SQL statement. Thus, your program can connect to virtually any database provided that a JDBC driver is available.

The database and query information must be stored in a `DBInfo` object prior to constructing the report. Use the following code to instantiate the `DBInfo` object.

```
DBInfo dbinfo = new DBInfo(

      "jdbc:hsqldb:help/examples/DataSources/database/woodview",     // URL
      "org.hsqldb.jdbcDriver",                                       // JDBC
 driver
      "myName",                                                      //
 Username
      "myPassword",                                                  //
 Password
```

```
     "select * from sales");                                        // SQL
 Query
```

Since `DBInfo` implements the interface `IDatabaseInfo`, you can use the `DBInfo` object in the following `QbReport` constructor:

```
QbReport(java.lang.Object parent, int reportType, IDatabaseInfo dbinfo,
 ColInfo[] mapping, java.lang.String template)
```

In some cases, you may not wish to pass the entire database information (such as userid, password, location, driver and the query) to EspressReport. You may want to make the connection yourself and just provide the resultset of the query directly to the API. This can be done by creating a `QueryResultSet` object from the `ResultSet` object you have generated and passing that `QueryResultSet` object as the data source, instead of the `DBInfo` object.

```
// Create a QueryResult object from the ResultSet object resultSet
QueryResultSet queryResultSet = new QueryResultSet(resultSet);

// QueryResultSet implements IResultSet, use it in the following QbReport
 constructor
QbReport(java.lang.Object parent, int reportType, IResultSet data, ColInfo[]
 mapping, java.lang.String template)
```

In the above example, an instance of class `QueryResultSet` is created from the `ResultSet` object passed to the API and this instance is passed to the `QbReport` constructor to create the report object. Please note that in this scenario, the report is **not** refreshable. To update the report, you will need to make the connection to the database again and pass the `ResultSet` object to the report and refresh it yourself.

For a working example using data from a database, see Appendix 2.C.4.2 - Creating the Report.

2.B.1.1. Data from JNDI

EspressReport also allows data to be obtained from a JNDI source. JNDI data sources are treated like database data sources and support the same functionalities. Using a JNDI data source makes it easier to migrate reports between different environments. If the data sources in both environments are setup with the same lookup name, reports can be migrated without any changes.

To connect to a JNDI data source, you must have a data source deployed in your application server. You must also provide the `INITIAL_CONTEXT_FACTORY` and `PROVIDER_URL` to successfully make the connection. Please note that when connecting to a JNDI data source deployed in Tomcat, the `INITIAL_CONTEXT_FACTORY` and `PROVIDER_URL` need not be provided (although EspressManager must be running under the Tomcat environment).

```
// create a DBInfo object with JNDI lookup name and query
String JNDIName = "java:comp/env/jdbc/TestDB";
String query = "select * from testdata";

// The environment hashtable is empty for tomcat because EspressManager is
// running inside Tomcat context. If other application server is used,
// need to set INITIAL_CONTEXT_FACTORY and PROVIDER_URL.
Hashtable env = new Hashtable();
DBInfo dbInfo = new DBInfo(JNDIName, query, env);
```

In the above example, an instance of class `DBInfo` provides the information that the program needs to connect to a JNDI data source and retrieve data. Use the following constructor containing `IDatabaseInfo` to construct the report:

```
public QbReport(Object parent, int reportType, IDatabaseInfo dbinfo,
 ColInfo[] colMap, String templateFile);
```

2.B.2. Data from a Data File (TXT/DAT/XML)

Data for generating a report can also be imported from a data file, which can be a text file as well as an XML formatted file. A report data file (with a `.dat` extension) is a plain text file where each line represents one record, except the first two lines, which contain the data types and field names, respectively. Here is an example of a data file, which contains four records and where each record has three fields. The first line specifies the data type of each field and the second line specifies the field name.

```
string, date, decimal
Name, Day, Volume
"John", "1997-10-3", 32.3
"John", "1997-4-3", 20.2
"Mary", "1997-9-3", 10.2
"Mary", "1997-10-04", 18.6
```

The use of the comma character is optional, but most database programs can output a file in this format (except for the first two lines) when exporting records from a table in text format. As a result, even if you do not have a JDBC driver for your database, you can still quickly produce reports. You can simply export the data from your database in text format and then your program can read it using Report API.

For XML format, the specification is as follows:

```
<EspressData>

      <DataType>string</DataType>
      <DataType>date</DataType>
      <DataType>decimal</DataType>

      <FieldName>Name</FieldName>
      <FieldName>Day</FieldName>
      <FieldName>Volume</FieldName>

      <Row>
            <Data>John</Data>
            <Data>1997-10-3</Data>
            <Data>32.3</Data> </Row>

      <Row>
            <Data>John</Data>
            <Data>1997-4-3</Data>
            <Data>20.2</Data> </Row>

      <Row>
            <Data>Mary</Data>
            <Data>1997-9-3</Data>
            <Data>10.2</Data> </Row>

      <Row>
            <Data>Mary</Data>
            <Data>1997-10-4</Data>
            <Data>18.6</Data> </Row>
```

```
</EspressData>
```

For more details about XML Data, please refer to Section 1.7.1.2 - XML Encoding.

Specifying the text file to use is very straight forward. Use the following constructor and replace the variable `dataFile` with the path (relative or full path) of the data file. If you are connecting to the server, relative paths should be in respect to the EspressReport root directory. Otherwise, the path will be relative to the current working directory.

```
QbReport(java.lang.Object parent, int reportType, java.lang.String dataFile,
 ColInfo[] mapping, java.lang.String template)
```

For a working example using data from a text file, see Appendix 2.C.2.2 - Creating the Report.

2.B.3. Data from an XML Data Source

In addition to the above, EspressReport allows you to retrieve data and query XML files. XML data can be in virtually any format, but you need to specify a DTD file or an XML schema along with the XML data. The following code demonstrates how to set up an XML query:

```
// Set up the XML Data Source
String xmlfilename = "Inventory.xml";
String dtdfilename = "Inventory.dtd";
String xmlcondition = "/Inventory/Category/Product/ProductID < 45";

XMLFieldInfo[] fields = new XMLFieldInfo[5];
fields[0] = new XMLFieldInfo(new String[] {"Inventory", "Category",
 "Product"}, "ProductID");
fields[0].setAttributeDataType(DTDDataType.INT);

fields[1] = new XMLFieldInfo(new String[] {"Inventory", "Category",
 "Product", "ProductName"});

fields[2] = new XMLFieldInfo(new String[] {"Inventory", "Category",
 "Product", "UnitPrice"});
fields[2].setElementDataType(DTDDataType.DOUBLE);

fields[3] = new XMLFieldInfo(new String[] {"Inventory", "Category",
 "Product", "UnitsInStock"});
fields[3].setElementDataType(DTDDataType.INT);

fields[4] = new XMLFieldInfo(new String[] {"Inventory", "Category",
 "Product", "ShipDate"});
fields[4].setElementDataType(DTDDataType.DATE);
fields[4].setDateFormat(XMLDataTypeUtil.YYYY_MM_DD);

XMLFileQueryInfo xmlInfo = new XMLFileQueryInfo(xmlfilename, fields,
 xmlcondition, fields, dtdfilename, false, null);
```

The `XMLFieldInfo` instance is created using one of two constructors. The first constructor, used to select xml fields, contains one parameter. For this constructor, you need to pass in a String array that specifies each xml tag in the hierarchy leading to the target field. In the above example, fields[1-4] are created using the first constructor. The second constructor, used to select xml attributes, contains two parameters. In addition to the String array parameter, the second constructor also requires another string for the attribute name. In the above example, field[0] is created using this constructor because `ProductID` is an attribute of `Product`.

You may have also noticed that for any non-String field, you must explicitly set the data type. Once you have created the `XMLFileQueryInfo` instance, you can use the following constructor to create the `QbReport`.

```
public QbReport(Object parent, int reportType, XMLFileQueryInfo xmlInfo,
 ColInfo[] colMap, String templateFile, boolean sideBySideLayout);
```

You can also pass in an XML stream instead of an XML file, when using XML data as a data source. To pass in an XML stream, you would pass in the byte array containing the XML data instead of the XML data file name.

In the above example, you can pass in a XML stream via a byte array (for example, a byte array called `xmlByteArray`) in the `XMLFileQueryInfo` constructor:

```
XMLFileQueryInfo xmlInfo = new XMLFileQueryInfo(xmlByteArray, fields,
 xmlCondition, fields);
```

For a working example using data passed in from an xml file source, see Appendix 2.C.5.2 - Creating the Report.

2.B.4. Data passed in an Array in Memory

The API allows input data to be passed directly in memory, as an array. This is made possible by the interface IResultSet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/reportdesigner/util/IResultSet.html] (defined in quadbase.reportdesigner.util package [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/reportdesigner/util/package-summary.html]). This interface is used to read data in tabular form and is quite similar to the `java.sql.ResultSet` interface used for JDBC result sets (but is much simpler). Users can provide their own implementation of `IResultSet`, or use one provided by EspressReport. The simplest implementation is the class `DbData` (Other classes that provide an `IResultSet` implementation are QueryResultSet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/reportdesigner/util/QueryResultSet.html] and StreamResultSet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/reportdesigner/util/StreamResultSet.html]). If you can fit all the data you need for the report in memory, you can simply pass the array as an argument in `DbData` with one line of code. There are three constructors for `DbData`:

```
      DbData(java.lang.String s)
            Construct DbData by parsing the data value argument from an HTML
 page

      DbData(java.lang.String[] fieldName, java.lang.Object[][] records)
            Construct a new DbData class

      DbData(java.lang.String[] dataType, java.lang.String[] fieldName,
 java.lang.String[][] records)
            Construct a new DbData class
```

Note that when passing in data, using this approach, for a Summary Break or Cross Tab report, make sure that the data is applicable and is already grouped and sorted to improve performance and generate a report successfully.

We will use the following constructor in the example here.

```
public DbData(String dataType[], String fieldName[], String records[][])
```

Here, the first argument presents the data types (the first line in the data file) and the second argument presents the field names (the second line). The third argument, `records[][]`, provides an array of records, `records[i]` being the ith record. The following shows how it works:

```
String dataType[] = {"varchar", "decimal"};
String fieldName[] = {"People", "Sales"};
String records[][] = {{"Peter", "93"}, {"Peter", "124"},
                      {"John", "110"}, {"John", "130"},
                      {"Mary", "103"}, {"Mary", "129"}};

DbData data = new DbData(dataType, fieldName, records);
```

To create the report, use the following `QbReport` constructor:

```
public QbReport(Object parent, int reportType, IResultSet data,
               ColInfo[] colMap, String templateFile);
```

You can pass sorted data to EspressReport to improve performance considerably. If the data is already sorted, you can set a flag to `true` in the constructor to avoid EspressReport sorting the data again. The constructor is given below:

```
public QbReport(Object parent, int reportType, IResultSet data,
                ColInfo[] colMap, String templateFile, boolean
 sideBySideLayout, boolean isDataSorted);
```

This is important especially when you are generating a summary break report with multiple break levels. If data in memory is already sorted (e.g. sorted by a database engine), you can take advantage of the fact and use this feature to prevent the report engine from unnecessarily building extra internal data structures and performing sorting, which consumes a large amount of memory and processor resources. Hence, performance can be greatly improved.

For a working example using data passed in an array in memory, see Appendix 2.C.1.2 - Creating the Report.

2.B.5. Data passed in a Custom Implementation

For maximum flexibility, you can retrieve and prepare the dataset in any way you want and pass it to the report engine. To pass in your class file as the data source, your class file must implement the IDataSource [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/reportdesigner/util/IDataSource.html] interface. Given below is a simple example that implements `IDataSource`:

```
public class CustomClassData extends Applet implements IDataSource {

      // Setting DbData for passing data as arguments
      String dataType[] = {"string", "String", "double"};
      String fieldName[] = {"Destination", "Time", "Price"};
      String records[][] = {{"Mayfair", "13:43", "3.50"},
            {"Bond Street", "13:37", "3.75"},
            {"RickmansWorth", "13:12", "5.25"},
            {"Picadilly", "13:24", "3.00"}};
      DbData data = new DbData(dataType, fieldName, records);

      public IResultSet getResultSet()
      {
            return data; }

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/CustomClassDataER.zip]

The example above creates data (DbData instance) and stores it in memory. When the getResultSet() method is called, it returns the DbData object which implements IResultSet. Keep in mind that it is not necessary to create your data in this manner. As long as you are able to return an object that implements IResultSet, you can obtain the data from any data source. Use the following constructor to create your report:

```
QbReport(java.lang.Object parent, int reportType, int fileType,
 java.lang.String filename, ColInfo[] mapping, java.lang.String template)
```

For custom class files, set the fileType to QbReport.CLASSFILE and the filename to the name of the classfile.

Please note that if you are passing in your own class file as the data source and you are using the EspressManager, the class file must be accessible from the CLASSPATH of the EspressManager.

You can also pass in a parameterized class file as the data source for the report. The parameter is obtained at run-time from the user and the data is then fetched and used to generate the report. Here is an example for a parameterized class file, this is the same file used in Section 2.3.5.7.11.5 - Open Report Designer with a Specific Class File Data Source .

```
public class ParamClassFile implements IParameterizedDataSource
{
 String url = "jdbc:hsqldb:help/examples/DataSources/database/woodview";
 String driver = "org.hsqldb.jdbcDriver";
 String username = "sa";
 String password = "";

 // Specify what the parameter properties are.
 public IQueryInParam[] getParameters()
 {
  SimpleQueryInParam[] params = new SimpleQueryInParam[2];
  //  SimpleQueryInParam(name, promptString, mapToColumn, TableName,
 ColumnName, dataType, defaultValue, actualValue)
  params[0] = new SimpleQueryInParam("price", "Max price:", false, null,
 null, Types.DOUBLE, new Double(500.00), null);
  params[1] = new MySimpleQueryInParam("popular", "Popular Items Only:",
 false, null, null, Types.VARCHAR, new String("NO"), null);
  return params;
 }

 private class MySimpleQueryInParam extends SimpleQueryInParam implements
 IQueryParamValuesProvider
 {
  public MySimpleQueryInParam(String paramName, String promptName, boolean
 mapToColumn,
    String tableName, String columnName, int sqlType, Object defaultValue,
 Object value)  {
   super(paramName, promptName, mapToColumn, tableName, columnName, sqlType,
 defaultValue, value);
  }
  public Vector getSelectionChoices() {
   Vector choices = new Vector();
   choices.add(new String("Yes"));
   choices.add(new String("No"));
   return choices;
  }
 }
 // Specify what data is to be returned
 public IResultSet getResultSet(IQueryInParam[] params)
 {
  double price = 500.00;
  int units = 999;

  if ((params != null) && (params.length >= 1))
  {
   Object obj = params[0].getValue();
   if ((obj != null) && (obj instanceof Double)) price =
 ((Double)obj).doubleValue();
   obj = params[1].getValue();
   if((obj != null) && (obj instanceof String) &&
 ((String)obj).equalsIgnoreCase("yes")) units = 15;
  }

  try{
   Class.forName(driver);
   Connection conn = DriverManager.getConnection(url, username, password);
   String query = "SELECT ProductName, Description, UnitPrice, UnitsInStock
 FROM Products WHERE UnitPrice < "
    + price + " AND UnitsInStock < " + units;
   Statement stmt = conn.createStatement();

   ResultSet rs = stmt.executeQuery(query);
   QueryResultSet qry = new QueryResultSet(rs);

   return qry;
  } catch(Exception e) {
   e.printStackTrace();
  }
  return null;
 }

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ParamClassFile.zip]

The parameterized class file must implement IParameterizedDataSource [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/reportdesigner/util/IParameterizedDataSource.html]. The above example obtains data from the WoodView HSQL database, so you will need to add database HSQL JDBC driver (`hsqldb.jar`) to your classpath. The driver is located in the `<EspressReportInstall>/lib` directory.

The query contains two parameters. The first parameter allows the user to set the maximum price for the query results. The parameter is of type `double` and the default is set to `500.00`. The second parameter is a custom parameter and its purpose is to give the user a `yes` or `no` option to determine whether or not to show popular items only. To construct a custom parameter, create a class that extends the SimpleQueryInParam [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/reportdesigner/util/SimpleQueryInParam.html] class, implements the IQueryParamValuesProvider [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/reportdesigner/util/IQueryParamValuesProvider.html] interface and overwrites the `getSelectionChoices()` method. If the user selects `yes`, then in `getResultSet()` only items with less than `15` `UnitsInStock` will be returned.

You can use the same constructor as before to create a report using this parameterized class. When using parameterized class file as the data source, keep in mind that you can not map it to a column directly (i.e. set the table name and column name in the `SimpleQueryInParam` constructor). In order to map a parameter to a column, you will have to use a custom parameter similar to the one shown above. However, the `getSelectionChoices()` method will look as follows:

```
public Vector getSelectionChoices() {
 System.out.println("getSelectionChoices called");
 try {
  Class.forName("org.hsqldb.jdbcDriver");

  String url = "jdbc:hsqldb:help/examples/DataSources/database/woodview";
  Connection conn = DriverManager.getConnection(url, "sa", "");
  Statement stmt = conn.createStatement();
  String query = "SELECT DISTINCT " + getColumnName() + " FROM " +
 getTableName();
  ResultSet rs = stmt.executeQuery(query);
  Vector v = new Vector();

  while (rs.next()) {
   switch (getSqlType()) {
   case Types.INTEGER:
    v.add(new Integer(rs.getInt(1)));
    break;

   case Types.VARCHAR:
    v.add(rs.getString(1));
    break;
   }
  }

  stmt.close();
  conn.close();

  return v;
 } catch (Exception ex) {
  ex.printStackTrace();
 }
 return null;
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ClassDataMapToColumn.zip]

In addition to mapping the parameter to a column, this example also uses a multi-value parameter. Although the process is similar to using a single value parameter, there are some places where the code deviates from the norm. Please see the comments in the above source code for details. Also, more information on using parameters from the API can be found in Appendix 2.C.7 - Parameterized Report.

For a working example using data passed in from a custom class file, see Appendix 2.C.3.2 - Creating the Report.

2.B.6. Data from Enterprise Java Beans (EJBs)

Data can be passed from an EJB data source to the report by allowing users to query data directly from an entity bean. To add an EJB as a data source, the EJB must first be deployed in the application server, and the client JAR file containing the appropriate stub classes must be added to your classpath (or the `-classpath` argument of the EspressManager batch file when using the API in conjunction with EspressManager).

To construct a report using an EJB as a data source, first you must construct a `EJBInfo` object with information for connecting to the EJB. The constructor is shown below:

```
public EJBInfo(java.lang.String jndiName,
               java.lang.String homeName,
               java.lang.String remoteName,
               java.lang.String selectedMethodName,
                    java.lang.Object[] selectedMethodParamVal)
```

For example, the following lines create an `EJBInfo` instance that connects to the `ProductEJB` data source.

```
Object[] vals = new Object[1];
vals[0] = new String("Printer");

EJBInfo ejbInfo = new EJBInfo("ProductEJB", "ejb.ProductHome",
                              "ejb.Product", "findByCategory", vals);
```

Finally, use the following constructor to create the `QbReport` object:

```
public QbReport(Object parent, int reportType, EJBInfo ejbInfo, ColInfo[]
 mapping, String template);
```

2.B.7. Data from a SOAP Data Source

EspressReport allows to retrieving data from SOAP services. To use a SOAP data source, you need to provide a location of WSDL file, which contains all the necessary information. The location can be either absolute path on the server or path relative to your EspressReport installation directory or URL.

Constructor for SOAP data source looks like this:

```
SOAPQueryFileInfo(String wsdlURI, QName serviceName, String portName, String
 operationName, XMLFieldInfo[] xmlFieldInfo, SOAPParam[] parameters)
```

The `wsdlURI` is URL or absolute/relative path to the WSDL file. The `serviceName` of type `QName` is name of SOAP service. The following constructor is used:

```
QName(String namespaceURI, String localPart),
```

where the `namespaceURI` is URI namespace (a part of the `serviceName` string in curly brackets) and the `localPart` is a service name (the rest of the string). For example, in the following servicename `{http://www.webservicex.net}WeatherForecast` . The `http://www.webservicex.net` is `namespaceURI` and the `WeatherForecast` is a service name.

The `portName` and `operationName` in the `SOAPQueryFileInfo` constructor are names of a port and operation. The `XMLFieldInfo` array is the same as for XML data sources (see Appendix 2.B.3 - Data from an XML Data Source). Finally, the `SOAPParam` field is dealing with parameters and uses the following constructor:

```
SOAPParam(String paramName, int sqlType, String paramPrompt, Object
 defaultValue, Object value, boolean alwaysUseDefault),
```

where the `paramName` is a parameter name, the `sqlType` represents parameter data type, which is a constant from `java.sql.Types`. The `paramPrompt` is the parameter prompt string, the `defaultValue` is the parameter default value, the `value` is the parameter value and the `alwaysUseDefault` of boolean type says whether the default value will be always used or not. This means that this parameter value will be fixed and users will not be prompted for it.

To create the `QbReport/QbChart` object use the constructors below:

```
     QbReport(Object parent, int reportType, SOAPQueryFileInfo soapInfo,
ColInfo[] mapping, String template, Properties props)

     QbChart(Applet applet, int dimension, int chartType, SOAPQueryFileInfo
soapInfo, boolean doTransposeData, int[] transposeCol, IColumnMap cmap,
String template)
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SOAP.zip]

2.B.8. Data from Multiple Data Sources

EspressReport also provides the functionality to merge multiple data sources together. You can create a `DataSheet` object from any combination of data sources, for example, data file, database or `IResultSet`. You can use a `QbReport` constructor to create a report object from an array of `DataSheet` objects.

The following example program fragment demonstrates how to combine the data from the examples in the above sections:

```
// Declaration of DataSheet object to be used to merge data
DataSheet dataSheet[] = new DataSheet[3];

// Declaration For Database (Data from Database section)
DBInfo dbinfo = new DBInfo("jdbc:hsqldb:help/examples/DataSources/database/
woodview",
"org.hsqldb.jdbcDriver", "sa","","select * from Orders");

//Declaration For Data Passed in from Memory (Data Passed in from Memory
 section)
String dataType[] = {"varchar", "decimal"};
String fieldName[] = {"People", "Sales"};
String records[][] = {{"Peter", "93"}, {"Peter", "124"},
                      {"John", "110"}, {"John", "130"},
                      {"Mary", "103"}, {"Mary", "129"}};
DbData data = new DbData(dataType, fieldName, records);

// Create DataSheet from data file (Data from a Text File section)
// DataSheet(Applet applet, String dataFile)
dataSheet[0] = new DataSheet(this, "../data/1_A_7_TextData.dat");

// Create DataSheet from database (Data from a Database section)
// DataSheet(Applet applet, IDatabaseInfo dbInfo)
dataSheet[1] = new DataSheet(this, dbinfo);

// Create DataSheet from IResultSet  (Data Passed in from Memory section)
// DataSheet(Applet applet, IResultSet data)
dataSheet[2] = new DataSheet(this, data);
```

Then, use the following `QbReport` constructor to create the report:

```
QbReport(java.lang.Object parent, int reportType, DataSheet[] dataSheet,
 ColInfo[] mapping, java.lang.String template)
```

Note that after you have created a `DataSheet` object, you can modify it (add, delete, or update row values) by using DataSheet API. Data is not refreshable if merging data from `IResultSet`.

2.C. Creating the Report

To create a new report, you must specify the report type, the input data source information, and a mapping of the data columns to the respective columns of the report. In this appendix, we look at the various report types and the methods used to map the columns. There are also a number of fully functional examples in this appendix.

Please note that unless otherwise noted, all examples use the Woodview HSQL database, which is located in the `<EspressReportInstall>/help/examples/DataSources/database` directory. In order to run the examples, you will need to add database HSQL JDBC driver (`hsqldb.jar`) to your classpath. The driver is located in the `<EspressReportInstall>/lib` directory.

There are five basic report types: *Simple Columnar, Summary Break, Cross Tab, Master & Details,* and *Mailing Labels* reports. Every report has a column mapping defined as well as properties specific to the type of the report.

The column mapping is done in the API using the ColInfo [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/reportdesigner/util/ColInfo.html] class (located in the quadbase.reportdesigner.util [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/reportdesigner/util/package-summary.html] package). You define the column mapping by declaring a `ColInfo` array object. Each element in the array represents the column's respective column. You can also assign specific properties for each element of the `ColInfo` array object, depending on the type of the report desired, and create the report.

The `QbReport` class contains numerous constructors. However, most constructors can be broken down into three sets of parameters. The first set is only the parent object. The parent object must be set if the report is to be used in an applet context, it can be null otherwise. The second set of parameters are always required and they determine the type of report you are constructing, the data source, and the mapping. There is a great deal of information in the two Appendices regarding this group of parameters. Finally, the third set of parameters are options and properties. For example, most constructors have the option to specify a template, although it is not necessary. Some options are specific to certain report types, for example the boolean parameter `sideBySideLayout` is only relevant if you are creating a Master & Details report. Some constructors contain a `Properties` parameter that allows you to set multiple properties and group them into one properties object. For more information on the various options and properties, please see the APIDocs [https://data.quadbase.com/Docs71/er/help/apidocs/index.html].

Caution

As you may know, creating objects in java is a resource intensive operation. It is always recommended that you do not create too many objects. One way to conserve resource is to reuse `QbReport` objects whenever possible. For example, if a lot of your users request for a Simple Columnar Report in your website, instead of creating a new `QbReport` object when each such request is received, you can have one (or a limited number of) such `QbReport` object(s) created and reuse the object(s) by simply modifying the data and attributes of the report for each particular request.

2.C.1. Simple Columnar

The *Simple Columnar* report is the most basic of all the types supported by EspressReport. It presents columnar data in a single table without any groupings, or breaks.

Here is an example:

Product Type	Product Name	Price
12	Chair	$20
12	Table	$30
14	Cabinet	$20
14	Table	$50

2.C.1.1. Column Mapping

With *Simple Columnar* reports, all you need to do is define the `ColInfo` array object with respect to the data source. For instance, given the dataset below:

```
String dataType[] = {"String", "String", "String"};
String fieldName[] = {"Product Type", "Product Name", "Price"};
String records[][] = {{"12", "Chair", "$20"},
                      {"12", "Table", "$30"},
                      {"14", "Cabinet", "$20"},
                      {"14", "Table", "$50"}};
```

The column mapping is set so that the columns of the data source are mapped to the report. Thus Column Zero of the report corresponds to Column Zero of the data source, Column One of the report to Column One of the data source and so on and so forth. Thus, the following report is created:

Product Type	Product Name	Product Price
12	Chair	$20
12	Table	$30
14	Cabinet	$20
14	Table	$50

Report after mapping and setting the Table Header Names

To generate the aforementioned report, the following column mapping has to be set using the API:

```
ColInfo[] colInfo = new ColInfo[3];
colInfo[0] = new ColInfo(0);
colInfo[0].setName("Product Type");
colInfo[1] = new ColInfo(1);
colInfo[1].setName("Product Name");
colInfo[2] = new ColInfo(2);
colInfo[2].setName("Product Price");
```

In the above code, the names for the columns are also changed to `Product Type`, `Product Name` and `Product Price` respectively rather than keeping the Header names defined in the original data.

The column mapping need not be in order or follow the same order as the data. You can also selectively choose the columns you desire. For instance, given a data source that has multiple columns, you can write the following code:

```
ColInfo[] colInfo = new ColInfo[3];
colInfo[0] = new ColInfo(2);
colInfo[0].setName("Product Name");
colInfo[1] = new ColInfo(7);
colInfo[1].setName("Units Sold");
colInfo[2] = new ColInfo(4);
colInfo[2].setName("Unit Price");
```

Here, the Column `0` of the report is mapped to column `2` of the data source, Column `1` of the report is mapped to column `7` of the data source and Column `2` of the report is mapped to column `4` of the data source. In addition to setting the column mappings, the above example also sets the Table Header for each column.

2.C.1.2. Creating the Report

Constructing a *Simple Columnar* report is relatively straight forward. We have already discussed how to set the `ColInfo` array and how to obtain the data in previous sections. The following code demonstrates how to create a *Simple Columnar* report.

```
// Data passed in an array in memory
DbData data = new DbData(dataType, fieldName, records);
```

```
//  Set Column Mapping
ColInfo colInfo[] = new ColInfo[3];
colInfo[0] = new ColInfo(0);
colInfo[0].setName("Product Type");
colInfo[1] = new ColInfo(1);
colInfo[1].setName("Product Name");
colInfo[2] = new ColInfo(2);
colInfo[2].setName("Product Price");

//  Create Report
QbReport report = new QbReport
    (parent, // Parent
    QbReport.COLUMNAR, // Type of Report
    data, // Data
    colInfo, // Column Mapping - use the column mapping in the template
    "SimpleColumnar.rpt"); // Template Name
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SimpleColumnar.zip]

Exported Unformatted Results [https://data.quadbase.com/Docs71/help/manual/code/export/SimpleColumnarNF.html]

Exported Formatted Results [https://data.quadbase.com/Docs71/help/manual/code/export/SimpleColumnar.html]

The Exported Unformatted Results link shows how the report appears when it was first created without any formatting or templates. The Exported Formatted Results presents the outcome after applying the template.

Even though this example creates a report from scratch, we still recommend that you apply a template. If you look at the source code, notice the large commented section after the `QbReport` constructor. This section modifies the visual appearance of the report in exactly the same way as applying the template. As you can imagine, writing the code takes much more effort since there is no visual aid to help you gauge position and dimensions.

When you apply a template, you have the option of using the column mapping stored in the template. In the above example, you could replace the `colInfo` in the constructor with null and it would still generate the same report. This is because the column mapping in the template is exactly the same. If you set the `ColInfo` and apply a template, the constructed report will use the `ColInfo` you supplied and ignore the column mapping in the template.

2.C.2. Summary Break

For a *Summary Break* report, you need to define a Row Break. The Row Break defines when to break the report and insert column summaries. The report essentially gets “broken” everytime the data in the “break” column changes. An aggregation must be set for any columns that are not specified as Row Breaks, even if the columns are non-numeric (set the aggregation to `NONE`). This is so that column summaries can be generated based on the aggregation. For instance, given the dataset below:

Order #	Product	Quantity
12	Chair	2
12	Table	3
14	Cabinet	2
14	Table	5

Original Data

If Column `0` (`Order #`) is assigned as `Row Breaks`, the `Aggregation` for Column `1` (`Product`) is set to `NONE`, and the `Aggregation` for Column `2` (`Quantity`) is set to `SUM`, we get the following report.

Order #	Product	Quantity
12	Chair	2
	Table	3

Order #	Product	Quantity
		5
14	Cabinet	2
	Table	5
		7

Report after applying Row Break and Aggregation

2.C.2.1. Column Mapping

Setting the column mapping for *Summary Break* reports requires that two more properties to be set than the *Simple Columnar* report. First, the row break columns must be specified for row group columns. Second, columns that are not row break columns must specify the aggregation type. For example, to generate the report in the previous section, you need the set the following lines of code:

```
ColInfo[] colInfo = new ColInfo[3];
colInfo[0] = new ColInfo(0);
colInfo[0].setRowBreak(true);
colInfo[1] = new ColInfo(1);
colInfo[1].setAggregation(false, ColInfo.NONE);
colInfo[2] = new ColInfo(2);
colInfo[2].setAggregation(false, ColInfo.SUM);
```

Again, the first step is to map the column in the report with the column from the data source. Then, Column Zero is set to be a row break column. Since Columns One and Two are not set as row break columns, you must set an aggregation for them even if it is `NONE`. There are two parameters in the `setAggregation` method, the first parameter determines if the report uses column aggregation. If column aggregation is used, the individual rows of data will be hidden and only the aggregation will be displayed. If you set the first parameter to `true`, the other non row break column must also be set to use column aggregation. The second parameter determines the type of aggregation. Here is a list of all available aggregations in the `ColInfo` class:

AVG	MAX	STDDEV
COUNT	MEDIAN	SUM
COUNT-DISTINCT	MIN	SUMSQUARE
FIRST	NONE	VARIANCE
LAST		

2.C.2.2. Creating the Report

The following example shows how to create a *Summary Break* report:

```
ColInfo[] colInfo = new ColInfo[5];
colInfo[0] = new ColInfo(1);
colInfo[0].setRowBreak(true);
colInfo[1] = new ColInfo(9);
colInfo[1].setRowBreak(true);
colInfo[2] = new ColInfo(4);
colInfo[2].setAggregation(true, ColInfo.SUM);
colInfo[3] = new ColInfo(5);
colInfo[3].setAggregation(true, ColInfo.SUM);
colInfo[4] = new ColInfo(6);
colInfo[4].setAggregation(true, ColInfo.SUM);

QbReport report = new QbReport
```

```
  (parent, // parent
  QbReport.SUMMARY, // Sumamry Break  Report
  "sample.dat", // filename
  colInfo, // column information
  null);
try {
 //   Apply template including scripts and formula
 report.applyTemplate("SummaryBreak.rpt", true);
} catch (Exception e) {
 e.printStackTrace();
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SummaryBreak.zip]

Exported Unformatted Results [https://data.quadbase.com/Docs71/help/manual/code/export/SummaryBreakN-F.html]

Exported Formatted Results [https://data.quadbase.com/Docs71/help/manual/code/export/SummaryBreak.html]

The Exported Unformatted Results link shows how the report appears when it was first created without any formatting or templates. The Exported Formatted Results presents the outcome after applying the template.

First thing you may notice is that the column mapping is different from the previous examples. This is often the case when using a text file as the data source. In this example, we are mapping column `0` from the report with column `1` in the text file, column `1` from the report with column `9` in the text file, and so on.

Also, notice that the example uses a template to format the appearance of the report eliminating the need to write excessive amounts of code. In this example, we cannot include the template in the constructor because by default, adding the template to the constructor will not import the formula and scripts. The way to import both the formula and the scripts is by calling the method `applyTemplate` after constructing the `QbReport` object. The two arguments that you will need to pass in are the file path of the template and a boolean specifying whether you want to import formula and scripts.

An alternate way to create a summary break report is by specifying that your data source is already sorted. That way, EspressReport will build a summary break report in the same manner as for database data. This feature is available for all data sources including text, XML, and class data. To do this, simply call the `QbReport` constructor with the boolean parameter `isDataSorted` set to `true`:

```
new QbReport(parent, reportType, data, mapping, template, sideBySideLayout,
 isDataSorted);
```

2.C.3. CrossTab

A *Crosstab* report is a report format that shows and summarizes columnar data in a matrix-like form. *Crosstab* reports often resemble spreadsheets. Both rows and columns are summarized, allowing multi-dimensional data to be displayed in a 2 dimensional format.

In addition to defining `Row Breaks`, you will also need to specify a column that serves as the `Column Break` as well as a column that serves as the `Column Break Value`. The `Column Break` column gets included in the Table Header with a separate column for each unique entry in the selected column while the `Column Break Value` column represents the field that is summarized in the report. You will also need to specify the type of `Aggregation` for the `Column Break Value` column. For instance, given the dataset below:

Region	Product	Total Sales
East	Chair	14500
Midwest	Chair	13250
South	Chair	15252
East	Table	10550
Midwest	Table	9150

Region	Product	Total Sales
South	Table	11250

Original Data

If Column 0 (Region) is assigned as Row Break, Column 1 (Product) is assigned as the Column Break, and Column 2 (Sales) assigned as the Column Break Value with a summation aggregation, we get the following report:

Region	Chair	Table	Total Sales
East	14500	10550	25050
Midwest	13250	9150	22400
South	15252	11250	26502
	43002	30950	73952

Report after applying Column Break and Column Break Value with Aggregation

This example demonstrates several qualities frequently exhibited by *CrossTab* reports. Notice that the distinct Product names are transformed to become column headers. This means that the number of columns in the report will vary based on the data source. Also, the numeric values represent Total Sales for a particular combination of Product and Region. The cells to the right and bottom of the data are aggregations cells. The Total Sales at the end of each row sums up the Total Sales for each region, while the values at the bottom of each column sums up the Total Sales of each product.

2.C.3.1. Column Mapping

To generate the aforementioned report, the following column mapping has to be set using the API:

```
ColInfo[] colInfo = new ColInfo[3];
colInfo[0] = new ColInfo(0);
// use this column as the row break. The number of
// new groups would be equal to the number of unique
// fields present.
colInfo[0].setRowBreak(true);

colInfo[1]=new ColInfo(1);
// use this column as the column break. The number of
// new columns would be equal to the number of unique
// column fields present.
colInfo[1].setColumnBreak(true);

colInfo[2] = new ColInfo(2);
// use this column as the value fields for the
// newly created chair and table columns
colInfo[2].setColumnBreakValue(true);
// aggregate by adding up the individual fields to
// get the summary field.
colInfo[2].setAggregation(ColInfo.SUM);
```

In the above code, the names for the columns are not set as the original names of the columns from the data source sufficed.

Note that in the example given above, only a single Column Break and a single Column Break Value was specified. However, just like Row Breaks, multiple Column Breaks and Column Break Values can be specified.

For example:

```
ColInfo[] colInfo = new ColInfo[7];
colInfo[0] = new ColInfo(0);
colInfo[0].setRowBreak(true);

colInfo[1]=new ColInfo(1);
colInfo[1].setRowBreak(true);

colInfo[2]=new ColInfo(2);
colInfo.setColumnBreak(true);

colInfo[3]=new ColInfo(3);
colInfo[3].setColumnBreak(true);

colInfo[4]=new ColInfo(4);
colInfo[4].setColumnBreakValue(true);
colInfo[4].setAggregation(ColInfo.SUM);

colInfo[5]=new ColInfo(5);
colInfo[5].setColumnBreakValue(true);
colInfo[5].setAggregation(ColInfo.SUM);

colInfo[6]=new ColInfo(6);
colInfo[6].setColumnBreakValue(true);
colInfo[6].setAggregation(ColInfo.SUM);
```

In the above code, Columns `2` and `3` of the data source are set to be `Column Break` columns while Columns `4, 5` and `6` are set to be `Column Break Value` columns.

2.C.3.2. Creating the Report

The following example shows how to create a *CrossTab* report.

```
ColInfo[] colInfo = new ColInfo[4];
colInfo[0] = new ColInfo(0);
colInfo[0].setRowBreak(true);

colInfo[1] = new ColInfo(1);
colInfo[1].setColumnBreak(true);

colInfo[2] = new ColInfo(2);
colInfo[2].setColumnBreakValue(true);
colInfo[2].setAggregation(ColInfo.AVG);

QbReport report = new QbReport
  (parent, // Parent
  QbReport.CROSSTAB, // Type of Report
  QbReport.CLASSFILE, // Data
  "ClassFile",
  colInfo, // Column Mapping
  "CrossTab"); // Template Name
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/CrossTab.zip]

Exported Unformatted Results [https://data.quadbase.com/Docs71/help/manual/code/export/CrossTabNF.html]

Exported Formatted Results [https://data.quadbase.com/Docs71/help/manual/code/export/CrossTab.html]

The Exported Unformatted Results link shows how the report appears when it was first created without any formatting or templates. The Exported Formatted Results presents the outcome after applying the template.

The example uses a class file as the data source. The class file generates the temperatures everytime you run the program so the results will vary. For more information on class file data sources, see Appendix 2.B.5 - Data passed in a Custom Implementation.

2.C.3.3. Fixed-field Crosstab Report

The fixed-field crosstab option improves some of the limitations of the free-form crosstab implementation, by making it easier for users to design crosstab reports that can expand and contract with changing data.

The key difference between a fixed field and a free-form crosstab is that in the design view, you are not able to position or control the individual columns for the crosstab report. Instead you can set formats for groups of elements (row break, headers, footers and formulas). The actual report is only constructed during running/preview. Since the crosstab table is essentially constructed from scratch every time the report is run it is easier to create a smoothly contracting and expanding crosstab table.

2.C.3.3.1. Creating the Report

The following example shows how to create a multi-dimensional (extra row/column breaks) fixed-field crosstab report with 3 column breaks.

```
ColInfo colInfo[] = new ColInfo[4];

colInfo[0] = new ColInfo(0);
colInfo[1] = new ColInfo(1);
colInfo[2] = new ColInfo(2);
colInfo[3] = new ColInfo(3);

colInfo[0].setRowBreak(true);
colInfo[1].setColumnBreak(true);
colInfo[2].setColumnBreak(true);
colInfo[3].setColumnBreakValue(true);
colInfo[3].setAggregation(ColInfo.SUM);

Properties props = new Properties();
props.put("crossTabFreeForm", "false"); // Using fixed-field form
props.put("crossTabSummaryPositionL", "false"); // Draw summary column to
 the right of the details matrix
props.put("crossTabFormulaOnHeader", "true"); // Draw formula column in the
 header section
props.put("crossTabColBkValAlignH", "false"); // Align Column Break Value
 vertically

QbReport report = new QbReport
  (parent, // Parent
  QbReport.CROSSTAB, // Report Type
  QbReport.DATAFILE, // Data Type
  "sample.dat", // Filename
  colInfo, // Column Mapping
  "field.rpt", // Template
  props); // Properties
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/FixedFieldCrosstab.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/Fixed_field_Crosstab.pdf]

In this example, we are mapping column `0` from the report with column `0` in the text file, column `1` from the report with column `1` in the text file, and so on. In the above code, columns `1, 2` and `3` of the data source are set to be `Column Break` columns and column `4` is set to be `Column Break Value` column with an sum aggregation. There are different crosstab options, because you cannot individually format or position crosstab options in the fixed-field layout. In this example the summary column is set to be drawn to the right of the report data, the formula column is set to be drawn in the header section and the column break value is set to be aligned vertically.

For more information about crosstab, see Section 1.4.3 - Crosstab Report.

2.C.4. Master & Details

A *Master & Details* report is a set of tabular data that is grouped according to a `Master` field. This report type is most commonly used when some fields in your data table are related to many other fields. A good example of this is an invoice. For each order number, there will be customer information, and several items with pricing information. A *Master & Details* report would be used to group the information according to order number.

In a *Master & Details* report, the data is grouped based on a “Primary” and/or “Master” column(s). Here, you **must** define a `Primary Key` column. The `Primary Key` column (order number in the invoice example) determines the grouping of the report. A new group will be created every time there is a value in the selected column (i.e. the `Primary Key` column). Please note that only one column can serve as the `Primary Key`. A `Master` field or several `Master` fields can also be selected in addition. Any column selected as the `Master` field (customer information) will result in the placing of the column value in the Group Header instead of the Data Section of the report. By definition, the `Primary Key` value must be unique for each grouping of the `Master` field column(s). For instance, given the dataset below:

Order #	Customer Name	Product	Unit Price	Quantity
12	Paul Campbell	Chair	$24.95	4
12	Paul Campbell	Table	$127.50	1
14	Sally Hayes	Cabinet	$227.25	2
14	Sally Hayes	Chair	$24.95	2
14	Sally Hayes	Table	$127.50	1

Original Data

If Column `0` (`Order #`) is assigned as the `Primary Key` and Column `1` (`Customer Name`) is assigned as the `Master` field, we get the following report:

Order #	**12**	
Customer Name	**Paul Campbell**	
Product	**Unit Price**	**Quantity**
Chair	$24.95	4
Table	$127.50	1
Order #	**14**	
Customer Name	**Sally Hayes**	
Product	**Unit Price**	**Quantity**
Cabinet	$227.25	2
Chair	$24.95	2
Table	$127.50	1

Report after applying Primary Key and Master field

2.C.4.1. Column Mapping

To generate the aforementioned report, the following column mapping has to be set using the API:

```
ColInfo[] colInfo = new ColInfo[5];
colInfo[0] = new ColInfo(0);
```

```
colInfo[0].setPrimaryKey(true);
colInfo[1]=new ColInfo(1);
colInfo[1].setMaster(true);
colInfo[2] = new ColInfo(2);
colInfo[3] = new ColInfo(3);
colInfo[4] = new ColInfo(4);
```

In the above code, the names for the columns are not set as the original names of the columns from the data source sufficed.

You can select any number of columns to be in the master section. The only restriction is that the primary key is automatically placed in the master section. If this is undesired, you can set that column to be invisible.

2.C.4.2. Creating the Report

The following example shows how to create a *Master & Details* report.

```
String url = "jdbc:hsqldb:database/woodview";
String driver = "org.hsqldb.jdbcDriver";
String query = "...";

DBInfo dbInfo = new DBInfo(url, driver, "sa", "", query);

ColInfo[] colInfo = new ColInfo[16];

for(int i = 0; i < colInfo.length; i++){
 colInfo[i] = new ColInfo(i);
 if(i < 8)
  colInfo[i].setMaster(true);
}
colInfo[0].setPrimaryKey(true);
colInfo[15].setVisible(false);

QbReport report = new QbReport
    (parent,                // Parent
    QbReport.MASTERDETAILS, // Type of Report
    dbInfo,                   // Database Info
    colInfo,                // Column Mapping
    null);                  // Template Name
try {
 //    Apply Template with Formula and Script
 report.applyTemplate("MasterDetails", true);
} catch( Exception e ) {
 e.printStackTrace();
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/MasterDetails.zip]

Please see the full source code for the query details. The Exported Unformatted Results link shows how the report appears when it was first created without any formatting or templates. The Exported Formatted Results presents the outcome after applying the template.

This example uses the WoodView HSQL, so you will need to add database HSQL JDBC driver (`hsqldb.jar`) to your classpath. The driver is located in the `<EspressReportInstall>/lib` directory. For more information on using a database data source, see Appendix 2.B.1 - Data from a Database.

Since there are a total of 16 columns in this example, we use a for-loop to set the column info. The first eight columns are set as master columns and will be shown in the Group Header section. Column `0`, `OrderID`, is set as the primary key.

We apply the template after constructing the `QbReport` object in order to import the formula and scripts stored in the template.

You can also position the master section to be side by side with the rest of the columns. To do so, use the following `QbReport` constructor and pass in true for the `sideBySideLayout` parameter.

```
QbReport(java.lang.Object parent, int reportType, IDatabaseInfo dbinfo,
 ColInfo[] mapping, java.lang.String template, boolean sideBySideLayout)
```

2.C.5. Mailing Labels

The *Mailing Labels* report presents data in a columnar format without any groupings or breaks. It is similar to the *Simple Columnar* report, all you need to do is merely define the `ColInfo` array object with respect to the data source. For instance, given the dataset below:

Name	Company	Region
Mary	ABC Inc.	New York
Peter	GHI Ltd	London

Original Data

The column mapping is set so that the columns of the data source are mapped to the report. Thus Column `0` of the report corresponds to column `0` of the data source, Column `1` of the report to column `1` of the data source and so on and so forth. Thus the following report is created:

Name:	Mary
Company:	ABC Inc.
City:	New York
Name:	Peter
Company:	GHI Ltd
City:	London

Report after mapping and setting the Table Header Names

2.C.5.1. Column Mapping

To generate the aforementioned report, the following column mapping has to be set using the API:

```
ColInfo[] colInfo = new ColInfo[3];
colInfo[0] = new ColInfo(0);
colInfo[1] = new ColInfo(1);
colInfo[2] = new ColInfo(2);
colInfo[2].setName("City");
```

In the above code, the name for the third column was also changed to `City` rather than keeping the Header names defined in the original data.

The column mapping need not be in order or follow the same order as the data. You can also selectively choose the columns you desire. For instance, given a data source that has multiple columns, you can write the following code:

```
ColInfo[] colInfo = new ColInfo[3];
colInfo[0] = new ColInfo(2);
colInfo[1] = new ColInfo(7);
colInfo[2] = new ColInfo(4);
```

Here, the Column `0` of the report is mapped to column `2` of the data source, Column `1` of the report is mapped to column `7` of the data source, and Column `2` of the report is mapped to column `4` of the data source.

2.C.5.2. Creating the Report

The following example shows how to create a *Mailing Labels* report.

```
String xmlfilename = "Inventory.xml";
String dtdfilename = "Inventory.dtd";
String xmlcondition = "/Inventory/Category/Product/UnitsInStock < 15";

XMLFieldInfo[] fields = new XMLFieldInfo[8];
fields[0] = new XMLFieldInfo(new String[] { "Inventory", "Category" },
 "CategoryName");
fields[1] = new XMLFieldInfo(new String[] { "Inventory", "Category",
 "Product" },
  "ProductID");
fields[2] = new XMLFieldInfo(new String[] { "Inventory", "Category",
 "Product",
  "ProductName" });

fields[3] = new XMLFieldInfo(
  new String[] { "Inventory", "Category", "Product", "UnitPrice" });
fields[3].setElementDataType(DTDDataType.DOUBLE);

fields[4] = new XMLFieldInfo(
  new String[] { "Inventory", "Category", "Product", "Material" });
fields[5] = new XMLFieldInfo(new String[] { "Inventory", "Category",
 "Product", "Material",
  "Units" });
fields[5].setElementDataType(DTDDataType.INT);

fields[6] = new XMLFieldInfo(new String[] { "Inventory", "Category",
 "Product", "Material",
  "CostPerUnit" });
fields[6].setElementDataType(DTDDataType.DOUBLE);

fields[7] = new XMLFieldInfo(new String[] { "Inventory", "Category",
 "Product",
  "UnitsInStock" });
fields[7].setElementDataType(DTDDataType.INT);

XMLFileQueryInfo xmlInfo = new XMLFileQueryInfo(xmlfilename, fields,
 xmlcondition, fields,
  dtdfilename, false, null);

ColInfo[] colInfo = new ColInfo[8];
for (int i = 0; i < 8; i++) {
 colInfo[i] = new ColInfo(i);
}

QbReport report = new QbReport
   (parent, // Parent
   QbReport.MAILINGLABELS, // Type of Report
   xmlInfo, // XML Info
   colInfo, // Column Mapping
   null, // Template Name
   false); // Side By Side

try {
 //    Apply Template with Formula and Script
 report.applyTemplate("MailingLabels.rpt", true);

} catch (Exception e) {
 e.printStackTrace();
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/MailingLabels.zip]

Exported Unformatted Results [https://data.quadbase.com/Docs71/help/manual/code/export/MailingLabelsNF.html]

Exported Formatted Results [https://data.quadbase.com/Docs71/help/manual/code/export/MailingLabels.html]

The Exported Unformatted Results link shows how the report appears when it was first created without any formatting or templates. The Exported Formatted Results presents the outcome after applying the template. This example uses a XML file as the data source, for more information on using an XML data source, see Appendix 2.B.3 - Data from an XML Data Source.

We apply the template after constructing the `QbReport` object in order to import the formula and scripts stored in the template.

2.C.6. Formula

You are not restricted to just using the original data in generating the columns for the report. You may create your own columns with your own value or use formula to calculate over one or more columns. For instance, given the dataset below:

Units Sold	Product	Unit Price
12	Chair	$25
2	Table	$38
18	Cabinet	$21
4	Sofa	$57

Original Data

If Column `0` (`Units Sold`) and Column `2` (`Unit Price`) were multiplied, you can get a new column, which would have the total revenue for that product, thus producing the following report:

Product	Units Sold	Unit Price	Total Revenue
Chair	12	$25	$300.00
Table	2	$38	$76.00
Cabinet	18	$21	$378.00
Sofa	4	$57	$228.00

Report after applying formula and creating a new column

To generate the aforementioned report, the following column mapping has to be set using the API:

```
ColInfo[] colInfo = new ColInfo[4];
colInfo[0] = new ColInfo(1);
colInfo[1] = new ColInfo(0);
colInfo[2] = new ColInfo(2);
IObject salesColumn =
 NumericObject.multiply(NumericObject.getColumnValue(1),
                     NumericObject.getColumnValue(2));
colInfo[3] = new ColInfo(salesColumn, "Total Revenue", Types.DOUBLE);
```

In the above code, the names for the columns are not set as the original names of the columns from the data source sufficed. For the newly created column, `Total Revenue` was specified as the header.

To learn more about formula, please see Section 1.8 - Using Formulas & the Formula Builder.

2.C.7. Parameterized Report

In addition to regular queries, you can pass in queries that have parameters and have the report prompt the user for values for the parameters, before generating the report. There is no limit to the number of parameters allowed in each query.

To use a parameterized query as your data source for your report, you must use the SimpleQueryFileInfo [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/reportdesigner/util/SimpleQueryFileInfo.html] class, which implements IQueryFileInfo [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/reportdesigner/util/IQueryFileInfo.html], to pass in the query information. Since `SimpleQueryFileInfo` is a subclass of `DBInfo`, you can use the same parameters to create a `SimpleQueryFileInfo` object.

```
SimpleQueryFileInfo(java.lang.String url, java.lang.String driver,
 java.lang.String username, java.lang.String password, java.lang.String
 query)
```

In addition to creating the query object, you must also create the parameters using either an array of `SimpleQueryInParam` for single valued parameters or an array of `SimpleQueryMultiValueInParam` for multi valued parameters. Finally, you must pass an instance of the parameter set to the query object using the method `setInParam(ParamSet)`.

The constructors for the single valued parameter and multi valued parameter are identical since one is a subclass of the other.

```
SimpleQueryInParam(java.lang.String paramName, java.lang.String promptName,
 boolean mapToColumn, java.lang.String tableName, java.lang.String
 columnName, int sqlType, java.lang.Object defaultValue, java.lang.Object
 value)
```

The `paramName` is the name for the parameter, having two parameters with the same name is permitted provided that they are created in separate parameter instances. The `promptName` is the message the user will see when they are asked to input the value for the parameter.

The parameters `mapToColumn, tableName` and `columnName` determines if you would like to specify a column from the database whose values will be used for the parameter input. Selecting `true` modifies the parameter `prompt` that the end user will see when previewing or running the report in the Report Viewer. If you map the parameter to a database column, the user will be prompted with a drop-down list of distinct values from which to select a parameter value. If you do not map, set the `tableName` and `columnName` to `null`, the user will have to type in the specific parameter value. If you are querying the database from a parameterized class file and using the class file as the datasource, keep in mind that you can not set `mapToColumn` to `true`. For an alternative method of mapping to a column in a parameterized class file, see Appendix 2.B.5 - Data passed in a Custom Implementation.

The `sqlType` determines the datatype of the parameter. Select the constant that matches the datatype from the `java.sql.Types` class. The `defaultValue` is displayed in the input box when the prompt first appears. If you selected to map to column and provide a default value that does not exist, the first element of the drop down list will be displayed. The last parameter is of type object and if this parameter is not `null`, the parameter prompt will not be displayed. Instead of allowing the user to specify the value, it will use the value in this parameter for the query.

2.C.7.1. Creating the Report with Single Value Parameter

The following example shows how to create a report with a single value parameter.

```
SimpleQueryInParam inParam = new SimpleQueryInParam(
  "param", "Please select", true, "Categories",
  "CategoryName", Types.VARCHAR, "Arm Chairs", null);
SimpleQueryInParam[] paramSet = { inParam };

SimpleQueryFileInfo reportInfo = new SimpleQueryFileInfo(
  "jdbc:hsqldb:woodview",
  "org.hsqldb.jdbcDriver",
  "sa", "",
  "...");
reportInfo.setInParam(paramSet);
//  End Code : Adding Parameter Info

//  Begin Code : Setting up Column Mapping
ColInfo colInfo[] = new ColInfo[4];
for (int i = 0; i < colInfo.length; i++) {
 colInfo[i] = new ColInfo(i);
}
colInfo[0].setPrimaryKey(true);
colInfo[1].setName("Product Name");
colInfo[2].setName("Price");
colInfo[3].setName("Units In Stock");
//  End Code : Setting up Column Mapping

QbReport report = new QbReport(parent,
  QbReport.MASTERDETAILS,
  reportInfo,
  colInfo,
  "SingleValueParameter.rpt");
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SingleValueParameter.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/SingleValueParameter.html]

This example uses the WoodView HSQL database, so you will need to add database HSQL JDBC driver (`hsqldb.jar`) to your classpath. The driver is located in the `<EspressReportInstall>/lib` directory. For more information on using a database data source, see Appendix 2.B.1 - Data from a Database.

Here the parameter is mapped to the `CategoryName` column of the `Category` table. The prompt will display “Please Select” and the values will be presented in a drop down list for the user to select.

2.C.7.2. Creating the Report with Multi Value Parameter

You can also assign a parameter to have multiple values, for example, in the case where a user wants to check against a range of values rather than just a single value. The range of values is usually specified within the `IN` clause of a SQL query. Note that EspressReport only considers parameters within the `IN` clause to be multi-value.

The following example shows how to create a report with a multi-value parameter.

```
SimpleQueryMultiValueInParam inParam = new
 SimpleQueryMultiValueInParam("param",
  "Please select", true, "products", "productid", Types.INTEGER, new
 Integer(1), null);
SimpleQueryMultiValueInParam[] paramSet = { inParam };

SimpleQueryFileInfo rootInfo = new SimpleQueryFileInfo(
  "jdbc:hsqldb:woodview",
  "org.hsqldb.jdbcDriver",
  "sa", "",
  "select productid, productname " +
  "from Products where productid in (:param)");
rootInfo.setInParam(paramSet);
//  End Code : Adding Parameter Info

//  Begin Code : Setting up Column Mapping
ColInfo rootColInfo[] = new ColInfo[2];
rootColInfo[0] = new ColInfo(0);
rootColInfo[0].setName("Product ID");
rootColInfo[1] = new ColInfo(1);
rootColInfo[1].setName("Product Name");

//  End Code : Setting up Column Mapping

QbReport rootReport = new QbReport(
  parent,
  QbReport.MAILINGLABELS,
  rootInfo,
  rootColInfo,
  "MultiValueParameter.rpt");
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/MultiValueParameter.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/MultiValueParameter.html]

2.C.8. Sub-Report

You can also add sub-reports to the main report. This helps in presenting more data as two reports can be shown in one. For more information on sub-reports, please refer to Section 1.11 - Sub-Reports.

To create a report containing a sub-report, you need to create the sub-report object and then insert that sub-report object as a report cell object in the main report. The sub-report can be placed anywhere in the main report. To create a `SubReportObject`, use the following static method.

```
SubReport.createSubReport(QbReport parentReport, int reportType,
 IDatabaseInfo dbinfo, ColInfo[] mapping, java.lang.String template)
```

Similar to the `QbReport` constructors, there are numerous variations for the parameters to compensate for different data sources. The above method retrieves the data from a database. The sub-report can also contain parameters and you can set up parameter sharing from the API as well. Once the `SubReportObject` is created, it can be inserted into the report using the `addData(subReportObject)` method.

2.C.8.1. Creating the Report

The following example shows how to create a report that contains a sub-report.

```
//  Begin Code : Creating the sub report
DBInfo subReportInfo = new DBInfo(
  "jdbc:hsqldb:woodview",
  "org.hsqldb.jdbcDriver",
  "sa",
  "",
  "SELECT Employees.FirstName + ' ' + Employees.LastName AS Employee, "
    +
    "Count(Order_Details.OrderID) AS Orders, "
    +
    "Sum( ( Products.UnitPrice + Order_Details.StainCost )  *
 Order_Details.Quantity) AS Sales "
    +
    "FROM Employees, Products, Orders, Order_Details " +
    "WHERE ((Orders.EmployeeID = Employees.EmployeeID) " +
    "AND (Orders.OrderID = Order_Details.OrderID) " +
    "AND (Products.ProductID = Order_Details.ProductID)) " +
    "AND (((Orders.OrderDate BETWEEN '2003-01-10' AND '2003-31-12'))) " +
    "GROUP BY Employees.FirstName + ' ' + Employees.LastName;");

ColInfo subReportColInfo[] = new ColInfo[3];
for (int i = 0; i < subReportColInfo.length; i++) {
 subReportColInfo[i] = new ColInfo(i);
}

try {
 //   createSubReport(root report, report type, sub report data source, sub
 report column mapping, template)
 SubReportObject reportCell = quadbase.reportdesigner.ReportAPI.SubReport
   .createSubReport(
     rootReport,
     QbReport.COLUMNAR,
     subReportInfo,
     subReportColInfo,
     "SubReportSub.rpt");
 reportCell.setWidth(rootReport.getTable().getHeader().getWidth() - 2);
 reportCell.setHeight(2);
 //   Or use the following method if the height of the subreport may change
 //   reportCell.setResizeToFitContent(true);

 reportCell.setY(0);
 reportCell.setX(2);

 //   Add SubReport to the Table Header
 rootReport.getTable().getHeader().addData(reportCell);
 rootReport.getTable().getHeader().setHeight(1.8);
} catch (Exception ex) {
 ex.printStackTrace();
}
//  End Code : Creating the sub report
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SubReport.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/SubReport.html]

The main report in this example obtains the data using a `DBInfo` object to create a summary break report. Both reports apply a template to reduce the amount of code in the example. The sub-report is created and inserted to the table header section of the main report. Notice that several lines of code are necessary to adjusts the dimensions of the `SubReportObject` and the table header section in order to see the entire sub-report.

You can also include sub-reports that take a parameter from the main report and use it to populate the data within the sub-report. This helps to put in relevant data in sections within the main report. A parameterized sub-report is created by passing in a parameterized query for the sub-report and then specifying the column from the main report that provides the parameter for the sub-report's query. The parameter passed to the sub-report is the first value of a given column.

Depending on the placement of the Sub-Report, you can have a linked sub-report for each distinct value of the column. For example, in a *Summary Break* report, you can have a sub-report whose parameter is linked to the row break column. If this sub-report is then placed in the Group Header section, a sub-report is generated in the main report for each distinct value in the row break column (since the column breaks on each distinct value).

A linked sub-report will always take the first value in the column mapped to its parameter. However, depending on the placement of the sub-report, it can appear multiple times within the report, each time taking the first value of the column within its grouping.

Given below is an example of a sub-report placed in the group header that takes in a parameter from the main report.

```
QbReport.setEspressManagerUsed(false);

//  Begin Code : Creating the root Report
DBInfo rootInfo = new DBInfo("jdbc:hsqldb:woodview",
  "org.hsqldb.jdbcDriver",
  "sa",
  "",
  "SELECT Customers.Company, Orders.OrderID, " +
    "Products.ProductName, Products.UnitPrice, " +
    "Order_Details.Quantity " +
    "FROM Order_Details, Products, Orders, Customers " +
    "WHERE (Orders.CustomerID = Customers.CustomerID) " +
    "AND (Order_Details.OrderID = Orders.OrderID) " +
    "AND (Products.ProductID = Order_Details.ProductID);");

ColInfo rootColInfo[] = new ColInfo[5];
rootColInfo[0] = new ColInfo(0);
rootColInfo[0].setRowBreak(true);
ootColInfo[1] = new ColInfo(1);
rootColInfo[1].setRowBreak(true);
rootColInfo[2] = new ColInfo(2);
rootColInfo[2].setAggregation(false, ColInfo.NONE);
rootColInfo[3] = new ColInfo(3);
rootColInfo[3].setAggregation(false, ColInfo.AVG);
rootColInfo[4] = new ColInfo(4);
rootColInfo[4].setAggregation(false, ColInfo.SUM);

QbReport rootReport = new QbReport(
  parent,
  QbReport.SUMMARY,
  rootInfo,
  rootColInfo,
  "ParameterizedSubReport.rpt");
//  End Code : Creating the root Report

int numberOfTableHeaderCells =
 rootReport.getTable().getHeader().getData().length;

//  Transferring Table Header cells to Row Break Zero Header
for (int i = 0; i < numberOfTableHeaderCells; i++)
{

 rootReport.getTable().getRowBreakHeader(0)
   .addData(rootReport.getTable().getHeader().getData(i));
 rootReport.getTable().getRowBreakHeader(0).getData(i).setY(1);
}

//  Deleting Table Header cells
for (int i = 0; i < numberOfTableHeaderCells; i++) {
 rootReport.getTable().getHeader().removeData(0);
}

//  Begin Code : Creating the SubReport
//  SimpleQueryInParam(name of Parameter, String to be displayed,
 MapToColumn?, tableName, ColumnName, SQL Type, DefaultValue, value)
SimpleQueryInParam inParam = new SimpleQueryInParam(
  "Company", "Please select", true, "Customers",
  "Company", Types.VARCHAR, "All Unfinished Furniture",
  "All Unfinished Furniture");
SimpleQueryInParam[] paramSet = { inParam };

SimpleQueryFileInfo subReportInfo = new SimpleQueryFileInfo(
  "jdbc:hsqldb:woodview",
  "org.hsqldb.jdbcDriver",
  "sa", "",
  "SELECT Customers.CustomerID, Customers.ContactName, " +
    "Customers.Address, Customers.City, " +
    "Customers.State, Customers.Zip " +
    "FROM Customers " +
    "WHERE (Customers.Company =:Company);");
subReportInfo.setInParam(paramSet);

//  Begin Code : Setting up Column Mapping
ColInfo subReportColInfo[] = new ColInfo[6];
for (int i = 0; i < subReportColInfo.length; i++) {
 subReportColInfo[i] = new ColInfo(i);
}

 //  End Code : Setting up Column Mapping

 try {

 //   createSubReport(root report, report type, sub report data source, sub
 report column mapping, template)
 SubReportObject reportCell = SubReport.createSubReport(
   rootReport,
   QbReport.COLUMNAR,
   subReportInfo,
   subReportColInfo,
   null);
 ((QbReport) reportCell.getSubReport()).applyTemplate(
   "ParameterizedSubReportSub.rpt", true);

 //   Get Parameter from root report column
 reportCell.setParameterMap(new String[]
 { rootReport.getTable().getColumn(0).getID() });

 //   Set SubReport cell's height and width
 reportCell.setWidth(7.2);
 reportCell.setResizeToFitContent(true);
 reportCell.setY(.3);
 reportCell.setX(0);

 //   Add SubReport to the Group Zero Header
 rootReport.getTable().getRowBreakHeader(0).addData(reportCell);

} catch (Exception ex) {
 ex.printStackTrace();
}
//  End Code : Creating the sub report

return (new Viewer().getComponent(rootReport));
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ParameterizedSubReport.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/ParameterizedSubReport.pdf]

You do not need to link a column from the main report to a sub-report's parameter if both the main report and the sub-report have a parameter with the same name.

2.C.9. Drill-Down

With the help of parameterized queries or classes, *Drill-Down* reports can be created. Instead of having reports with large amounts of data in it, you can show a top level report showing the minimum data required and then delve deeper on the selected data. For more information on *Drill-Down* reports, please refer to Section 1.10 - Drill-Down.

To create a *Drill-Down* report, you need to create the various `QbReport` objects (please note that all `QbReport` objects, other than the root report object, will have parameterized queries as their data source) and then specify the order of the drill-down as well as the column of the report to attach the next level of the *Drill-Down* report. To create a `Drill-Down` report use the following method:

```
rootReport.createDrillDownReport(java.lang.String name, int reportType,
 IDatabaseInfo dbInfo, ColInfo[] mapping, java.lang.String template, int[]
 columnMapping)
```

Unlike Sub-Reports, *Drill-Down* reports must be parameterized queries, so there are fewer construction methods for *Drill-Down* reports.

2.C.9.1. Creating the Report

The following example shows how to create a report that contains a *Drill-Down* report.

```
Component doDrillDownReport(Object parent) {
QbReport.setEspressManagerUsed(false);

//  Begin Code : Creating Report 1  - the Root Report
DBInfo rootInfo = new DBInfo("jdbc:hsqldb:woodview",
  "org.hsqldb.jdbcDriver",
  "sa",
  "",
  "select Categories.CategoryName, sum(Products.UnitsInStock) " +
    "from Categories, Products " +
    "where Categories.CategoryID = Products.CategoryID " +
    "group by Categories.CategoryName " +
    "order by Categories.CategoryName;"
  );

ColInfo rootColInfo[] = new ColInfo[2];
for (int i = 0; i < rootColInfo.length; i++) {
 rootColInfo[i] = new ColInfo(i);
}

rootColInfo[0].setName("Product Name");
rootColInfo[1].setName("Total Units In Stock");
QbReport rootReport = new QbReport(
  parent,
  QbReport.COLUMNAR,
  rootInfo,
  rootColInfo,
  "DrillDown.rpt");
//  End Code : Creating Report 1  - the Root Report
//  Begin Code : Creating Report 2  - the drill-down Report
SimpleQueryInParam inParam = new SimpleQueryInParam("param", "Please
 select", true,
  "Categories", "CategoryName", Types.VARCHAR, "Arm Chairs", null);

SimpleQueryInParam[] paramSet = { inParam };
SimpleQueryFileInfo levelOneReportInfo = new SimpleQueryFileInfo(
  "jdbc:hsqldb:woodview",
  "org.hsqldb.jdbcDriver",
  "sa",
  "",
  "select Categories.CategoryName, Products.ProductName,
 Products.UnitPrice, Products.UnitsInStock from Categories,
 Products where Categories.CategoryID=Products.CategoryID and
 Categories.CategoryName=:param order by Categories.CategoryName,
 Products.ProductName;");
levelOneReportInfo.setInParam(paramSet);
ColInfo levelOneReportColInfo[] = new ColInfo[4];
for (int i = 0; i < levelOneReportColInfo.length; i++) {
 levelOneReportColInfo[i] = new ColInfo(i);
}

levelOneReportColInfo[0].setName("Product Type");
levelOneReportColInfo[1].setName("Product Name");
levelOneReportColInfo[2].setName("Price");
levelOneReportColInfo[3].setName("Units In Stock");

try {
 //   createDrillDownReport(Name, Type of Report, database information,
 column mapping, template, column of root report to be mapped to)
 rootReport.createDrillDownReport("TestDrillDownReport",
   QbReport.COLUMNAR,
   levelOneReportInfo,
   levelOneReportColInfo,
   "lvl1.rpt", new int[] { 0 });
} catch (Exception ex) {
 ex.printStackTrace();
}

rootReport.getTable().getColumn(0).setDrillDownName("TestDrillDownReport");
//  End Code : Creating Report 2  - the drill-down Report
return (new Viewer().getComponent(rootReport));
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/DrillDown.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/DrillDown.html]

When you generate drill-down reports without using the EspressManager, you MUST have a sub directory called `DrillDown` under the working directory of the `.class` file.

The above exported results, only contain the root report and the *Drill-Down* report for `Oval Tables`.

When you export *Drill-Down* reports, to either DHTML or PDF format, you must have the `DrillDownReportServlet` servlet provided. This servlet provides the link in the report to the next level. Depending on the link clicked, this servlet delivers the appropriate next level report. To use this servlet, you must make it accessible from your servlet runner/application server. The URL to access this servlet MUST be `http://<machine name>:<port number>/servlet/DrillDownReportServlet`. Then in your application code, you must use the following method:

```
QbReport.setDynamicExport(boolean state, String serverName, int
 servletRunnerPort);
```

For example, given a *Drill-Down* report and the following method call:

```
report.setDynamicExport(true, "Aphrodite", 8080);
```

When the report is exported, the above code sees to it that the links are pointing to machine `Aphrodite` and the port `8080`. The `DrillDownReportServlet` servlet gets called using the URL, `http://Aphrodite:8080/servlet/DrillDownReportServlet` and thus computes the next level based on the link clicked and shows the next level.

You can also change the url for this servlet by using the `setServletDirectory(String)` method in the `QbReport` class. For instance, given the example above, if the following line of code were added:

```
report.setServletDirectory("ER/Test/");
```

then the links will show `http://Aphrodite:8080/ER/Test/DrillDownReportServlet` instead of the default servlet/ in the link.

Note that *Drill-Down* reports exported to formats other than HTML, DHTML, or PDF will only show the current level.

Chapter 3. Charting Guide

3.1. Introduction

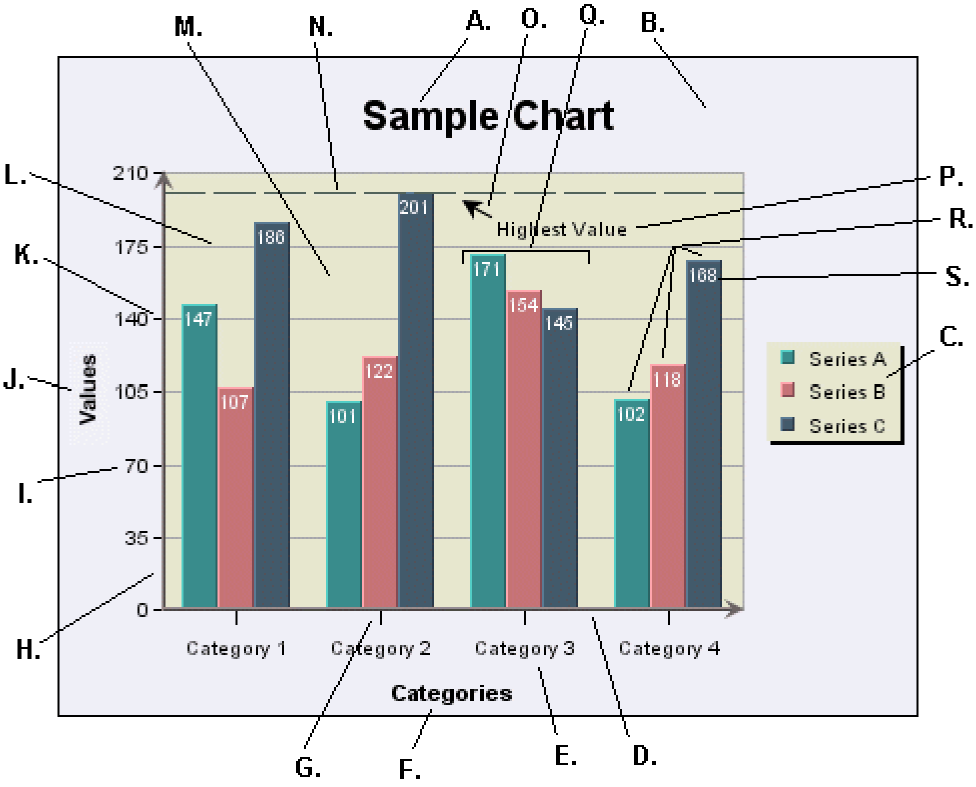
A sophisticated and powerful charting engine is incorporated with EspressReport that allows you to include many different types of visual data representation in reports. You can also deploy charts independently from reports, adding stand-alone graphs to web pages and applications. Charts can be created using the embedded Chart Designer interface or by using the charting APIs. This guide covers both designing charts via the Chart Designer and the API, as well as deploying charts with reports and independently.

3.2. Charting Basics

This section covers some of the basics of charts including the different parts of a chart and the way in which tabular data either from the report or another data source is mapped to a chart.

3.2.1. What is a Chart

The following diagram illustrates the various components that make up a chart. Also, this diagram refers to various terms that will be used throughout this guide.



A. Main Title

This is the main title for the chart.

B. Chart Canvas

This is the background on which the chart is drawn. The canvas serves as the size/boundary for the chart and it is generally the same size as any exported image. You can modify the canvas color or place/add an image file as a background.

C. Legend

This is the chart legend. The legend shows your category or series names along with a color point. Secondary values, as well as added trend/control lines and control areas, can also be displayed in the legend.

D. Category (X) Axis

This is the X or category axis of the chart. Generally, the category axis plots the distinct entries from the dataset for which you want to plot values in the chart. (The values are generally plotted on the Y-axis.) Certain chart types such as bar and Gantt reverse this by drawing the categories on the Y-axis, and plotting the values on the X-axis. Other chart types like scatter and bubble plots, plot values on each axis to create a point in 2D or 3D space.

E. X-Axis Labels These are the labels for the X-axis elements or categories.

F. X-Axis Title This is the X-axis title.

G. X Ticker These are the X-axis tickers. By default the tickers match each data point in the chart.

H. Value (Y) Axis This is the Y or value axis of the chart. Generally, the Y-axis plots the values for each of the categories. By default the scale of the Y axis is generated to provide a best fit for the dataset; however, it can be manually adjusted. For combination charts, the second value axis is drawn at the right-hand side of the plot.

I. Y-Axis Labels These are the labels for the Y-axis values.

J. Y-Axis Title This is the Y-axis title.

K. Y-Axis Ticker These are the Y-axis tickers.

L. Y Grid These are grid lines drawn along each scale step in the Y-axis. Grid lines can also be drawn for the points on the X-axis (and Z-axis for 3D charts).

M. Plot Area This is the area, bounded by the axes, where all the data points are plotted. You can fill the area with color and/or draw a border around the area, along with other options. The plot area can be moved and resized on the chart canvas.

N. Control Line This is one of the special types of lines that can be added to a chart. In this instance, it is a control line that follows the highest value in a series. Control lines can also be drawn for averages and multiples of standard deviation. Users can also add a variety of trend lines to charts.

O. Floating Line A floating line is an arbitrary line added to a chart. In this case it is being used as a pointer with an arrow. Floating lines move in relative position with the chart plot. They can also be used to create filled shapes.

P. Annotation Text This is a piece of arbitrary text added to a chart (not labels or titles). You can place text anywhere in the chart canvas. Like floating lines, annotation text moves in relative position to the chart plot.

Q. Category Elements This is the plot for a category element. There are three points plotted for each category because this chart has a data series.

R. Data Series Elements These are the individual points that make up a category. A series allows groups of data to be plotted on a single chart. For more about information about categories and series, see Section 3.2.2 - Basic Data Mapping

S. Data Top Labels These are labels placed at each data point in the chart that display the exact value for each point.

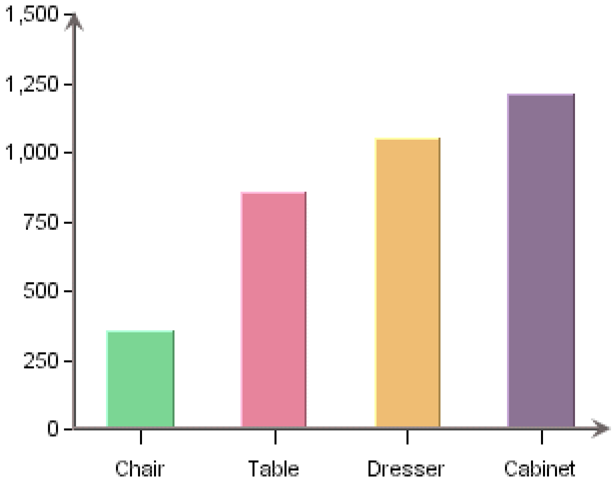
3.2.2. Basic Data Mapping

Data mapping is the way that raw data is rendered in the chart. Although data can be drawn from many sources, a chart looks for the basic structure of the data to be in the form of a table. Hence, data passed in as arguments, from the report or from XML files is converted to a table structure before mapping.

A basic set of data might look something like this:

Product	**Sales**
Chair	362
Table	862
Dresser	1052
Cabinet	1211

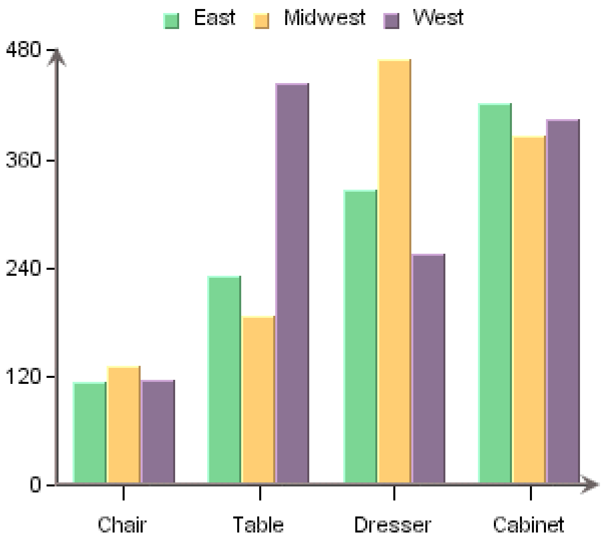
To plot this data in a chart, you would want to plot the **Sales** value for each entry in the **Product** column. Hence, the products are your category and the sales numbers are your values. In a chart you would map the **Product** column to the X (category) axis and the **Sales** column to the Y (value) axis. The resulting plot would look like this in a column chart:



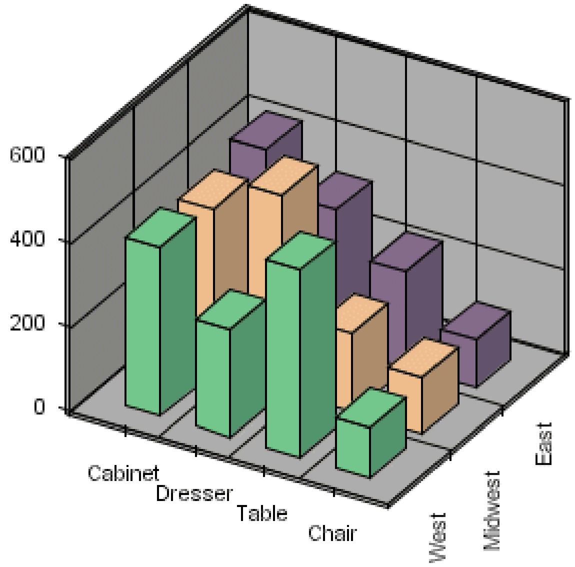
Here, a column is drawn to show the value for each distinct element in the category column. On top of the basic category values, additional information can be displayed in the form of a data series. For example, say that there is another element to our data that shows sales data not only for product, but also over a sales region. Our adjusted table would look like this:

Product	**Region**	**Sales**
Chair	East	114
Chair	Midwest	131
Chair	West	117
Table	East	231
Table	Midwest	187
Table	West	444
Dresser	East	327
Dresser	Midwest	469
Dresser	West	256
Cabinet	East	422
Cabinet	Midwest	386
Cabinet	West	403

In order to show the value for each region per product we could add the **Region** column to the data mapping as a data series. Doing this gives us the following chart:



Now each category has three data points, one for each region. For two-dimensional charts the series is always displayed in-line. In three-dimensional charts, the series is drawn on the Z-axis by default, although it can be drawn in-line as well. Below, is the same chart show in 3D.



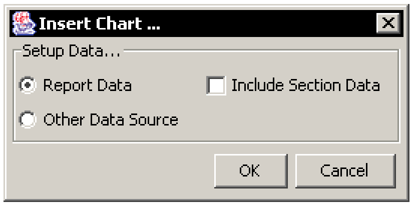
In this chart, the data series is drawn along the Z-axis. Note that the order of the categories has been changed to provide a better view of the data. This is the basic concept behind data mapping. Most chart types use this mapping technique or mapping options similar to this. Detailed data mapping instructions for each chart type are available in Section 3.4 - Chart Types and Data Mapping.

3.3. Creating a Chart

To create a chart, you can select *Insert Chart* from the *Insert* menu or select the *Insert Chart* icon on the toolbar. A small rectangle will follow your cursor around the Design window. Position the cursor where you would like to add the chart and click. This will bring up a dialog prompting you to specify data source options for the chart.

3.3.1. Data Source Options

The following options are available for obtaining a data source for your chart.


Insert Chart ...

Setup Data...

● Report Data □ Include Section Data

○ Other Data Source

Chart Data Options

Report Data: This will plot the chart using the data contained in the report. This data includes all of the report columns and computed columns that have been added using the report functions. When using report data for the chart you also have the option to include section data. This will take any formulas or labels from the report section in which the chart is placed and make them available to the chart. Data for labels and formulas is imported as additional columns to the chart input data. The cell ID is used as the column name.

However, when you use report data, the chart cannot be deployed independently from the report and certain charting features like time-based zooming and histograms are not available.

Other Data Source: This will allow you to select a different data source than the report data. If you select this option, you will be returned to the Data Source Manager window where you can create/select a data source for the chart. You can use data from any source as you can for a report. For more information about data sources, please see Section 1.3 - Working with Data Sources.

When you use an independent data source the chart can later be deployed independently from the report. Also, all of the charting features are available. Note that some features like drill-down are only functional when the chart is deployed independently from the report.

3.3.1.1. Chart Parameter Linking

If you use a parameterized query as the chart data source, by default it will run with default values when the chart is run in the report. However, you can use the chart parameter linking feature for the chart to retrieve its parameter values based on column values in the report. This feature works in the same manner as the sub-report linking feature described in Section 1.11.4 - Linked Sub-Reports. To link a parameterized chart to a column, first insert a parameterized chart into a report. Then select *Chart Parameter Mapping* from the *Data* menu. This will bring up a dialog allowing you to map column fields from the report to query parameters in the chart.

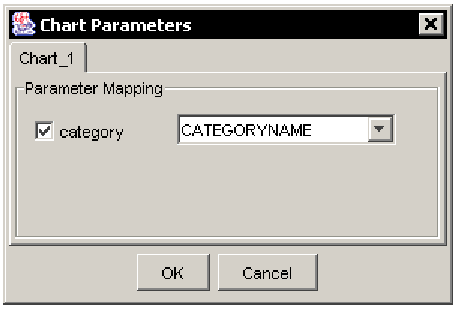


Chart Parameters

Chart_1

Parameter Mapping

☑ category CATEGORYNAME

Chart Parameter Mapping Dialog

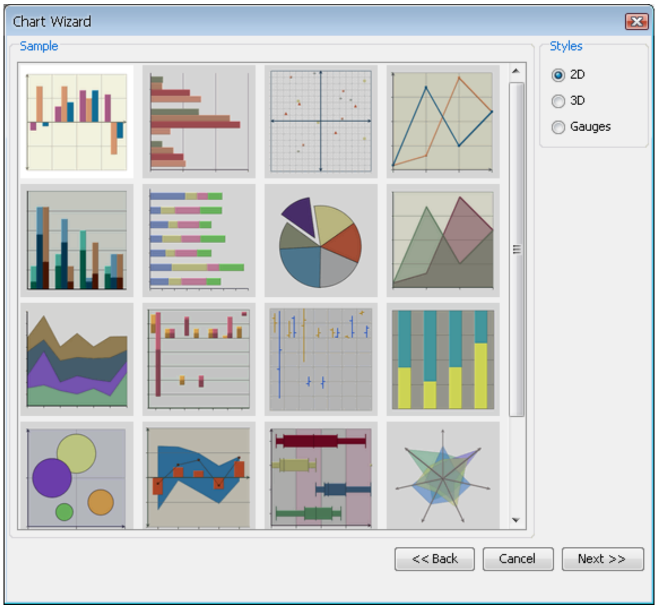
The dialog contains a tab for each parameterized chart. Each parameter in the chart can be mapped to a different column field in the report.

3.3.2. Multiple Data Sources

If you have selected to use an independent data source for the chart, you can use multiple data sources to build a composite table as you can with report. The feature works in the same manner as in the report wizard. For more information, please see Section 1.3.8 - Data from Salesforce.

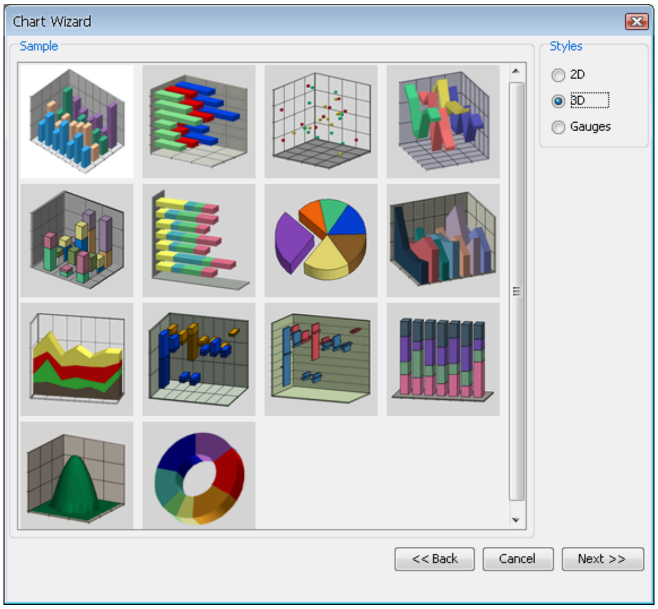
3.4. Chart Types and Data Mapping

Once you have selected the data that you would like to use for the chart, the next step in the Chart Wizard is to select the chart type that you would like to use. You will be presented with a dialog that allows you to specify the chart type.



Two-Dimensional Chart Types Selection Dialog

Each chart type represents a different way in which the data points are plotted to give a pertinent representation to all kinds of data. The different types of charts are broken down by dimension. In addition to basic chart types, users can create many different types of composite/combination charts by adding secondary values/series to the chart. You can toggle between the chart categories by selecting either *2D*, *3D* or *Gauges* in the right-hand side of the chart types dialog.



Three-Dimensional Chart Types Selection Dialog

This dialog enables you to select your chart type by either selecting the image and clicking *Next* or by double clicking on the chart image. You can select from one of the following chart types:

- Column

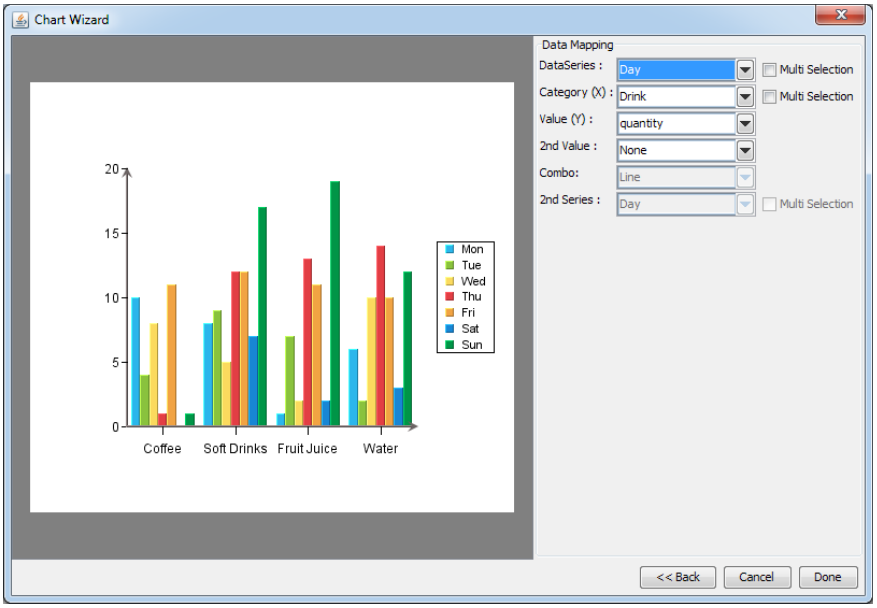
- Bar
- XY(Z) Scatter
- Line
- Stack Column
- Stack Bar
- Pie
- Area
- Stack Area
- High Low
- HLCO
- Percentage Column
- Doughnut
- Surface (Three-Dimensional Only)
- Bubble (Two-Dimensional Only)
- Overlay (Two-Dimensional Only)
- Box (Two-Dimensional Only)
- Radar (Two-Dimensional Only)
- Dial (Two-Dimensional Only)
- Gantt (Two-Dimensional Only)
- Polar (Two-Dimensional Only)
- Circular Gauge
- Square Gauge
- Semi Circular Gauge
- Square Gauge
- Quarter Circular Gauge
- Heatmap

Each of the chart types is described in detail later in this chapter, starting with Section 3.4.2 - Column Charts

For information on gauges, see Section 3.4.20.2 - Gauges

3.4.1. Data Mapping

After you have selected the chart type, the next step is to specify the data mapping for the chart. Data mapping is the way that the selected data source is rendered in the chart. The basics of data mapping are described in Section 3.2.2 - Basic Data Mapping. The data mapping screen in the Chart Wizard allows you to set the mapping options and also preview the results.



Data Mapping

DataSeries : Day ☐ Multi Selection

Category (X) : Drink ☐ Multi Selection

Value (Y) : quantity

2nd Value : None

Combo: Line

2nd Series : Day ☐ Multi Selection

Data Mapping Dialog

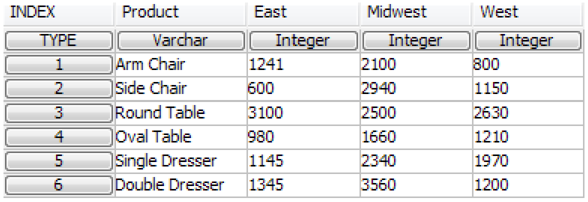
The left-hand side of the dialog shows a preview of your chart. The right-hand side shows which columns from your data source have been selected to plot in the different chart elements (series, category, value, etc). By default, EspressReport will select the first available columns, based on the data type, to plot. Changing the data mapping is easy - click on the down-facing arrow next to a data field and select a different field. The chart preview in the left-hand part of the data mapping dialog will be updated immediately.

The specific mapping options vary for each chart type and are discussed later in this chapter starting with Section 3.4.2 - Column Charts.

3.4.1.1. Data Transposition

You can also set data transposing from the data mapping dialog. Transposing allows you to plot multiple data source columns in one chart without modifying the data source.

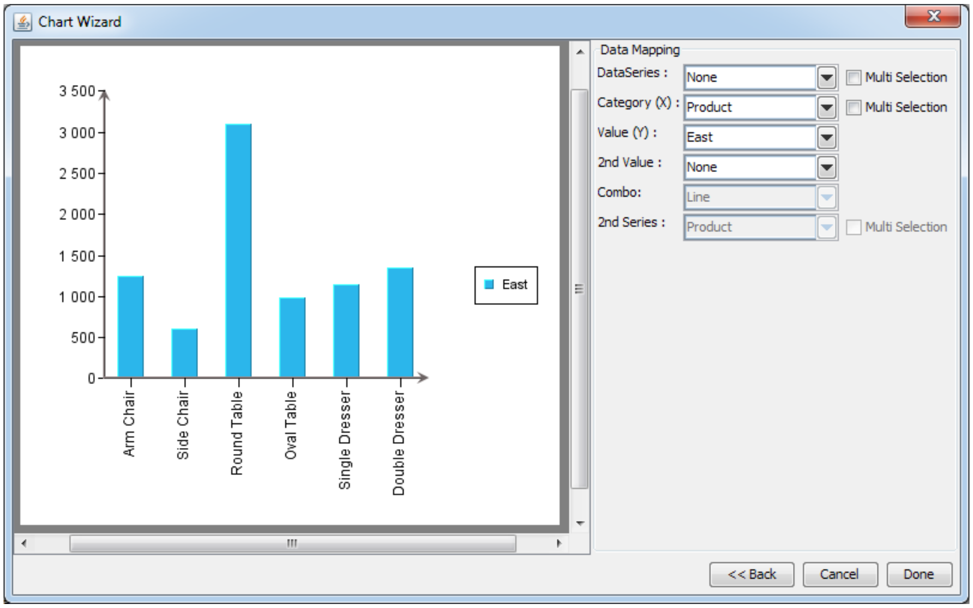
For example, we have a data source that looks like this:



INDEX	Product	East	Midwest	West
TYPE	Varchar	Integer	Integer	Integer
1	Arm Chair	1241	2100	800
2	Side Chair	600	2940	1150
3	Round Table	3100	2500	2630
4	Oval Table	980	1660	1210
5	Single Dresser	1145	2340	1970
6	Double Dresser	1345	3560	1200

Example Data Source

We would like to plot data from all regions in one chart. The default mapping without data transposing does not allow us to do that. We would have to plot only one region in the chart or modify the data source, so we will have to transpose the data source.



Data Mapping

DataSeries : None ☐ Multi Selection

Category (X) : Product ☐ Multi Selection

Value (Y) : East

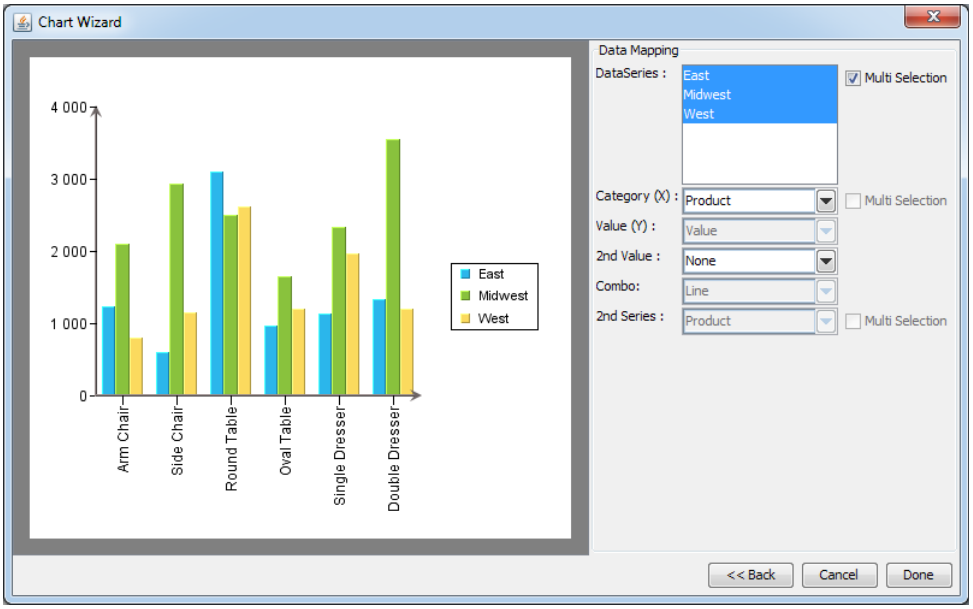
2nd Value : None

Combo: Line

2nd Series : Product ☐ Multi Selection

Default Chart Data Mapping

To transpose the data, select the *Multi Selection* option next to the *Data Series* field. The field will change to a list allowing us to select several columns at the same time. We will just select all region columns and the chart will look like this:



Data Mapping

DataSeries : East, Midwest, West ☑ Multi Selection

Category (X) : Product ☐ Multi Selection

Value (Y) : Value

2nd Value : None

Combo: Line

2nd Series : Product ☐ Multi Selection

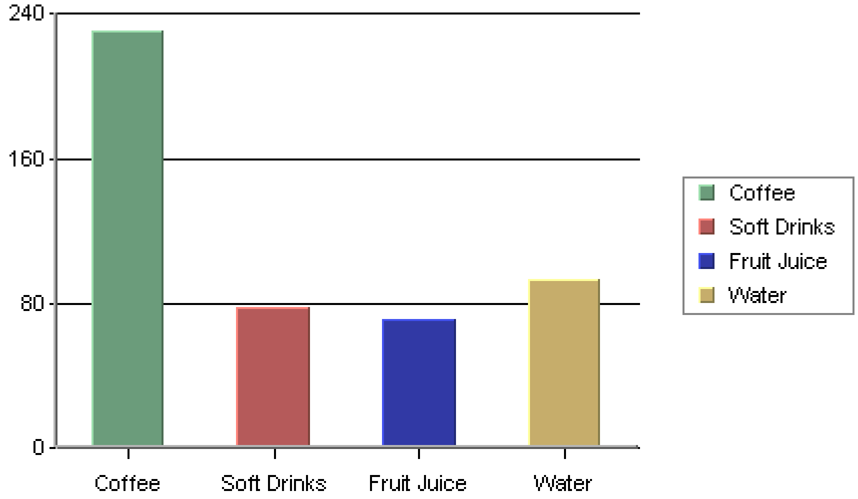
Default Chart Data Mapping

Note

Only one chart element (such as data series, category, value, or second series) can be transposed at a time. If you select the *Multi Selection* checkbox for one chart element, the *Multi Selection* checkboxes will be deactivated for other chart elements.

Transposing is not available for drill-down charts.

3.4.2. Column Charts

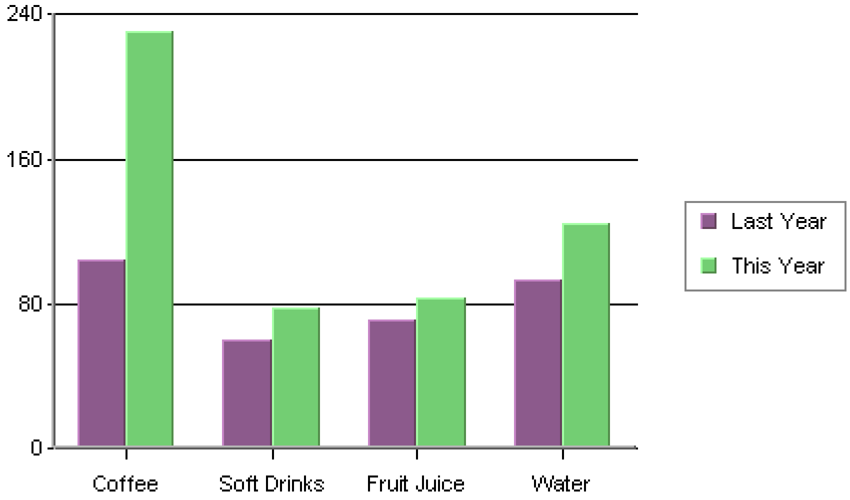


Column Chart (Without Data Series)

A column chart displays each row in the data table as a vertical bar (or column). The categories are listed along the X-axis and the values plotted along the Y-axis. Column charts are good for comparing discrete values for different groups. Each group is represented by a different color.

In a two-dimensional chart, if a data series has been selected then the entire series for a single category will be displayed in the XY plane. If a three-dimensional column chart is being used then all the vertical bars in a given data series are drawn using the same color along the Z-axis. If a data series is not present in the chart, the categories will be represented by the same color by default. Different colors for the categories are possible to set by clicking the *Change chart options* icon on the toolbar (or selecting Format → Chart Options), and unchecking the *Single Color For All Categories* option.

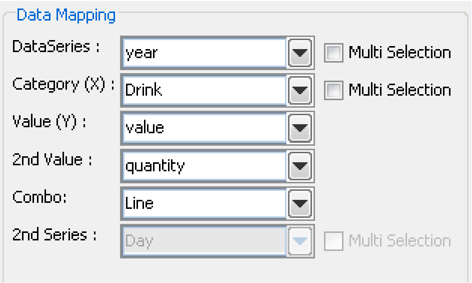
In the examples given, we have chosen Drink as the category variable and Value as the value variable (most examples use the `Sample.dat` file included in the EspressReport installation). Year has been selected as the data series variable for the chart below. Therefore, each name has two columns shown: one for *last year* and one for *this year* which are the only two values present in the data series column.



Column Chart with Data Series

3.4.2.1. Data Mapping

Data mapping for column charts is fairly straightforward. It is similar to the examples first presented in Section 3.2.2 - Basic Data Mapping. For column charts, the following options are available:



Data Mapping

DataSeries : year ☐ Multi Selection

Category (X) : Drink ☐ Multi Selection

Value (Y) : value

2nd Value : quantity

Combo: Line

2nd Series : Day ☐ Multi Selection

Mapping Options for Column Charts

Data Series: Choose a data column whose distinct values will determine the number of data series in the chart. Each element in a data series is drawn using the same set of colors and other attributes.

Category (X): Choose a data column whose distinct values will determine the categories.

Value (Y): Choose a data column to provide values for each category.

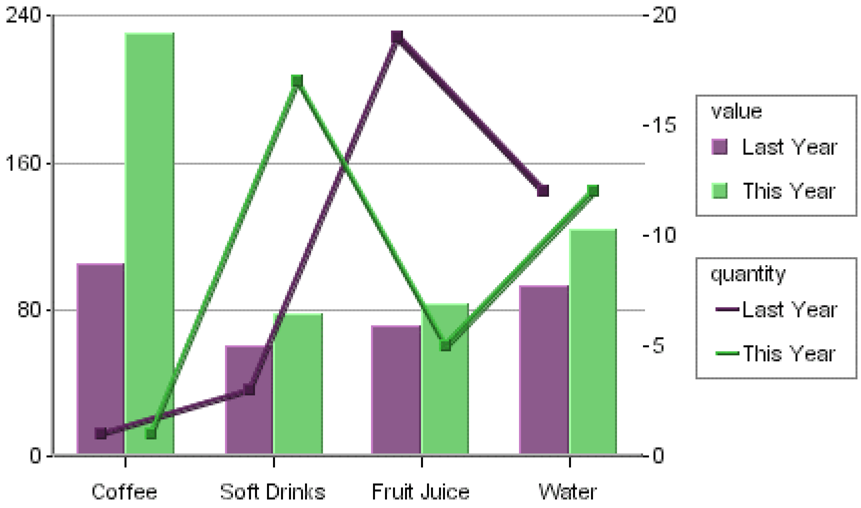
2nd value: Add a second value to create a combination chart.

2nd Series: Choose another column to be series for the secondary chart. This option is applicable only if the secondary chart is an overlay chart.

Combo: Choose the chart type for the secondary chart. For column charts the combo options are *Line* and *Overlay*.

The data mapping also allows you to transpose the data (in other words: to select several columns for a single category). To learn more about data transposition, please see Section 3.4.1.1 - Data Transposition.

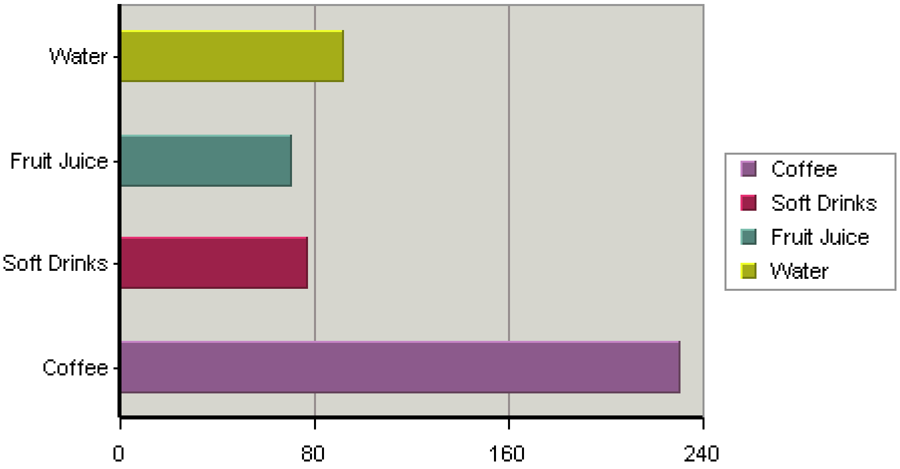
The last three data mapping options allow you to add a second value to the chart. ChartDesigner supports secondary values for all chart types except pie, radar, bubble, dial, surface, and scatter charts. In the example below, a second value of Quantity has been added to our column chart.



Column Chart With 2nd Value

As you can see, the second axis labels are drawn on the right-hand side of the plot area (you can choose to make this axis visible). Usually, the second value will share the same categories and series as the primary value. However, for two-dimensional column, stack column, stack area, high low, HLCO, and percentage column charts you can specify an overlay combination, which allows you to specify a second series. For more on overlay charts, please see Section 3.4.17 - Overlay Charts.

3.4.3. Bar Charts

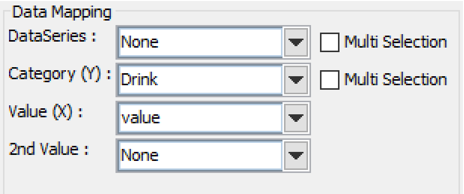


Bar Chart

A bar chart is essentially the same as a column chart, except that horizontal bars are drawn in the chart as opposed to the vertical bars which are used in a column chart. In a bar chart, the categories are plotted along the Y-axis and the values along the X-axis.

Just as for a column chart, if a data series has been selected then the entire series for a single category is displayed in the XY plane. If a three-dimensional bar chart is being used, all the horizontal bars in a data series are drawn using the same color. Each category is drawn using a different color. If the data series is not present in the chart, the categories will be represented by the same color by default. Different colors for the categories are possible to set by clicking the *Change chart options* icon on the toolbar (or selecting Format → Chart Options), and unchecking the *Single Color For All Categories* option.

3.4.3.1. Data Mapping



Data Mapping

DataSeries : None ☐ Multi Selection

Category (Y) : Drink ☐ Multi Selection

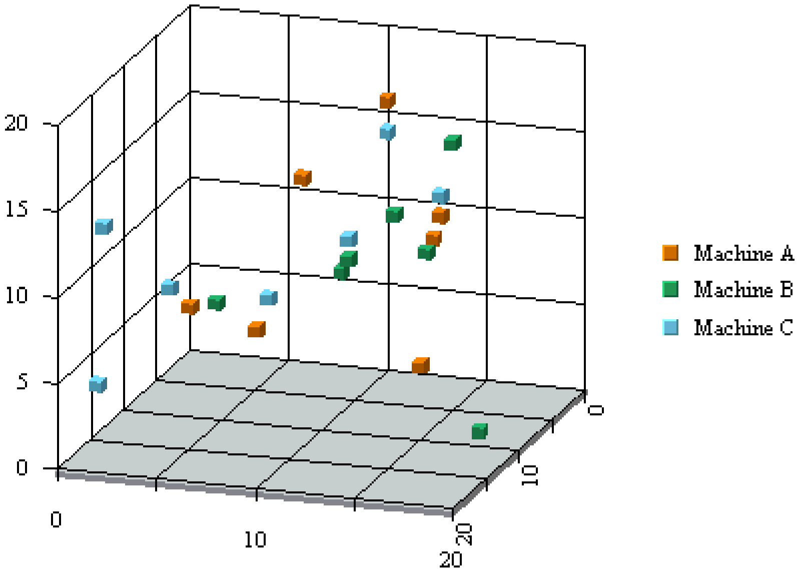
Value (X) : value

2nd Value : None

Mapping Options for Bar Charts

The mapping for this chart type is similar to that of a column chart, except the category (X) and values (Y) under column chart become category (Y) and values (X) respectively. This is because values are represented vertically in a column chart, but horizontally in a bar chart. Please note that the *2nd Series* and *Combo* options are not available for bar charts. This is because the only combination available with bar charts is a line.

3.4.4. XY(Z) Scatter Charts

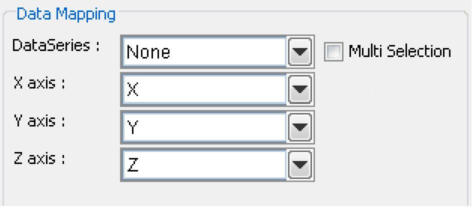


XY(Z) Scatter Chart

In an XY(Z) scatter chart, each selected row in the data table defines a point in two or three-dimensional space. Thus each column must contain either numeric or date/time values. A marker represents each point. The data columns that are in each row of the data table determine the spatial position of the marker.

Optionally, another data table column can be chosen to separate the markers into groups. Elements of each group have the same value on this column which is referred to as the data series column. Markers in the same group are drawn using the same drawing attributes; in other words, using the same shapes and colors. The X-axis scale of a scatter chart is linear. This means that unlike other chart types, the data points may or may not be evenly spaced along the X-axis.

3.4.4.1. Data Mapping


Data Mapping

DataSeries : None ☐ Multi Selection

X axis : X

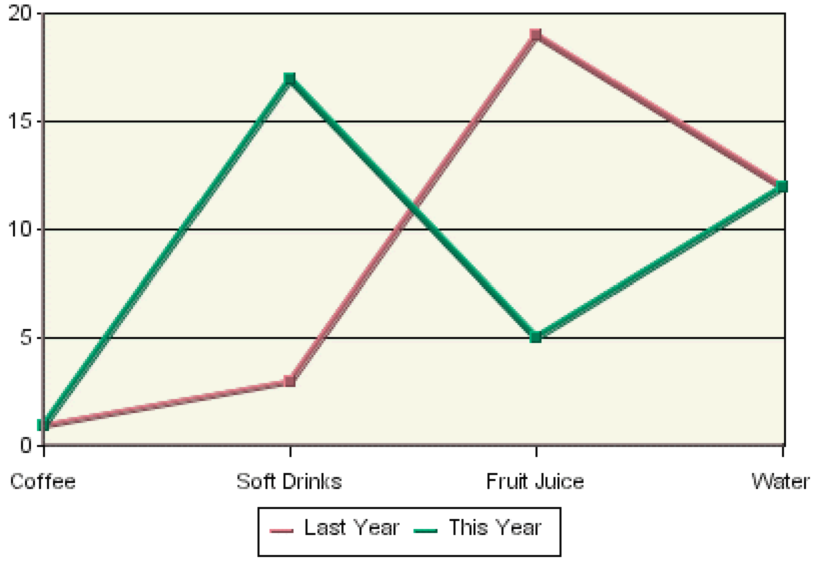
Y axis : Y

Z axis : Z

Mapping Options for Scatter Charts

In a scatter chart, the X-axis, Y-axis, and Z-axis values determine the X, Y, and Z coordinates of a point respectively. The data series box allows you to choose a data column whose distinct values will determine the number of data series in the chart. Each element in a data series is drawn using the same set of drawing attributes(e.g., colors and markers).

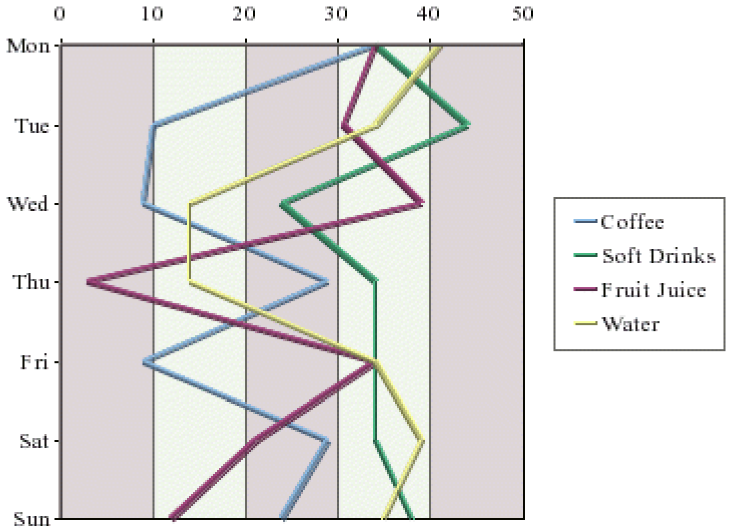
3.4.5. Line Charts



Line Chart

Line charts present data in a manner similar to column charts. For a two-dimensional line chart, a marker denotes a data point for each category. The value column, which must be numeric, determines the height of a marker. Each marker is joined with a line. If the chart has a data series, a separate line is drawn for each element in the series. Please note, the markers (points) are not shown by default. You can enable showing markers (points) in the *Line and Point* dialog. This dialog can be opened from the *Format* menu or by clicking the *Format line and points* icon on the toolbar.

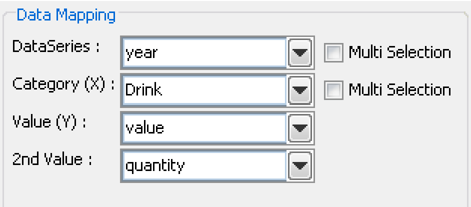
ChartDesigner allows 2D line charts to be displayed vertically in addition to the default horizontal setting. To use this feature in Chart Designer, create a line chart and then select Format → Chart Options. Next, select *Vertical*. The chart is rotated clockwise 90 degrees.



Vertical Line Chart

A three-dimensional line chart is an extension of its two-dimensional counterpart. It contains no additional information. The markers disappear (they are not available for 3D line charts) and thicker lines, spaced apart on the Z-axis, replace the thin lines.

3.4.5.1. Data Mapping



Data Mapping

DataSeries : year ☐ Multi Selection

Category (X) : Drink ☐ Multi Selection

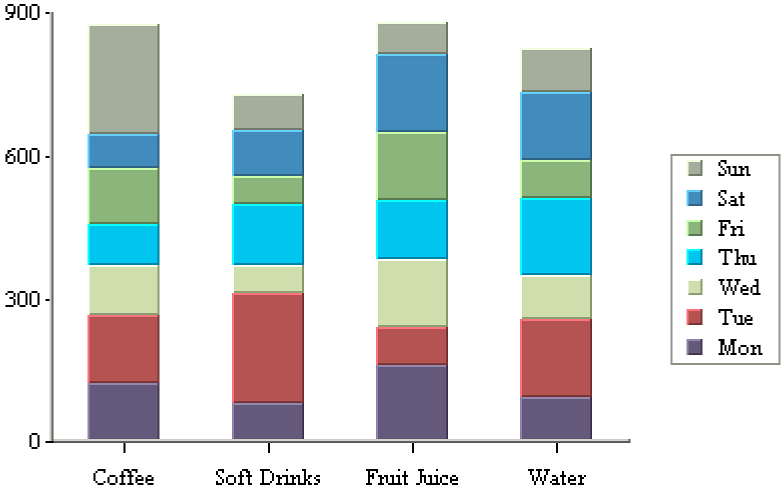
Value (Y) : value

2nd Value : quantity

Mapping Options for Line Charts

Data mapping for line charts is almost exactly the same as for column charts (discussed in Section 3.4.2.1 - Data Mapping). Line charts, however, do not have the *2nd Series* and *Combo* options. This is because the only combination available with line charts is a line. To create line combinations with other chart types (bar, column, stack column, etc) select the other chart type as the primary chart, and select *Line* as the *Combo* option.

3.4.6. Stack Column Charts



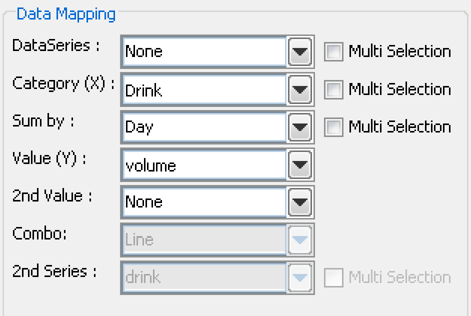
Stack Column Chart

A stack column chart is a derivative of a column chart in which each vertical column comprises a stack of bars. Each selected row in the data table is represented by a component of a chart column. For a two-dimensional stack column chart, the categories are plotted on the X-axis and the values on the Y-axis. The value column, which must be numeric, determines the length of each bar component. A third column, known as the sum-by column, represents a group of values for each category. A stack for a given category value is built up stacking the sum-by values for that category value. Each stack component of a chart column has a distinct sum-by value denoted by a distinct color. Each row in the data table must have a unique pair of values on both the category and sum-by columns. Components with negative values are separated from components with positive values and they are stacked up in the “negative” direction below the X-axis.

In our example above, we have chosen types of drink as category value and day representing the sum-by values. Another independent category (to be displayed on the Z-axis) may be optionally specified based on a fourth column as a data series. In such a case, each row in the data table must have a unique triplet of values on the category, sum-by and data series columns. In a two-dimensional chart, the entire data series for a single category is displayed side-by-side in the XY plane. In a three-dimensional stack column chart, the data series is displayed along the Z-axis.

3.4.6.1. Data Mapping

In stack column charts, each category is further subdivided into components. Hence, there is an additional sum-by option used to determine the sum-by values for the chart.



Data Mapping

DataSeries : None □ Multi Selection

Category (X) : Drink □ Multi Selection

Sum by : Day □ Multi Selection

Value (Y) : volume

2nd Value : None

Combo: Line

2nd Series : drink □ Multi Selection

Mapping Options for Stack Column Charts

The data mapping options are as follows:

Data Series: Allows you to choose a data column whose distinct values will determine the number of data series in the chart. Each element in a data series is drawn using the same color.

Category (X): Allows you to choose a data column whose distinct values determine the categories. Values from this data column are used to calibrate the X-axis. Each category in a data series is drawn as a distinct column in the chart.

Sum-by: Allows you to choose a data column whose distinct values determine the components in each category. Each distinct value in this column determines a distinct stack component of a column.

Value (Y): Choose a data column to provide values for each category.

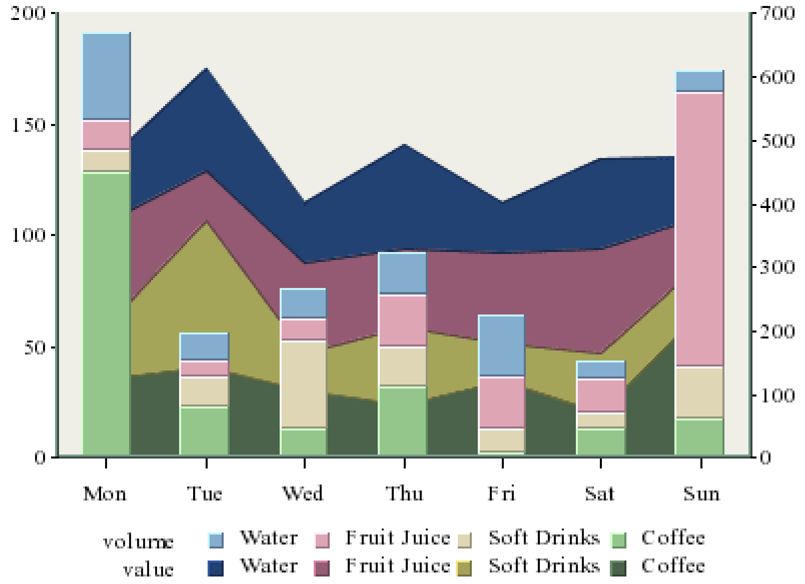
2nd value: Add a second value to create a combination chart.

2nd Series: Choose another column to be series for the secondary chart. This option is applicable only if the secondary chart is an overlay chart.

Combo: Choose the chart type for the secondary chart. For stack column charts the combo options are *Line*, *Stack Area*, and *Overlay*.

The data mapping also allows you to transpose the data (in other words: to select several columns for a single category). To learn more about data transposition, please see Section 3.4.1.1 - Data Transposition.

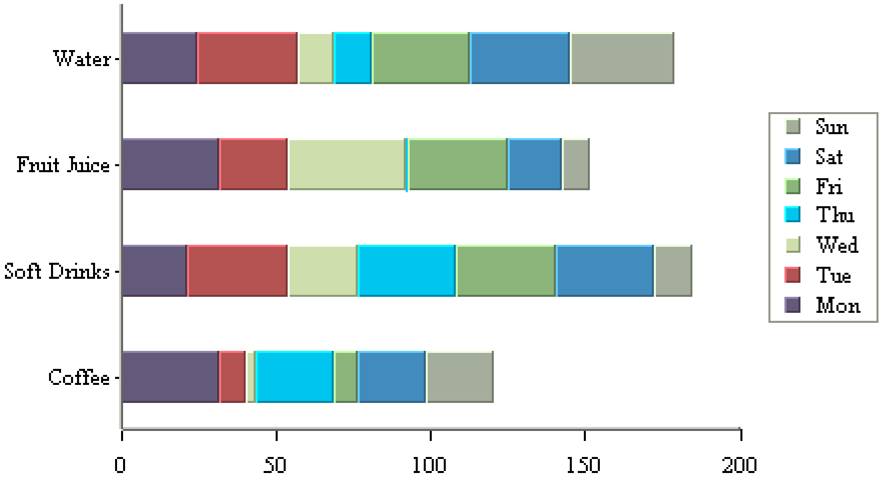
Stack column charts have a unique combination option which allows users to plot a second value as a stack area chart. Both values share the same category and sum-by values. This chart can compare component based categories for different values in the data.



Stack Column-Stack Area Combination Chart

With this combination chart you can specify to hide particular sum-by component in the chart to achieve a strip area effect.

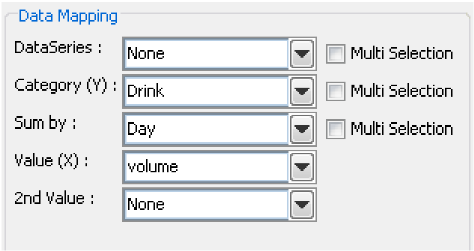
3.4.7. Stack Bar Charts



Stack Bar Chart

Like a stack column chart, the stack bar chart displays the chart categories as a sum of different stacked values - the sum-by column in the data source. The difference for a stack bar chart is that the categories are plotted on the Y-axis and the values are plotted along the X-axis.

3.4.7.1. Data Mapping



Data Mapping

DataSeries : None □ Multi Selection

Category (Y) : Drink □ Multi Selection

Sum by : Day □ Multi Selection

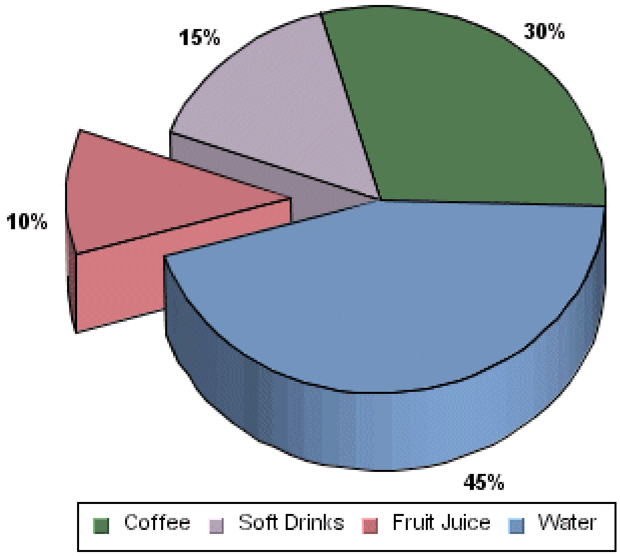
Value (X) : volume

2nd Value : None

Data Mapping Options for Stack Bar Charts

The mapping for this chart type is similar to that of a stack column chart, except the category (X) and values (Y) under column chart become category (Y) and values (X), respectively. This is because values are represented vertically in a column chart, but horizontally in a bar chart. Please note that the *2nd Series* and *Combo* options are not available for stack bar charts. This is because the only combination available with stack bar charts is a line.

3.4.8. Pie Charts

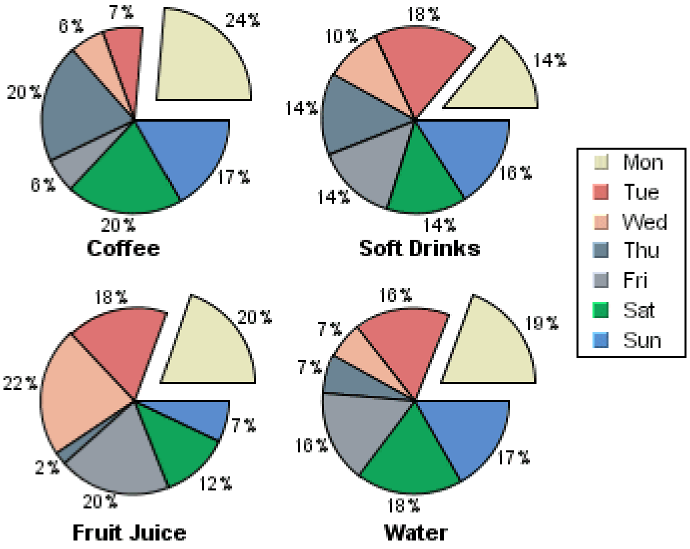


Pie Chart

In a pie chart, each row in the data is represented as a pie sector. The pie chart requires a category column and a value column. The value column must be numeric. The number of distinct values in the category column determines the number of pie sectors in the chart. All the values are summed up and each slice is assigned an angle according to its share in the total. Thus, the value column for each category determines the size of the slice representing that category.

A three-dimensional pie chart is simply an extension of its two-dimensional counterpart and contains no extra information.

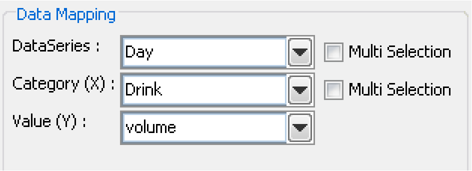
Pie charts can also have a data series. If you specify a series in the data mapping, a separate pie will be drawn for each category element and it will be comprised of the series elements. A pie chart with a data series is displayed below:



Pie Chart With Data Series

When a series is present, all the separate pies are drawn on a single plot area and resized in proportion with the plot. You can choose to either stack the different pies or draw them in a line.

3.4.8.1. Data Mapping


Data Mapping

DataSeries : Day ☐ Multi Selection

Category (X) : Drink ☐ Multi Selection

Value (Y) : volume

Mapping Options for Pie Charts

For pie charts, the mapping is as follows:

Data Series: Allows you to choose a data column whose distinct values will determine the number of data series in the chart. Each element in a data series is drawn using the same color.

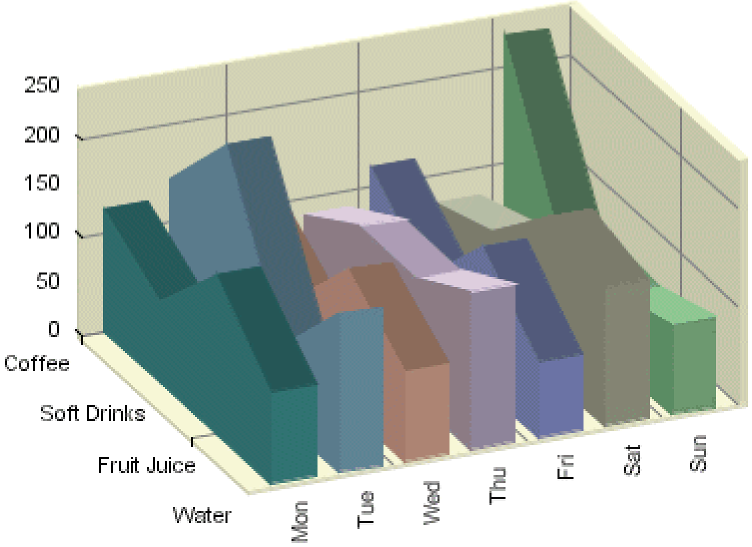
Category (X): Allows you to choose a data column whose distinct values determine the various categories.

Value (Y): Allows you to choose a data column to provide values for each category.

There is no *2nd Value* option, as you cannot make any combination charts with a pie chart.

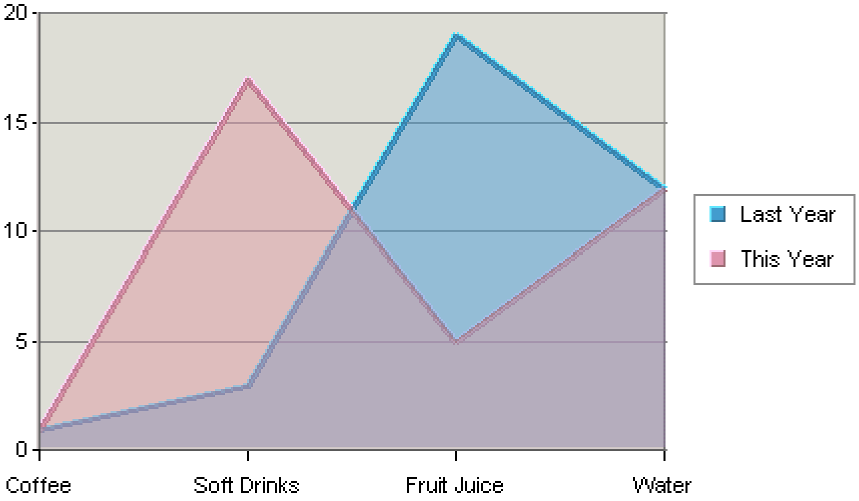
The data mapping also allows you to transpose the data (in other words: to select several columns for a single category). To learn more about data transposition, please see Section 3.4.1.1 - Data Transposition.

3.4.9. Area Charts



Three-Dimensional Area Chart

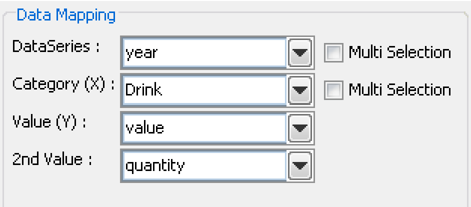
A three-dimensional area chart may be viewed as a derivative of a column chart. A three-dimensional area chart may be constructed from a three-dimensional column chart in the following manner. For each data series (Z-axis) value, the tops of all the columns are joined together by a thick line. The columns are then removed and the area (in the XY plane) under each line is filled with a distinct color.



Two-Dimensional Area Chart

A two-dimensional area chart is essentially a projection of its three-dimensional counterpart viewed along the Z-axis. As a result, a two-dimensional area chart may sometimes hide a data series value altogether. Therefore, it must be used with caution. The chart above illustrates this point: in a two-dimensional display some parts of each series are concealed by a series in front. To ameliorate this problem, you can change the ordering of the data series (see Section 3.5.8.3 - Data Ordering) or set the area translucent (see Section 3.5.1.3 - Format Menu) as in the example above.

3.4.9.1. Data Mapping



Data Mapping

DataSeries : year ☐ Multi Selection

Category (X) : Drink ☐ Multi Selection

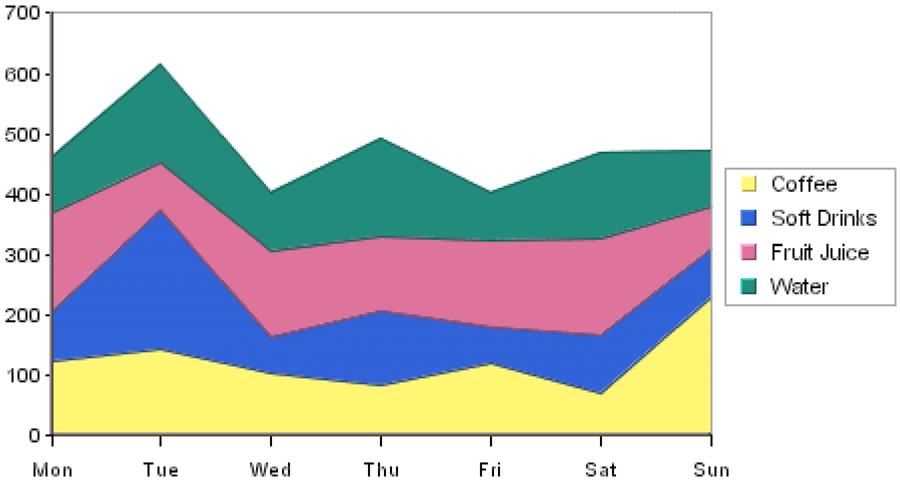
Value (Y) : value

2nd Value : quantity

Mapping Options for Area Charts

Data mapping for area charts is almost exactly the same as for column charts (discussed in Section 3.4.2.1 - Data Mapping). However, area charts do not have the *2nd Series* and *Combo* options. This is because the only combination available with area charts is a line.

3.4.10. Stack Area Charts



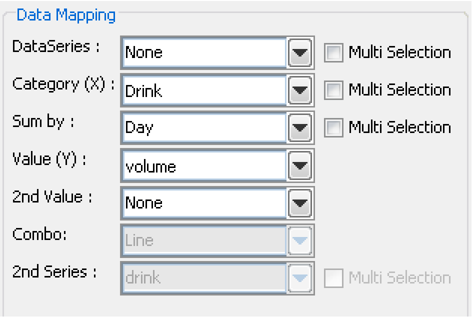
Stack Area Chart

A two-dimensional stack area chart may be viewed as a derivative of a two-dimensional stack column chart but without a data series. This chart may be constructed from a stack column chart as follows: The tops of each stack component that have the same color are joined together by lines. The stack columns are removed and the area between each set of lines is filled with a distinct color.

A three-dimensional stack area chart can have a data series and can also be derived from a three-dimensional stack column chart using the process stated above. In this case, the stack columns for each distinct data series value (a fixed Z-axis value) are joined separately, giving rise to multiple stacks.

Both two-dimensional and three-dimensional stack area charts may be constructed using a similar set of data as needed for their respective stack column chart counterparts. Note, as stated earlier, that a two-dimensional stack area chart cannot have a data series.

3.4.10.1. Data Mapping



Data Mapping

DataSeries : None □ Multi Selection

Category (X) : Drink □ Multi Selection

Sum by : Day □ Multi Selection

Value (Y) : volume

2nd Value : None

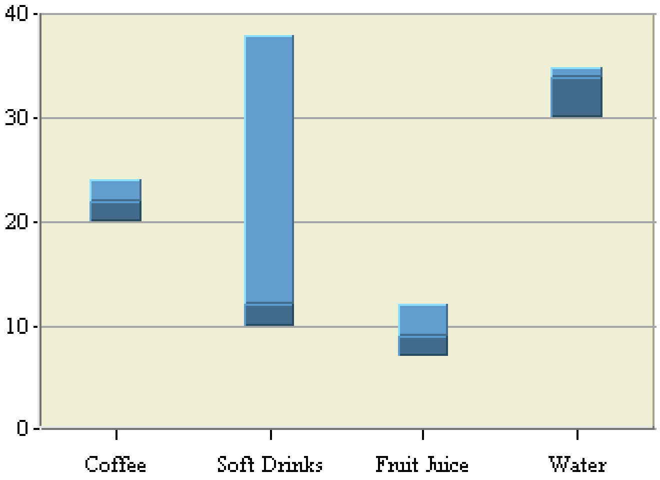
Combo: Line

2nd Series : drink □ Multi Selection

Mapping Options for Stack Area Charts

Data Mapping for stack area charts is almost exactly the same as for stack column charts (covered in Section 3.4.6.1 - Data Mapping). However, two-dimensional stack charts cannot have a data series, and the combo options are *Line* and *Overlay*.

3.4.11. High-Low Charts

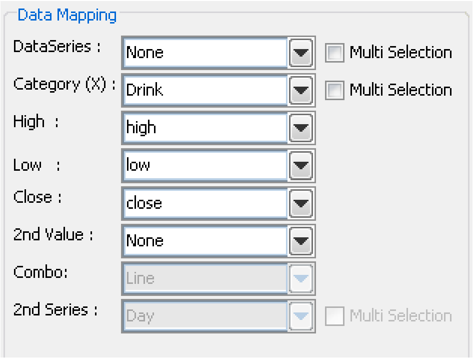


High-Low Chart

A high-low chart is also a derivative of a column chart, only that instead of one value column it uses two columns for high and low bounds of values. The data for the chart must contain at least three columns: category, high, and low. The category column can contain values of any type, while the high and low columns must be numeric.

Another option for a high-low chart is a data column called the *close* column. If a close column is also used, then the close value will lie somewhere between the high and low values. The portion of the bar between the low point and the lose point is rendered in a darker form of the same color as the portion between the close point and the high point. As for many other types of charts, a high-low chart may include a data series column.

3.4.11.1. Data Mapping



DataSeries : None ☐ Multi Selection

Category (X) : Drink ☐ Multi Selection

High : high

Low : low

Close : close

2nd Value : None

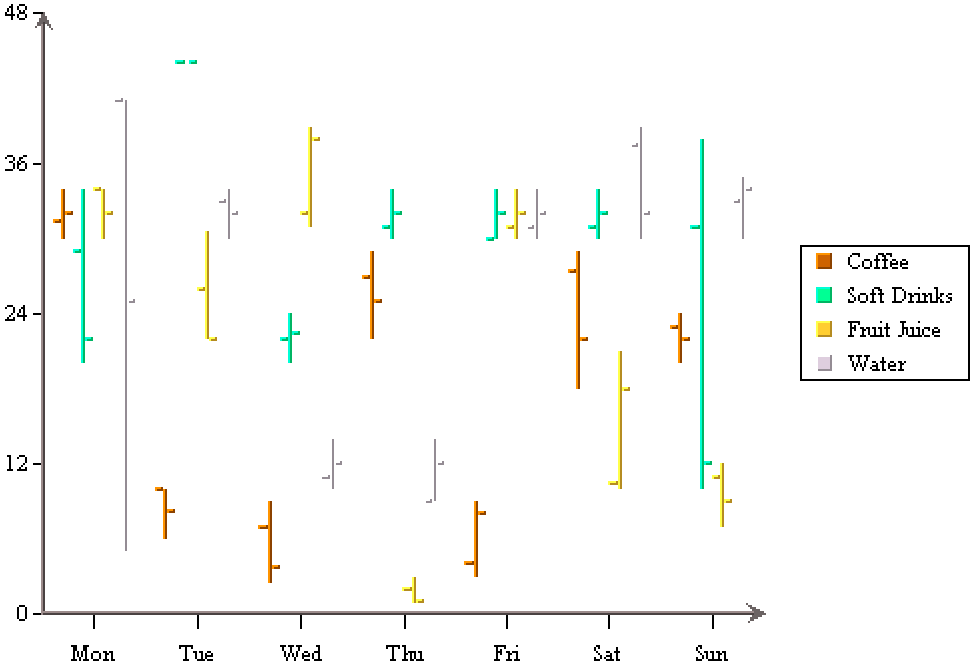
Combo: Line

2nd Series : Day ☐ Multi Selection

Mapping Options for High-Low Charts

Data mapping for high-low charts is similar to column charts. You can select series and category columns in the same manner. The difference for high-low charts is that you need to specify at least two value columns - a high and a low value. These are the two points that are plotted for each category. A third value called *Close* can also be specified. Combo options for high-low charts are *Line*, *Column*, and *Overlay*.

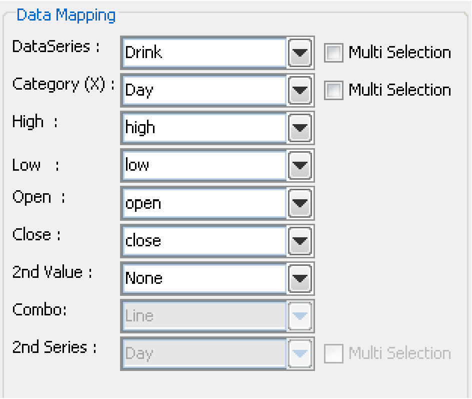
3.4.12. HLCO Charts



HLCO Chart

A HLCO (High Low Close Open) chart is similar to the high-low chart described above, except that it also contains an *open* column. It is useful in presenting data that fluctuates over discrete periods of time, such as stock prices or inter-day temperatures. The HLCO chart can also be displayed in a Candlestick format for both 2D and 3D charts. This feature is described in Section 3.5 - The Designer Interface.

3.4.12.1. Data Mapping



Data Mapping

DataSeries : Drink □ Multi Selection

Category (X) : Day □ Multi Selection

High : high

Low : low

Open : open

Close : close

2nd Value : None

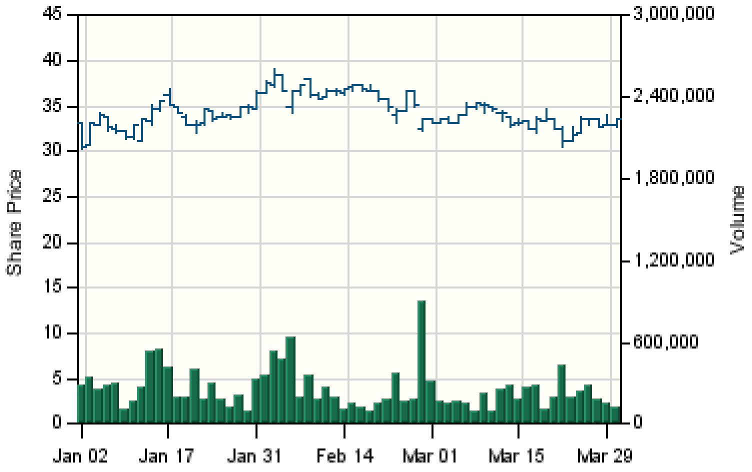
Combo: Line

2nd Series : Day □ Multi Selection

Mapping Options for HLCO Charts

Data mapping for HLCO charts is similar to that for High-Low charts. The only difference is that instead of specifying two different value columns, you must specify four different columns - *High*, *Low*, *Open* and *Close*. Combo options are *Line*, *Column* and *Overlay*.

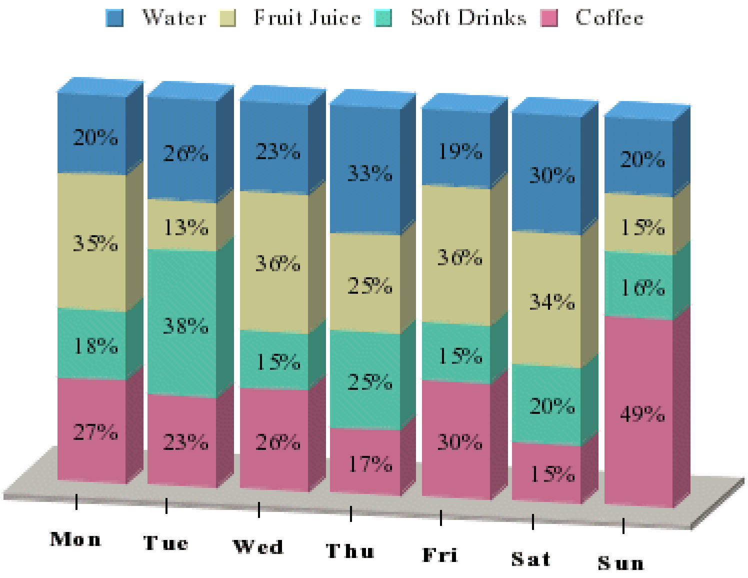
A common kind of chart is a one that plots both stock price and trading volume in the same chart. This can be accomplished using an HLCO-column combination in ChartDesigner.



HLCO-Column Combination Chart

This example plots stock price and volume data over a three month period.

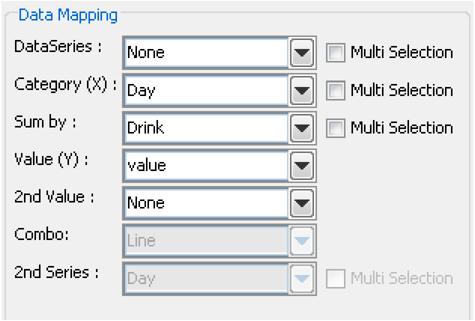
3.4.13. Percentage Column Charts



Percentage Column Chart

A percentage column chart may be viewed as a derivative of pie and column charts together. Each column in the chart corresponds to one pie. A percentage column chart has a category column corresponding to the X-axis value. A sum-by column represents the category in this case and the value column is the same as that of a pie chart.

3.4.13.1. Data Mapping



Data Mapping

DataSeries : None □ Multi Selection

Category (X) : Day □ Multi Selection

Sum by : Drink □ Multi Selection

Value (Y) : value

2nd Value : None

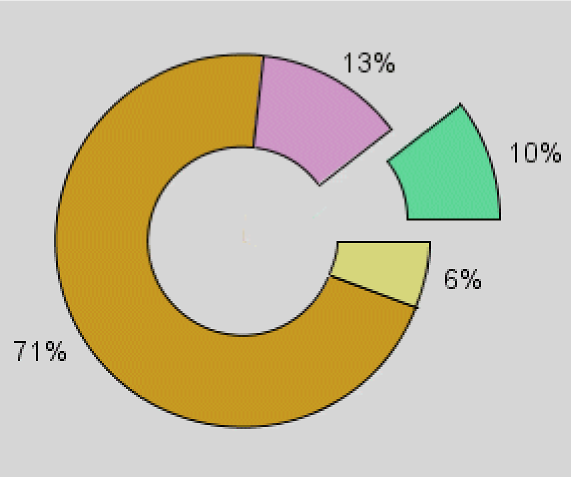
Combo: Line

2nd Series : Day □ Multi Selection

Mapping Options for Percentage Column Charts

Data mapping for percentage column charts is the same as for stack column charts (covered in Section 3.4.6.1 - Data Mapping). The only difference is that the sum-by components are plotted as a percentage of the Value column instead of discrete values. Combo options for percentage column charts are *Line* and *Overlay*.

3.4.14. Doughnut Charts

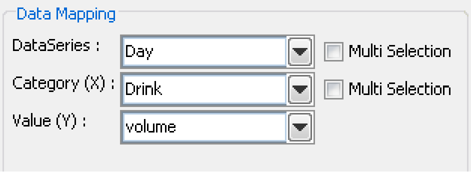


Doughnut Chart

A doughnut chart is the same as a pie chart, except for the “hole” in the center.

Just as for a pie chart, if a data series is selected, a separate doughnut will be drawn for each category element and each doughnut will be comprised of the series elements. When a series is present, all the separate doughnuts are drawn on a single plot area and resized in proportion with the plot. You have an option to either stack the different doughnuts or draw them in a line.

3.4.14.1. Data Mapping



Data Mapping

DataSeries : Day ☐ Multi Selection

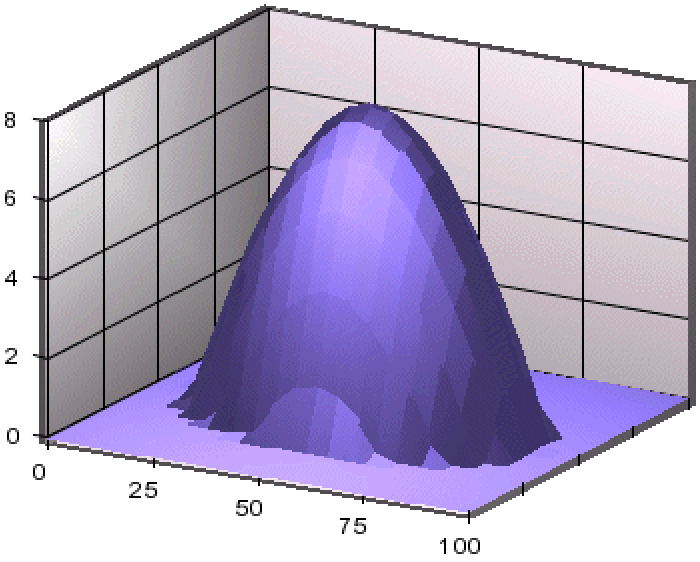
Category (X) : Drink ☐ Multi Selection

Value (Y) : volume

Mapping Options for Doughnut Charts

Data mapping for doughnut charts is exactly the same as for pie charts (discussed in section Section 3.4.8.1 - Data Mapping).

3.4.15. Surface Charts



Surface Chart

A 3D surface chart is a scientific/business chart, which uses three sets of numeric data values to form a smooth surface in a three dimensional space. For each data point, two of the three values represent the coordinates on the horizontal plane and the third value is used to determine the vertical position of a data point. Input data can be in a form of a rectangular matrix (as shown below) or arranged in three columns where each column represents an axis. In the following sample data, the top row (column header) and the left column (row header) represent the coordinate values of the axes that form the plane on which the surface will be drawn. When the chart is viewed from above the plane, you can see a grid formed by joining every four adjacent points together. A given set of data cannot plot a surface chart if this grid cannot be drawn.

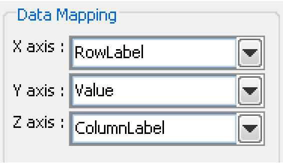
Sample surface chart data:

	0	20	40	60	70	80	100
20	0	0	0	0	0	0	0
30	0	10	10	10	10	10	10
40	0	10	25	25	25	2	10
80	0	10	25	30	30	30	25
90	0	10	25	30	30	30	25
100	0	10	10	25	25	25	10

The values of the column header and the row header are not required to be equally separated because a surface chart can properly scale the horizontal distances. This dataset will create a distorted inverted cone.

A surface chart uses a set of X, Y, and Z coordinates to plot a 3D chart. The vertical axis is called Y-axis and each Y value is represented by a function of X and Z, f(X, Z). Data on XY plane (horizontal plane) simply looks like a square matrix. Surface charts do not support data series and cannot be converted to a 2D chart.

3.4.15.1. Data Mapping



Data Mapping

X axis : RowLabel

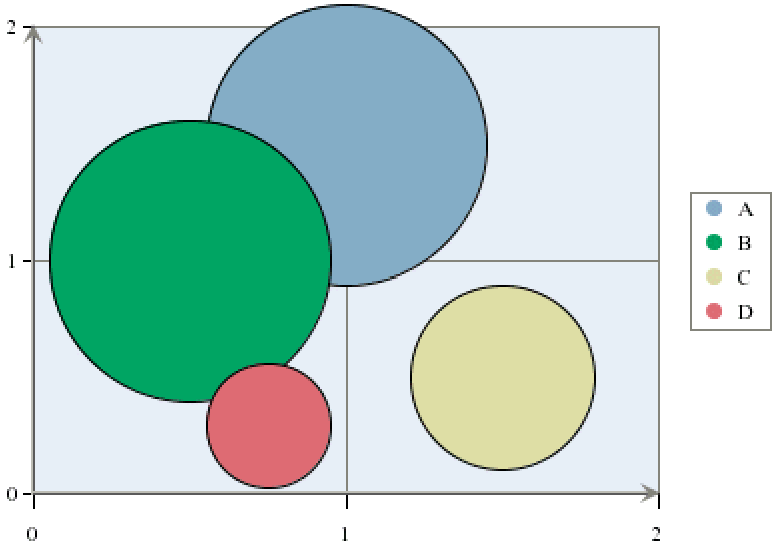
Y axis : Value

Z axis : ColumnLabel

Mapping Options for Surface Charts

The data mapping for a surface chart is similar to a three-dimensional scatter chart; the X-axis, Y-axis, and Z-axis values determine the X, Y, and Z coordinates of a point respectively. However, unlike a scatter chart, surface charts do not support data series and cannot be converted to a two-dimensional chart.

3.4.16. Bubble Charts



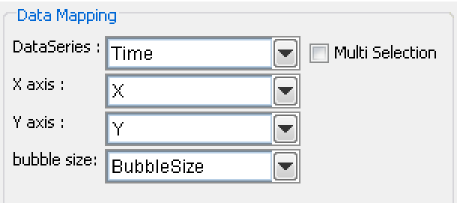
Bubble Chart

A bubble chart is used to represent data wherein the size of the bubble also provides information. A data point is represented by an XY coordinate and a third point, which is the radius of a circle or bubble.

This chart is available in a two-dimensional form only. For more information about the bubble charts and their display options, please see Section 3.5.9.1 - Bubble Charts.

3.4.16.1. Data Mapping

The data mapping for bubble charts is similar to three-dimensional scatter charts; however, instead of plotting a Z-axis position, the third value determines the size of the bubble (specifically the radius).



Data Mapping

DataSeries : Time □ Multi Selection

X axis : X

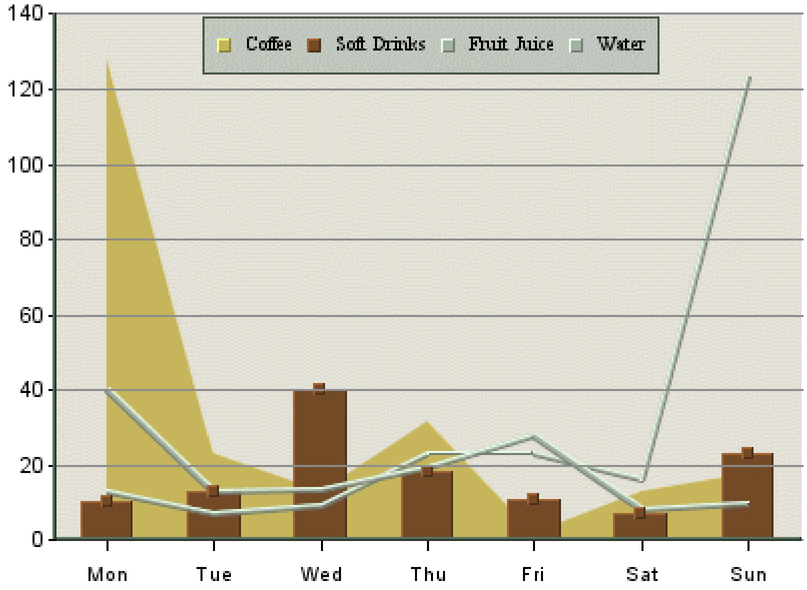
Y axis : Y

bubble size: BubbleSize

Mapping Options for Bubble Charts

Note that similarly to a scatter chart, the columns have to be numeric in order to be mapped successfully.

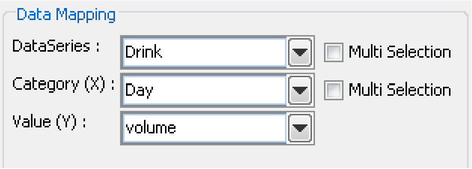
3.4.17. Overlay Charts



Overlay Chart

ChartDesigner supports a special chart type called overlay chart, which allows you to super-impose more than two charts with a common category axis. A different chart can be used to represent each element of a data series. This allows for more freedom while creating the chart and also allows more information to be represented. The chart types supported in an overlay chart are column, area, and line charts. Only two-dimensional charts can be included in an overlay chart.

3.4.17.1. Data Mapping


Data Mapping

DataSeries : Drink □ Multi Selection

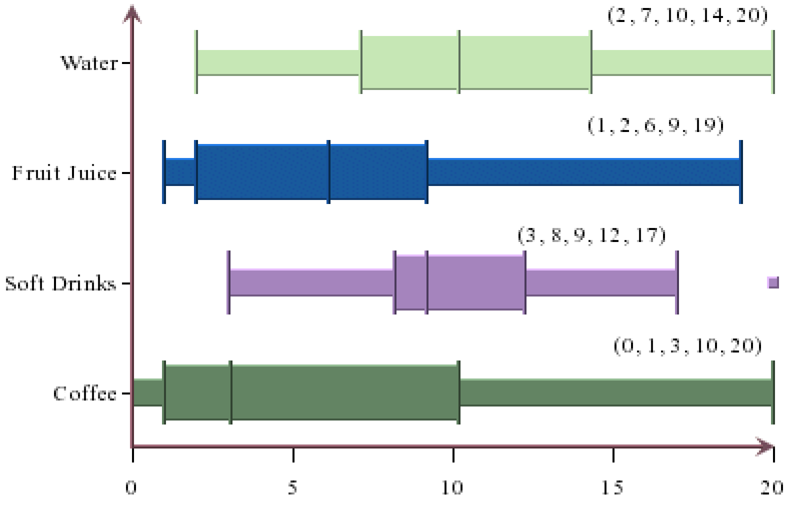
Category (X) : Day □ Multi Selection

Value (Y) : volume

Mapping Options for Overlay Charts

The data mapping for an overlay chart is similar to a column chart (covered in Section 3.4.2.1 - Data Mapping), except that the overlay chart plots each element in the data series as a separate chart. Please note that the *2nd Value*, *2nd Series* and *Combo* options do not apply to overlay charts. Overlay charts do not support secondary values. However, the different series elements can be plotted on different axes. For more about plotting options for overlay charts, see Section 3.5.9.3 - Overlay Charts.

3.4.18. Box Charts

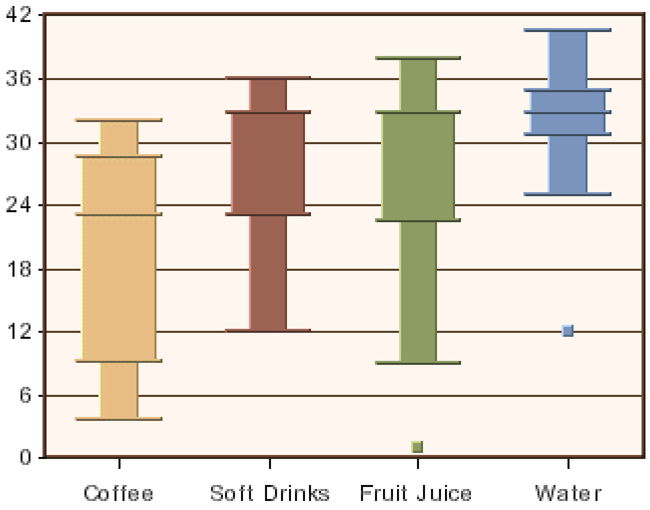


Box Chart

ChartDesigner provides a statistical chart type, the Box Chart, also known as Box and Whiskers Chart. Basically, the box chart provides a quick visual realization of summary statistics. A histogram shows the distributions of observed values so you can identify the peak(s), minimum, maximum values and clustering of data. A box chart, on the other hand, displays a summary of data distribution in quarterly percentiles (Minimum data point, 25th percentile, median, 75th percentile, and Maximum data point).

The middle line in the box represents the median. The other lines in each direction represent the 25th percentile increment (decrement). The minimum and maximum points are the observed minimum and maximum which are not outliers. The lines that extend from the edge of the box to the minimum/maximum are sometimes called whiskers. Hence, the name box and whisker chart. Outliers are any values that are 1.5 times (or more) the length of the box, that being the difference between the 75th and 25th percentiles. Outliers are calculated and shown as dots away from the box. The other nice feature about box plot is that you can stack up the boxes in one chart to compare summaries of different data sets.

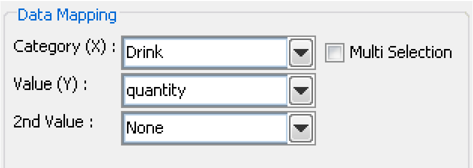
ChartDesigner allows box charts to be displayed vertically in addition to the default horizontal setting. To use this feature in Chart Designer, create a box chart and then select Format → Chart Options, and select *Vertical*.



Vertical Box Chart

This chart is available in two-dimensional form only.

3.4.18.1. Data Mapping



Data Mapping

Category (X) : Drink ▾ ☐ Multi Selection

Value (Y) : quantity ▾

2nd Value : None ▾

Mapping Options for Box Charts

For box charts the mapping is as follows:

Category (X): Allows you to choose a data column whose distinct values determine the various categories.

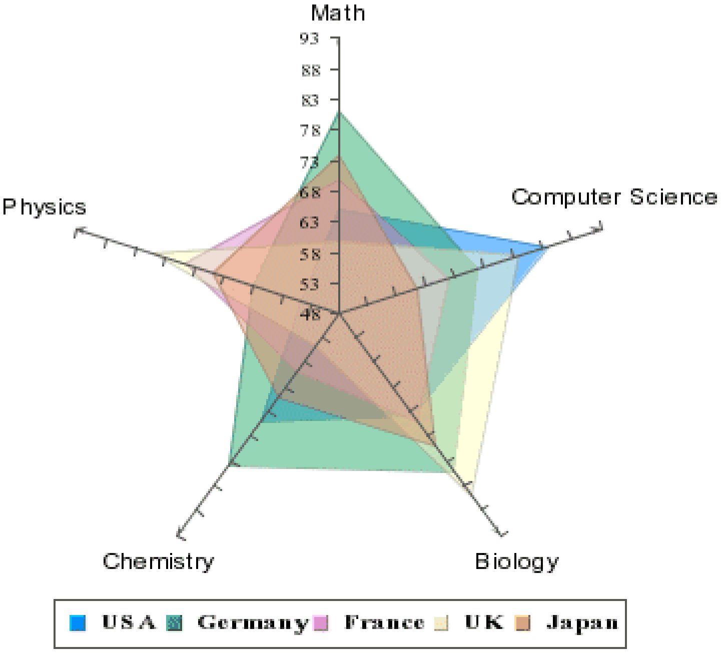
Value (Y): Allows you to choose a data column to provide values for each category. These values are used to calculate the minimum data point, the 25th percentile, the median, the 75th percentile, and the maximum data point.

2nd value: Add a second value to create a combination chart.

Please note that the *2nd Series* and *Combo* options are not available for box charts. This is because the only combination available with box charts is a line.

The data mapping also allows you to transpose the data (in other words: to select several columns for a single category). To learn more about data transposition, please see Section 3.4.1.1 - Data Transposition.

3.4.19. Radar Charts

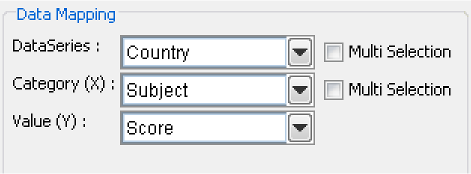


Radar Chart

Radar charts are useful when you need to compare performance/measurement results, statistics, etc from different sources. For example, we can compare median test scores of children in the industrial nations on subjects such as math, computer science, and biology. A radar chart has multiple axes along which data can be plotted. Each axis is a category. The data is shown as points on the axis. The points belonging to one data series can be either joined or the enclosed area filled. A point close to the center on any axis indicates a low value while a point near the end represents a high value. In some cases, low values may be more desirable than high values, e.g. price/earning, price/sales, price/book of stocks.

This chart is available in a two-dimensional form only.

3.4.19.1. Data Mapping



Data Mapping

DataSeries : Country □ Multi Selection

Category (X) : Subject □ Multi Selection

Value (Y) : Score

Mapping Options for Radar Charts

For radar charts the mapping is as follows:

Data Series: Allows you to choose a data column whose distinct values will determine the number of data series in the chart. Each element in a data series is drawn using the same color along each axis.

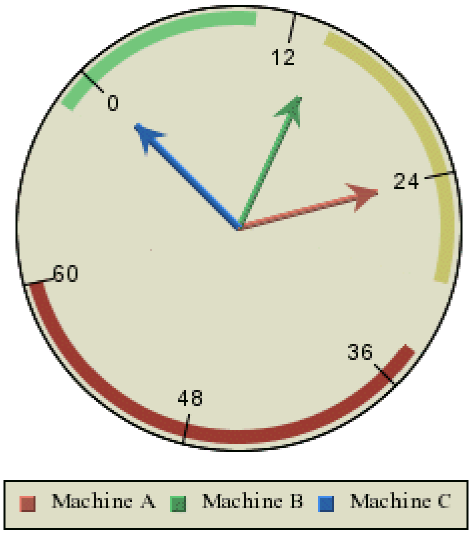
Category (X): Allows you to choose a data column whose distinct values determine the categories. The number of unique elements in this column determines the number of axes in the radar chart.

Value (Y): Allows you to choose a data column to provide the numeric value to plot against the category.

Please note that the *2nd Value*, *2nd Series*, and *Combo* options do not apply for radar charts. Radar charts do not support secondary values.

The data mapping also allows you to transpose the data (in other words: to select several columns for a single category). To learn more about data transposition, please see Section 3.4.1.1 - Data Transposition.

3.4.20. Dial Charts

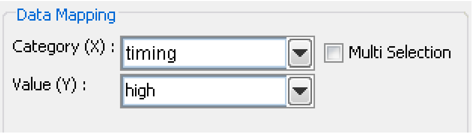


Dial Chart

Dial Charts are used to build circular charts (such as temperature gauges, speedometers, and clocks), create dashboards, or balanced scorecard applications. Here the data is represented as a “hand” on a “dial”. You can either use the whole dial (for instance for a clock) or just part of it (a semi- circular chart such as a gasoline gauge). Dial Charts support pop-up labels, drill-down, and user interactions such as rotating the hands or sectors of a dial.

This chart type is available in a two-dimensional form only.

3.4.20.1. Data Mapping



Data Mapping

Category (X) : timing

Value (Y) : high

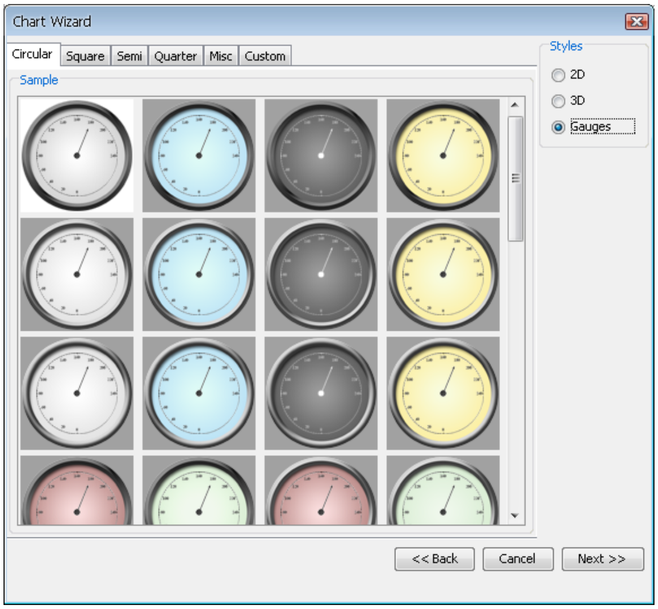
□ Multi Selection

Mapping Options for Dial Charts

The data mapping of a dial chart is similar to a pie chart (detailed in Section 3.4.8.1 - Data Mapping). However, instead of pie wedges, the categories become dial hands. Also, dial charts do not support data series or secondary values.

3.4.20.2. Gauges

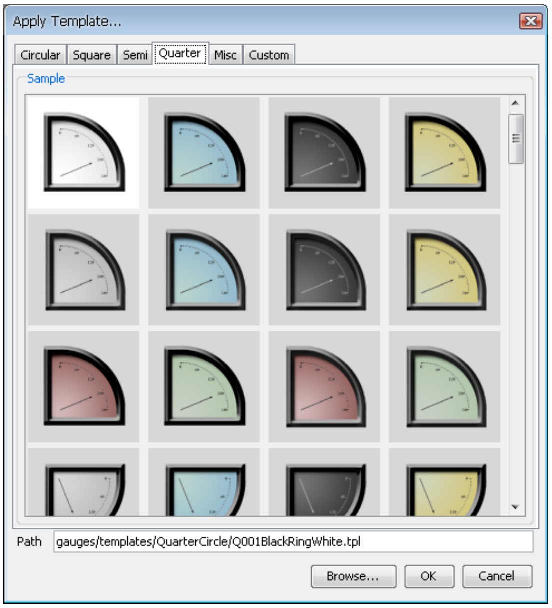
Gauges are special types of dial charts containing a plot background image and/or plot foreground image. To create a new gauge, select the *Gauges* radio button on the *Chart Type Selection* dialog.



Selecting a Gauge Template

You can also create your own gauge by creating a template, saving the tpl file in the `<EspressReport Install>/gauges/templates/Custom/` folder (please note that the `Custom` folder is not created automatically by the EspressReport installator so you will have to create it manually in your favorite file manager). To add this template to the Custom tab, you will also need to provide two images, a screenshot of the template placed in `<EspressReport Install>/gauges/screenshots/selected/Custom/`, and a dimmer version of the same template placed in `<EspressReport Install>/gauges/screenshots/unselected/Custom/`. An easy way to create the screenshot is to export the template to gif for the regular screenshot and resize it to 100px by 100px. Then change the background to a darker color, export and resize again for the dimmer version.

It is also possible to apply a gauge template onto an existing dial chart. Selecting apply template when the current chart is a dial chart will display the gauge tabs just like when creating a new chart.

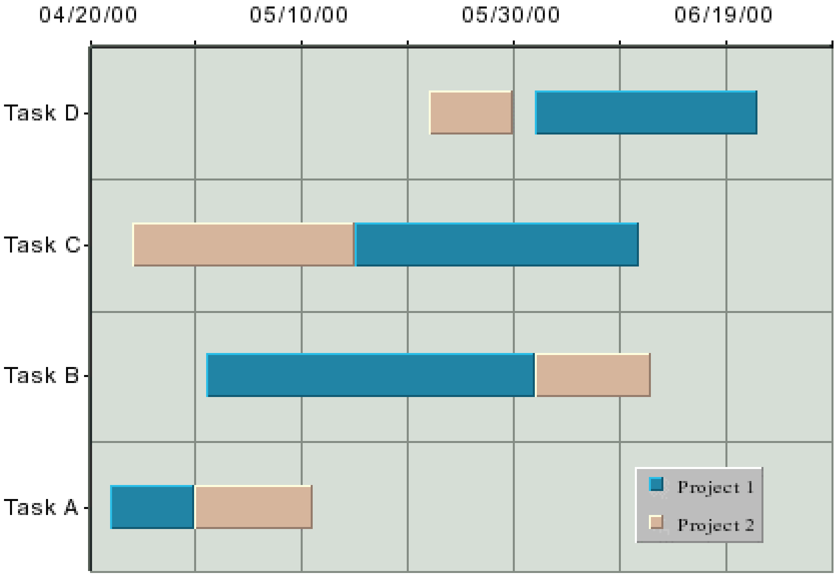


Path gauges/templates/QuarterCircle/Q001BlackRingWhite.tpl

Applying a Gauge Template

For more information regarding the dial background and foreground images, see Section 3.5.9.2 - Dial Charts.

3.4.21. Gantt Charts



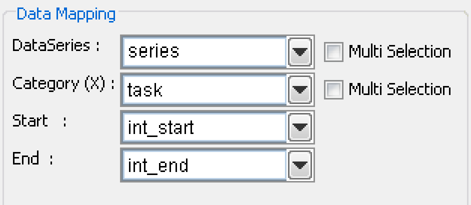
Gantt Chart

A Gantt or time chart is used to represent data where a time factor is being measured. This is often used in project or time management situations. A Gantt chart resembles a bar chart with the category axis on the Y-axis and the value axis on the X-axis. The value axis uses date, time, or timestamp data (numeric values can also be used). Each data point has a start and end time associated with it.

This chart type is available in a two-dimensional form only.

3.4.21.1. Data Mapping

The data mapping for Gantt charts is similar to high-low charts. However, instead of selecting a column for the high and low values, you select the start and end values.



Data Mapping

DataSeries : series □ Multi Selection

Category (X) : task □ Multi Selection

Start : int_start

End : int_end

Mapping Options for Gantt Charts

The mapping is as follows:

Data Series: Allows you to choose a data column whose distinct values will determine the number of data series in the chart.

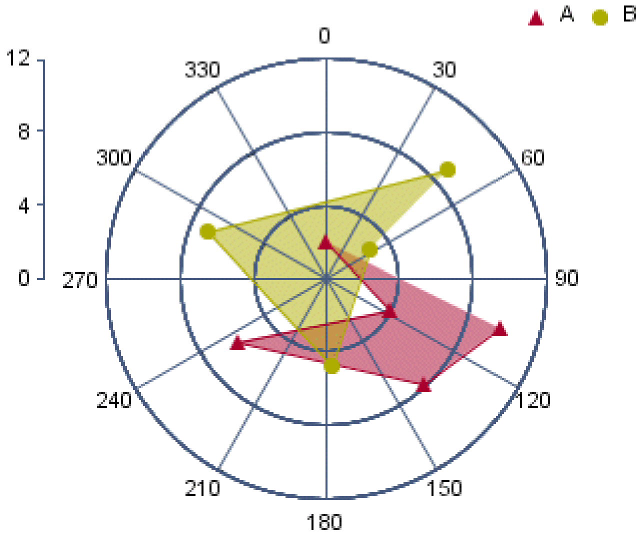
Category (X): Allows you to choose a data column whose distinct values will determine the categories.

Start: Allows you to choose a data column whose distinct values represent the starting time interval for the category.

End: Allows you to choose a data column whose distinct values represent the ending time interval for the category.

The data mapping also allows you to transpose the data (in other words: to select several columns for a single category). To learn more about data transposition, please see Section 3.4.1.1 - Data Transposition.

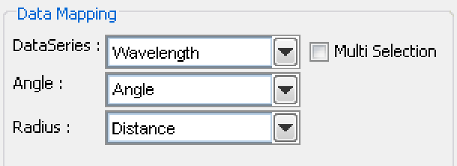
3.4.22. Polar Charts



Polar Chart

Like a two-dimensional scatter chart, a polar chart plots points on a plane. However, instead of rectangular coordinates, a polar chart plots points using polar coordinates (r, θ), where r represents the distance from the center of the circular plot and θ represents the angle of a ray emanating from the center of the plot and passing through the point. Like scatter charts, the data for polar chart coordinates must be numeric. The θ values can be supplied in degrees or radians. A third data series column can be used to separate the data points into groups.

3.4.22.1. Data Mapping



Data Mapping

DataSeries : Wavelength □ Multi Selection

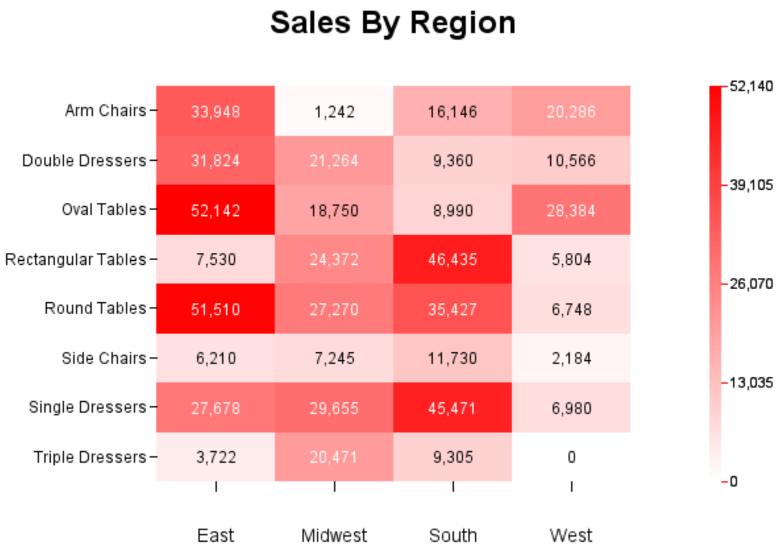
Angle : Angle

Radius : Distance

Mapping Options for Polar Charts

The data mapping for polar charts is similar to scatter charts. The angle and radius options allow you to select the columns whose values will make up the polar coordinates. These must both be numeric. The data series box allows you to choose a data column whose distinct values will determine the number of data series in the chart. Each element in a data series is drawn using the same set of drawing attributes, e.g., colors and markers.

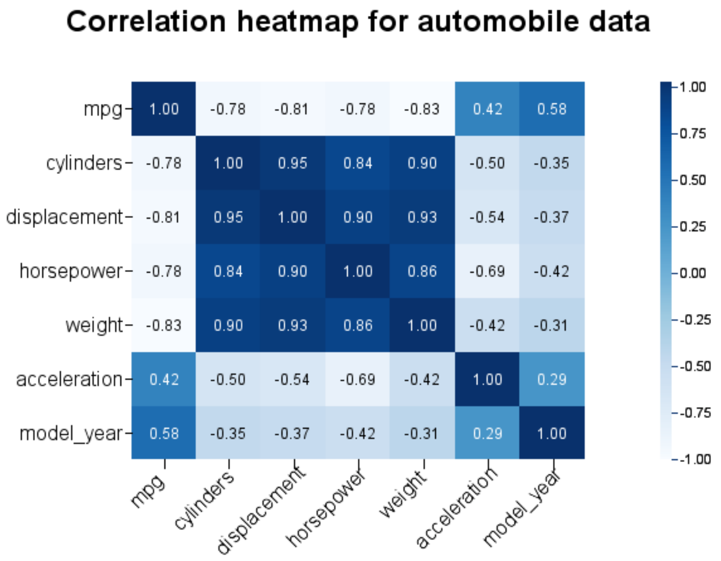
3.4.23. Heatmap Charts



Heatmap Chart

A heatmap is a data visualization tool that uses color to represent the values of a matrix. This method is particularly effective for displaying large amounts of data in a way that is easy to understand at a glance. The colors in a heatmap typically range from light (white) to dark (e.g.,red) along the color gradient, or from cool (e.g., blue) to warm (e.g., red), with each color representing a different value. This visual representation allows for quick identification of patterns, trends, and outliers within the data. A simple example that shows the revenue of certain product category in certain region is presented above. In this case, region is the X axis, product category is the Y axis and the plot is (X,Y, value).

In addition to just plotting (X,Y. value) straight from the input data, there is a correlation option that you can apply to the data. As such, you can easily visualize the correlation coefficients between pairs of columns from the data source, i.e. each entry in the matrix is the correlation coefficient between the corresponding columns on the X axis and the Y axis. Below is an example. In this example, the columns along the X axis are the same as those along the Y axis. But they do not have to be the same.



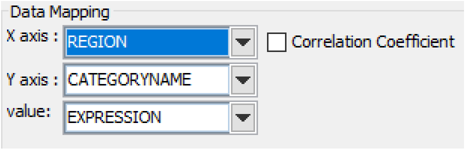
Correlation Heatmap Chart

Note

Heatmap chart is available in a two-dimensional form only.

3.4.23.1. Data Mapping

Data mapping of heatmap without applying the correlation function is straightforward.



Data Mapping

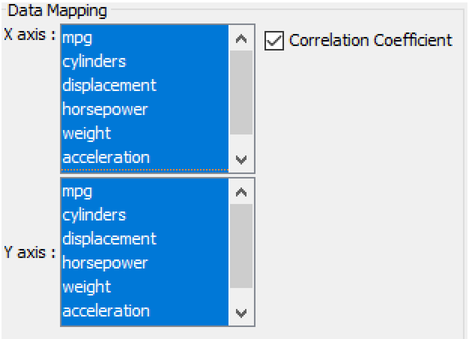
X axis : REGION □ Correlation Coefficient

Y axis : CATEGORYNAME

value: EXPRESSION

Heatmap data mapping

Data mapping of heatmap with correlation function applied.



Data Mapping

X axis : mpg, cylinders, displacement, horsepower, weight, acceleration ☑ Correlation Coefficient

Y axis : mpg, cylinders, displacement, horsepower, weight, acceleration

Correlation heatmap data mapping

Note that the “Correlation Coefficient” checkbox is checked, and the X-axis and Y-axis show all the available numeric columns in the data source. You can select multiple columns in each of the axes.

In our earlier example, we selected all the columns for the X axis and all the columns for the Y axis. But you can choose different columns for each axis.

3.4.24. Changing Data Mapping or Data Source

Once you have finished setting the mapping options you will be taken to the main Chart Designer interface. Once you are in the Chart Designer you can return to the Data Mapping window by selecting Data → Modify Data Mapping, or by clicking the *Change data mapping* icon. This will return you to the data mapping screen, allowing you to adjust the mapping for the chart.

You can also change a data source directly from the Chart Designer by selecting Data → Modify Data Source, or by clicking the *Modify data source* icon on the toolbar. This will open the Data Source Manager.

If your chart uses a different data source than the report, you can also change the charts data source from the Chart Designer interface. Once you are in the Chart Designer, select *Modify Data Source* from the *Data* menu. This will return you to the Data Source Manager window and allow you to pick a new data source for the chart. Note that this option is not available if you select to use the report data as the data source. Also, if you change your data source, you will need to perform data mapping again.

3.5. The Designer Interface

Once you have completed all the steps in the Wizard, the main Chart Designer interface will appear. Here you can customize and modify the appearance of the chart by changing properties and adding new elements.

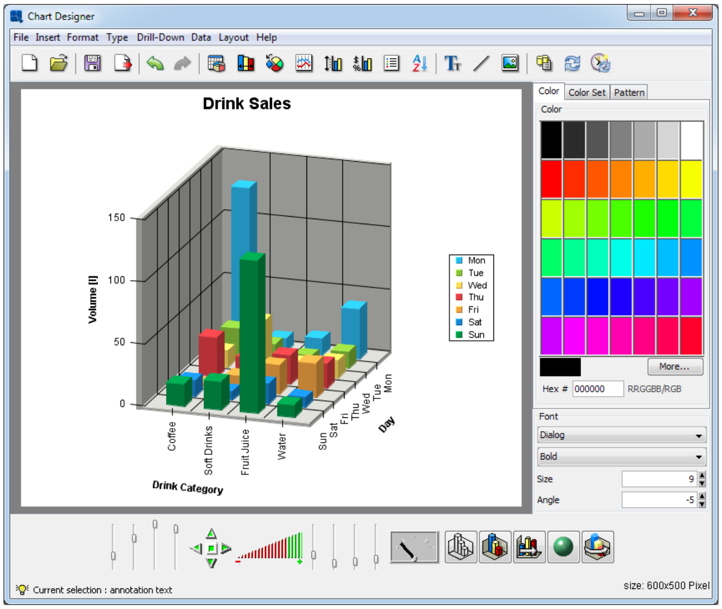


Chart Designer Interface

3.5.1. The Designer Menus

Most of Chart Designer's functions can be controlled from the menu bar at the top of the designer window. This section provides a brief overview of the available options. All of the features here are discussed later in this chapter.

3.5.1.1. File Menu

This menu performs basic file operations such as opening, closing, and saving files. Note that saving and opening options are not available when you use the report data as the data source for an embedded chart. For more information about file options, please see Section 3.7 - Saving & Exporting Charts.

New: This will return you to the Data Source Manager to begin creating a new chart. If you have not saved your current chart, you will be prompted to do so.

Open: This allows you to open a saved chart definition. Files that can be loaded by Chart Designer include `.cht`, `.tpl`, and `.xml` chart definition files.

Close: This closes the current chart.

Apply Template: This option allows you to apply a template to the current chart. You can apply any chart template in either .tpl or .xml format. For more information about working with chart templates, please see Section 3.7.2.1 - Working with Templates.

Save: This saves the current chart.

Save As: This allows you to save the current chart. You can save the chart as a `.cht, .pac` or `.tpl` (a chart, a complete packed chart or a template file). The

```
Create XML
```

check-box allows you to create an XML chart definition file.

Export: This allows you to export the chart to a number of static image formats. The chart data can also be exported as XML file.

Exit: This closes the Chart Designer with the possibility to save the current file.

3.5.1.2. Insert Menu:

This menu allows you to add various elements to a chart.

Titles: This option allows you to automatically add titles to the chart. A main title can be specified as well as titles for each of the axes. Unlike annotation text, titles will size and position automatically with the chart.

Text: This allows you to add text or annotation to a chart. Text can be added anywhere on the chart and can have a number of different formatting properties. Variables can be added to the text for run-time substitution. For more on adding text to charts, see Section 3.5.6.1 - Adding Text

Background: This allows you to select an image to use as the chart background. Background images can be tiled, centered, or stretched to fit the entire canvas.

Dial Foreground: This option is only available for dial charts. This allows you to select an image to use as the dial chart foreground. The image can be stretched.

Dial Background: This option is only available for dial charts. This allows you to select an image to use as the dial chart background. The image can be stretched.

Link: This allows you to add a hyperlink to a data point or a collection of data points in a chart. To use hyperlinks in a chart, you will need to export a map file along with the image.

Line: This allows you to add an arbitrary floating line to a chart. These lines can be placed anywhere and can be used to draw enclosed shapes as well. Floating lines are often used as pointers and can be generated with arrowheads.

TrendLine: This allows you to add a trend line to the chart. ChartDesigner allows you to draw many different types of trend lines including: linear, a polynomial of any degree, a power, exponential, logarithmic, a moving average, exponential moving average, triangular moving average, cubic B-spline, and a normal distribution curve.

Horz/Vert Line: This allows you to add a fixed horizontal or vertical line to a chart. You have the choice of either adding a constant line (one that draws at a fixed value on the X or Y-axis) or a control line (draws lines based on a certain value range either average, minimum, maximum, or multiples of standard deviation).

Control Area: This allows you to draw a fixed area (either on the plot area of a 2D chart or on the face of a dial chart) to compare against the data values of the chart.

3.5.1.3. Format Menu

This menu allows you to edit and modify the properties of many different chart components.

Undo: Cancels the last operation performed in the designer and reverts back to the previous state. The designer will remember the last 10 actions made.

Redo: Reverses the action of the undo command. For example, if you change the font color from black to red, you can undo this command to change it back to black, and then redo this command to have it change to red again.

Data Properties: This option allows you to control several options for how the data is displayed. You can control the thickness of columns/bars, how null data is displayed, whether to draw data top labels or not, as well as enable and select a color for negative top labels.

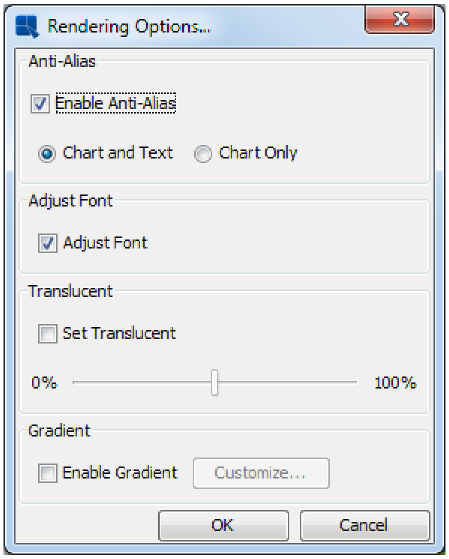
Histogram Options: This allows you to specify if you want to draw the chart as a histogram and display additional options to specify the frequency count.

Aggregation Options: This allows you to specify whether to aggregate the data before drawing the chart and display additional options to specify type of aggregation.

Zoom Options This allows you to enable/disable and set options for time based zooming. This option only applies if you have date/time data mapped to the category axis.

3D Display Options: This allows you to set several options that control the display of 3D charts. You can specify an inline series for 3D column (or similar) charts, as well as specify rendering approximation for 3D scatter and surface charts. (This improves performance for charts with a lot of data points.)

Rendering Options: This allows you to specify various rendering options for the chart for a better presentation.



Anti-Alias

☑ Enable Anti-Alias

● Chart and Text ○ Chart Only

Adjust Font

☑ Adjust Font

Translucent

□ Set Translucent

0% 100%

Gradient

□ Enable Gradient

Rendering Options Dialog

Among the options available are:

Enable Anti-Alias: This allows you to specify anti-aliasing for the chart. Anti-aliasing will smooth text and lines in the chart, creating a smoother appearance. This feature can be applied to either the entire file or just to the chart, which will leave the text unaffected.

Adjust Font: This allows you to specify the font size for text in charts to be relative to the screen resolution.

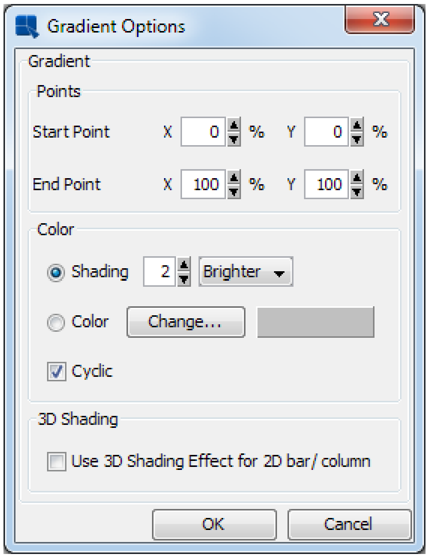
This feature allows for more precise conversions of charts to various export formats and between installations.

Translucent

This allows you to specify whether the columns/bars/areas should be translucent. This allows any columns/bars/areas "hidden" behind to show through. You can also specify the opacity by adjusting the slider, where 0% is completely opaque and 100% is completely transparent. This option is available for all 3D charts and for 2D Area, Radar and Gantt charts with data series.

Gradient:

This allows you to enable gradients. The gradient can be applied either across the entire canvas or to chart data points only. You can change the gradient options by clicking on Customize to show the following dialog:



Gradient Options

Gradient

Points

Start Point X 0 % Y 0 %

End Point X 100 % Y 100 %

Color

● Shading 2 Brighter

○ Color Change...

☑ Cyclic

3D Shading

□ Use 3D Shading Effect for 2D bar/ column

OK Cancel

Gradient Options Dialog

This dialog allows you to specify the start and end points for the gradient (based on the x-y plane), the gradient color scheme, and whether the gradient should be cyclic. The start and end points are represented as percentages with (0,0) being the top left corner of the canvas and (100, 100) being the bottom right corner of the canvas. You can also specify whether the gradient change is shading (becoming darker or brighter) or a different color. Lastly, you can specify the gradient to be cyclic i.e., toggle between the two opposites in the gradient. Note that the start and end points represent the first segment if the gradient is cyclic.

You can also enable 3D shading for certain 2D charts (Column, Bar, Stack Column, Stack Bar, 100% Column and Overlay) by selecting the *Use 3D Shading Effect for 2D bar/column* option (please note that this option is available for 2D bar/column charts only).

Font Mapping: This allows you to map system (true type) font files for the PDF export. For more information about this feature, please see Section 3.7.3.1 - PDF Font Mapping

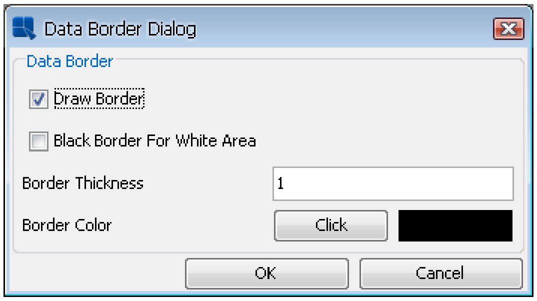
Chart Options: This brings up some specific options for the chart type that you are using. Options vary depending on chart type.

Axis Scale: This allows you to adjust the scale for any value axes. Automatic (best fit) scaling is used by default.

Axis Elements: This allows you to modify the appearance of the axes and axis labels. Options here include axis thickness, grid lines, label steps, and data formatting.

Canvas: This allows you to adjust the background canvas size. You can also specify whether to use scroll bars when the canvas size exceeds the view port.

Data Border: Add a border to data elements (columns, bars, etc).



Data Border Dialog

Data Border

☑ Draw Border

□ Black Border For White Area

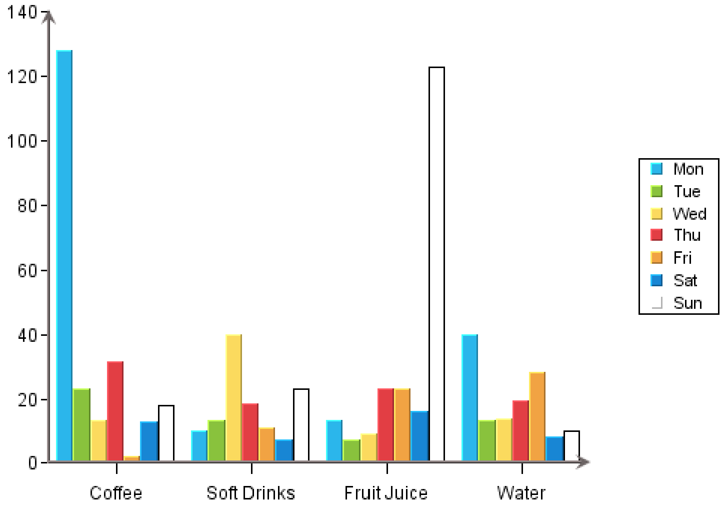
Border Thickness: 1

Border Color: Click ■

Data Border Dialog

If you select the *Draw border* option, the *Border thickness*, and the *Border color* fields will be enabled allowing you to configure the border properties.

The *Black border for white area* option can be selected even when the *Draw border* is disabled. If you do so, a simple black border will be added only to white data elements.



Column chart with the “Black border for white area” enabled

Note

This option is not available for Surface, Scatter, Line, Bubble, Box, Radar, Dial, and Polar charts.

Legend: This allows you to modify the display properties of the chart legend.

Lighting Model: This allows you to modify some of the lighting options for three-dimensional charts. You can modify both the light ambient color and the intensity.

Line and Point: This allows you to modify the display properties of any lines in a chart. For two-dimensional charts you can choose to display lines and points for any dataset as well as customize their appearance. You can also use this menu item to modify the display properties for any trend, floating, or horizontal/vertical lines.

Plot Area: This option allows you to customize appearance of the area bounded by the X and Y axes for two-dimensional charts.

No Data To Plot Message: This option allows you to set a message that appears if there is insufficient data to plot the chart. By default, the “No Data To Plot” message appears.

Flash Hintbox Customization: This option allows you to specify the font properties as well as the border and background color for the hint box in the flash export.

NULL Data Properties: This option allows you to show any category axis point that contains Null data and replace it with a different string. By default, any Null data category point is skipped.

Table: This allows you to add and configure a table displaying the chart data.

Text Properties: This option allows you to set a resize ratio for any text in the chart. From this option you can also specify to use Java 2D rotate text. This option gives rotated text a cleaner appearance. You can also specify any text replacement options.

Viewer Options: This menu item allows you to specify some configuration options for the Chart Viewer JNLP (More about Applets in JNLP: Section 1.2.7 - Run Applets in WebStart with JNLP file) when the chart is viewed. You can control what options are available in the Viewer pop-up menu. These options can also be controlled through HTML parameters.

3.5.1.4. Type Menu

This menu allows you to change the current chart and its dimension. You can switch between 2D and 3D for chart types that support representations in each dimension. You can also change chart types. Note that as you switch between chart types, some formatting information will be lost. Also, you cannot switch between Gantt charts and other chart types.

3.5.1.5. Drill-Down Menu

This menu contains options allowing you to add and navigate drill-down layers in a chart. The drill-down features are explained in Section 3.6 - Drill-Down. Note that drill-down is only available (functional) when charts are deployed independently from reports.

Add: This allows you to add a layer of data drill-down.

Remove This: This removes the current level of data drill-down.

Remove All: This removes all levels of data drill-down.

Previous This navigates to the previous layer of data drill-down.

Next: This navigates to the next layer of data drill-down.

Go To Top Level: This navigates to the top-level chart for data drill-down.

Dynamic: This allows you to enable dynamic data for drill-down.

Parameter Drill-Down: This brings up the parameter drill-down navigation window allowing you to edit, add, and remove layers of parameter drill-down.

3.5.1.6. Data Menu

This menu contains options that allows you to refresh, re-order or completely change the chart data.

Modify Data Mapping: This will bring back the Data Mapping Window, allowing you to change the data mapping for the chart.

Modify Data Source: This will take you back to the Data Source Manager, allowing you to select a new data source for the chart.

Modify Database: If the chart uses an independent data source that is from a database, this feature, like in Report Designer, will allow you to modify the database connection that the chart use. For more information about this feature, please see Section 1.3.2.5 - Editing Database Connections.

Modify Query: If the chart uses an independent data source that is from a database, this feature, like in Report Designer, will allow you to modify the query that retrieves the chart data. For more information about this feature, please see Section 1.3.2.4 - Editing Queries.

Query Parameters: This will allow you to re-initialize query parameters and change parameter values for the current chart. This option is only available if the chart uses a parameterized query as the data source.

Ordering: This option allows you to re-order the data points in a chart. You can arbitrarily change the order of any of the category elements or you can sort the categories by value.

Refresh: This will update the current chart with the latest data. The original data source must be available for this option to work.

Schedule Refresh: This allows you to set a schedule to refresh the data. This option is for deploying charts in the Chart Viewer JNLP (More about Applets in JNLP: Section 1.2.7 - Run Applets in WebStart with JNLP file). You can set a periodic refresh interval so that the chart will update itself with the latest data.

View Table: This option will bring up a window containing the data table from which the current chart is generated. The table will initially display only the first 20 records. Clicking on the *Show All Records* checkbox will display all of the records.

View Chart Data: Rather than viewing the entire data table you can just view the data points plotted in the current chart.

View Data Source Info: If the chart contains an independent data source, this option will bring up a dialog containing information about the data source that was used to create the chart. The data source type and location are displayed.

Go Back: This will go back to the original chart if you have traversed a link.

3.5.1.7. Help Menu

This menu allows you to view a version of the program and to open a documentation.

About: This shows you information about the version of the program.

Contents: This opens the EspressReport User's Guide.

3.5.1.8. Layout Menu

The options in this menu allow you to toggle various elements in the Chart Designer interface. From this menu you can turn on and off the font and color panel, the Chart Designer toolbar, and the navigation panel for 3D charts.

3.5.2. The Designer Toolbar

The toolbar at the top of the Chart Designer window offers easy access to the Chart Designer's most commonly used features and functions. The buttons perform the following functions:

Save the current chart

Undo the last change

Redo the last undone change

Change data mapping for the current chart

Modify chart-specific options

Modify line and point attributes

Change axis scale

Modify axis elements/display properties

Modify legend display

Change data ordering

Insert annotation text

Insert floating line

Add/change background image

Modify/change chart data source

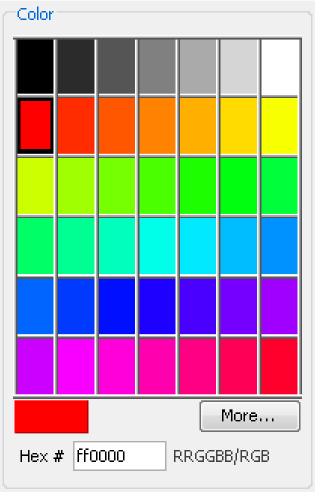
3.5.3. Color, Color set, Pattern, and Font Panels

The color, color set, and font panels on the right-hand side of the Chart Designer window allow you to modify the color of any chart object, as well as modify the font size and style for any text/labels in the chart. You can choose not to display these panels by toggling the Layout → Show font/color panel option.

For heatmap charts, you can use the *Color Panel* or *Color Set Panel* to set the colormap. The *Select Axis* option in the *Color Set Panel* is not applicable to heatmap charts. Additionally, the *Pattern Panel* does not respond to heatmap charts.

3.5.3.1. Color Panel

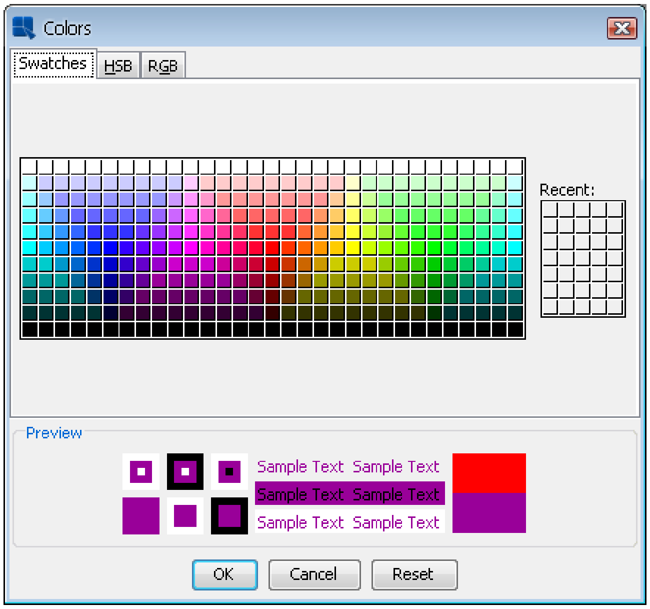
You can use the color panel to change the color of any element in the chart. To modify an element's color, first, click on it. The status bar at the bottom of the Chart Designer window will indicate which element has been currently selected. After you have selected the object, click on one of the panels in the color panel and the color of the object will change to reflect the color you selected.



Hex # ff0000 RRGGBB/RGB

Color Panel

To create a custom color for the object, first select it, then click the *More* button. This will bring up a new dialog allowing you to pick a color from a larger palette or to create a new color.

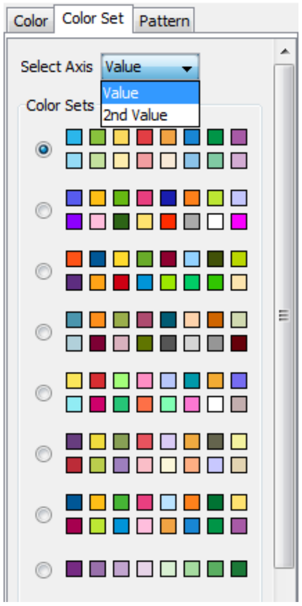


Additional Colors Dialog

From this dialog you can pick a new color from swatches or configure a custom color using HSB values or RGB values.

3.5.3.2. Color Set Panel

The color set panel allows you to choose a color scheme for your chart. It contains predefined color sets that can be applied to chart data points on the value or the second value axis.



Color Set Panel

In order to change a color set for the value or the second value axis, first select the *Color Set* tab. Then select the axis (*Value* or *2nd Value*) from the *Select Axis* select box and click the appropriate color set radio button. After that chart data points will get colors from the selected color set. Note that the *Select Axis* select box is only visible when the chart has the second value axis. By default, the value axis uses the first color set while the second value axis uses the seventh.

Please note that if you change a color of a data point manually, it will automatically unselect the selected color set. This is because of the color set that no longer corresponds to colors of data points in the chart. It is also important to note that if chart data points have more different colors than there are colors in the color set, it will automatically use colors from the beginning of the next color set, and so on. If there is no next color set available, it will use the first one instead.

3.5.3.2.1. Save Colors for Categories feature

Data points colors are closely related to the *Save Colors for Categories* feature that is available in the *Data Properties* dialog. The dialog can be opened by clicking the *Change data properties* icon on the toolbar, or by using Format → Data Properties. (Please note that this feature is not available for Stack charts and it is disabled if the chart has Single Color for All Categories.) If the feature is turned on, colors of chart data points are assigned to names of categories (or series for charts with series). If such categories (or series) then appear in the chart, it will automatically use their assigned colors. If the feature is turned off (default), colors are assigned by data points order (i.e. the first data point will get the first available color from the color set, the second will get the second color, etc.). *Save Colors for Categories* setting is automatically saved in the `.pac` file. The following example shows scenarios with the feature on and off:

Assume that a chart has three categories (“A”, “B”, and “C”) with colors taken from the following color set (blue, green, yellow, red).

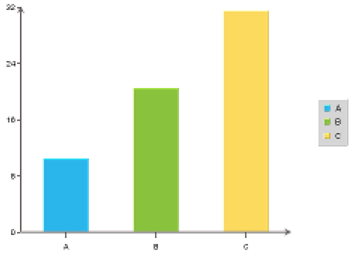


Image 1 - Example Chart

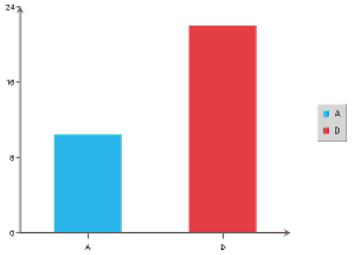


Image 2 - "Save Colors for Categories" Feature On

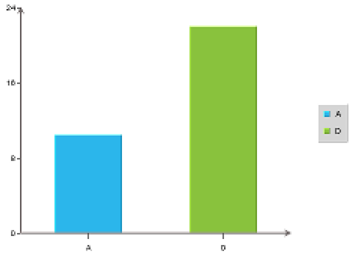


Image 3 - "Save Colors for Categories" Feature Off

Save Colors for Categories feature on

We saved the chart with the "Save Colors for Categories" feature enabled (see Image 1), so the following categories and colors were saved to the list of saved categories:

`category A` ... blue color

`category B` ... green color

`category C` ... yellow color

Now if you open the chart and data changes (e.g. there will be only categories `A` and `D` in the data - see Image 2), categories will have the following colors:

`category A` ... blue color (category `A` has blue color, because the category name `A` is present in the list of saved categories)

`category D` ... red color (category `D` has the next available color in the color set, because the category name `D` isn't present in the list of saved categories. In this case, the category will have red color, because blue, green, and yellow colors are already assigned to categories `A, B,` and `C`.

For this scenario, the color set will not be selected under the *Color Set* tab because colors of data points do not correspond to the color set.

Save Colors for Categories feature off

Here is the same situation, but assume that the chart has been saved with the "Save Colors for Categories" feature off (see Image 3). The list of saved categories will be empty this time.

After opening the chart, the categories will have the following colors:

`category A` ... blue color (category `A` has blue color, as it is the first available color from the color set)

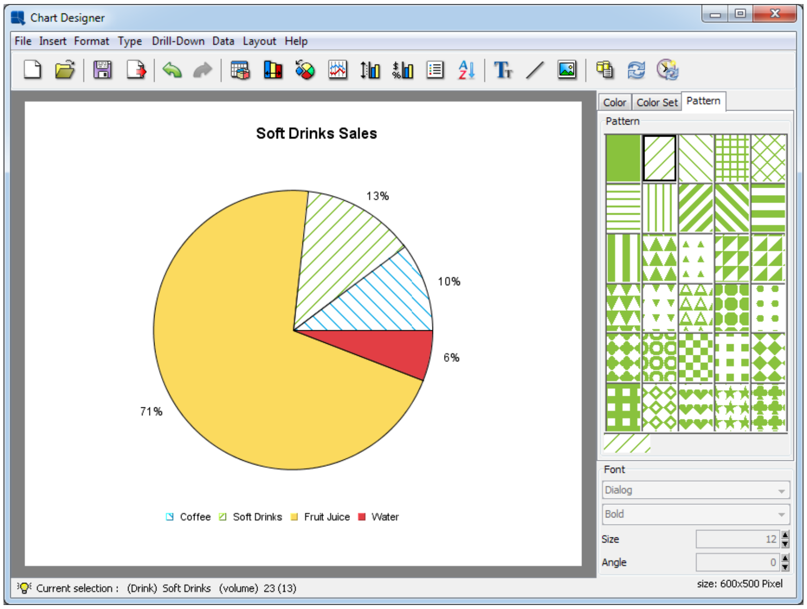
`category D` ... green color (category `D` has the second color from the color set, because the first is already assigned to the category `A`)

For this scenario, the color set will be selected under the *Color Set* tab because colors of data points correspond to the color set.

3.5.3.3. Pattern Panel

Unlike color and font panels, the pattern panel can only be applied to data points. There is a predefined pattern palette available for you to use. Similarly to how the color panel is used, you will need to first pick the data point you would like to change and then pick any pattern from the pattern palette. The pattern will be applied to the data point directly.

If a pattern has already been defined for a data point, the user can still change the color by selecting the color panel tab and picking a different color from the color palette. The color on the pattern will be changed to the new color immediately. The patterns shown on the pattern palette will change to the new color as well.



Dialog

Bold

Size 12

Angle 0

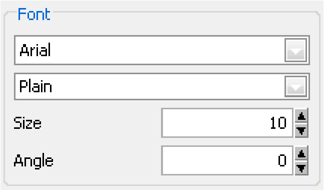
Current selection : (Drink) Soft Drinks (volume) 23 (13)

size: 600x500 Pixel

Pattern Example

3.5.3.4. Font Panel

You can use the font panel to modify the font, the font style, font size, and the font angle for labels, titles, or other text in a chart. To modify the font, you must first select the text object whose font you would like to change by clicking on it. The status bar at the bottom of the Chart Designer window will indicate which is the currently selected object.



Arial

Plain

Size 10

Angle 0

Font Panel

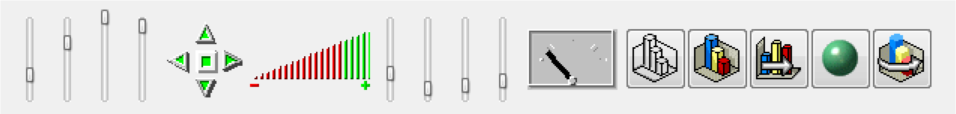
The first drop-down box allows you to select the font that you would like to use. The second drop-down box allows you to select the font style either plain, bold, italic, or bold and italic. The third box allows you to select the font size. The last box allows you to specify the angle of the text.

Certain groups of text (i.e. axis labels or data top labels) will all have the same properties. Hence, if you select one of text and modify the font properties it will apply to all of them.

ChartDesigner allows you to use the Java graphics libraries to give your text a cleaner appearance. For regular text you can use the chart anti-alias feature by selecting Format → Rendering Options. For rotated text (i.e. text not a 0 degrees), you can use the Java 2D rotate text feature by selecting Format → Text Properties. Note that these methods require Java 1.2 or higher.

3.5.4. The Navigation Panel

The navigation panel provides options for controlling a number of the properties specific to three-dimensional charts. It does not appear for two-dimensional charts and it can be hidden for three-dimensional charts by toggling the Layout → Show Navigation Panel option.



Navigation Panel

There are six controls and five buttons in the Navigation panel. Starting from the left, the six controls are step size, light position for X, Y and Z axes, navigation, zoom, scale and navigation speed. The buttons on the right control wire frame/solid mode, on/off border drawing, on/off inline series (for columnar and bar charts with series), on/off Gouraud shading, and on/off animation.

Step size: This slide bar allows you to set the incremental amount used by the navigation control to rotate or move the 3D chart. The higher the position of the horizontal bar, the larger is the increment for each step.

Light position for X, Y, and Z: User-defined light position is useful to control the position of the light and therefore the direction from which it will shine on each axis.

Navigation: This control allows you to rotate or translate the chart. The center button is a toggle switch. When it is in a depressed state (denoted by the red color of the button), clicking on any of the four triangular buttons moves the chart in the direction indicated by that button. When the center button is in an elevated state, clicking on any of the four triangular buttons rotates the chart in the direction indicated by that button. The speed of navigation may be controlled by the Navigation Speed control.

Zoom: This control allows you to effectively move the chart closer to or farther away from the viewer. To operate, you point the mouse to the control and click on the left mouse button. Then, while holding down the button, you drag the mouse to the left or right. When you drag the mouse to the right, the red area is extended to the right, and the chart will appear to be closer to you (zoom in to blow up the chart). When you drag the mouse to the left, the red area will move to the left, and the chart will appear to move farther away from you (zoom out to shrink the chart).

Scale: This is a set of four slide bars. The first three bars allow you to change the scale factors for the X, Y, and Z axes respectively. The fourth bar determines the thickness of a three-dimensional column, bar, line, or pie (depending upon the chart type). The higher the position of the horizontal bar, the larger the value. For two-dimensional charts, you can adjust the bar width for bar, column, stack bar, stack column, high-low, and HLCO charts by selecting Format → Data Properties.

Animation speed: This allows you to set the speed for chart navigation. This control is useful with the Navigation control and the Animation On/Off switch. It determines how fast the chart moves or rotates. To operate, click and drag the indicator needle to the desired position. Moving the indicator needle to the right will speed up the translation or rotation and moving the needle to the left will slow it down.

In addition, the five buttons do the following:

Wire frame/ solid mode On/Off: This toggle switch allows you to view the three-dimensional chart as a wire frame when the switch is on or as a solid object when the switch is off.

Border Drawing On/Off: This toggle switch will draw a black outline on each edge of the chart when the switch is on.

Inline Series On/Off: This toggle switch will draw the series column in the same XY plane. This option is only available for columnar and bar charts with a data series and it does not appear on the navigation panel if the chart type is not applicable.

Gouraud Shading On/Off: Gouraud shading is a sophisticated and very realistic shading feature for three-dimensional charts. When the switch is on it will begin rendering the chart to shade each individual surface.

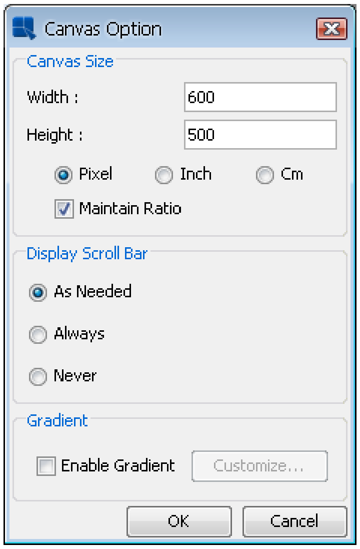
Animation On/Off: This toggle switch allows you to start/stop the animation of a three-dimensional chart. The speed of animation may be set using the Navigation Speed control. During animation, all panel controls except the animation speed control are disabled.

3.5.5. The Viewport

The viewport compromises the central portion of the Chart Designer window. Within the viewport you can select, move, and size all of the various chart elements on the canvas.

3.5.5.1. The Chart Canvas

The chart canvas is the background on which all of the chart elements are drawn. Its dimensions are the size of the finished chart. You can modify the size of the chart canvas by selection Format → Canvas. This will bring up a dialog prompting you to specify the new canvas dimensions.



Canvas Option

Canvas Size

Width : 600

Height : 500

● Pixel ○ Inch ○ Cm

☑ Maintain Ratio

Display Scroll Bar

● As Needed

○ Always

○ Never

Gradient

□ Enable Gradient

Canvas Formatting Dialog

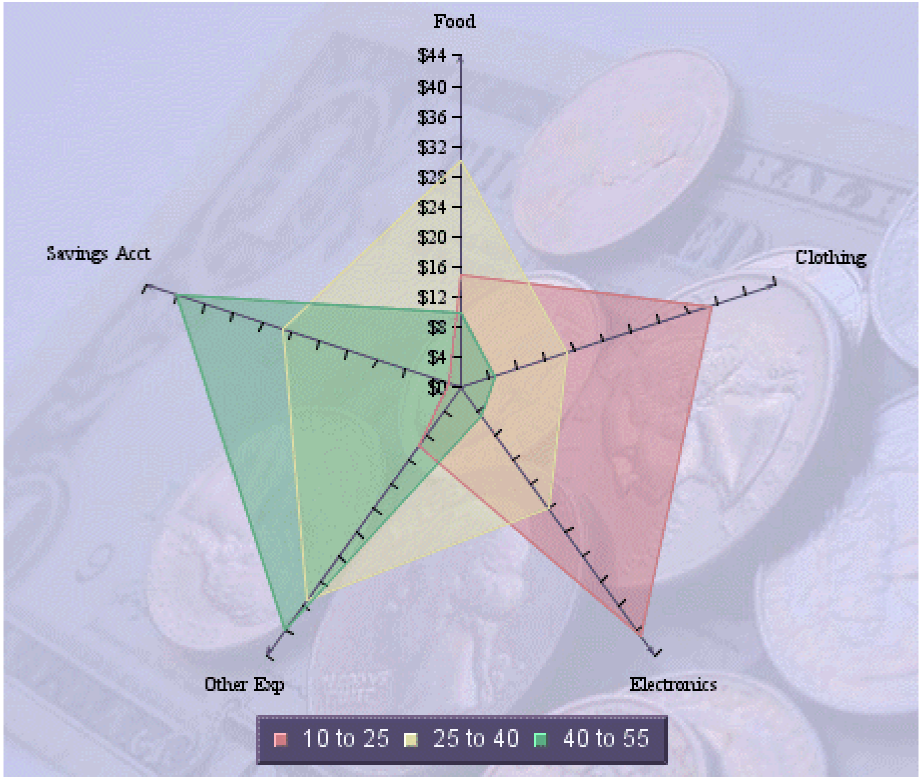
You can specify the canvas size in pixels, inches, or centimeters. From this dialog you can also specify when to use scroll bars in the viewport. By default the viewport will display scroll bars when the canvas is larger than the window. When the canvas is smaller than the viewport window, a dark gray area will appear around it.

If you're creating an embedded chart in a report, the canvas will resize to match the space that you have defined for the chart in the Report Designer. For example, if you set the space in the report to be 3 inches by 4 inches, the next time you edit the chart, the canvas will automatically size to 3 inches by 4 inches.

On this dialog, you can also set up gradient background for the canvas. The gradient settings are the same as in the *Rendering options* described in the Section 3.5.1.3 - Format Menu.

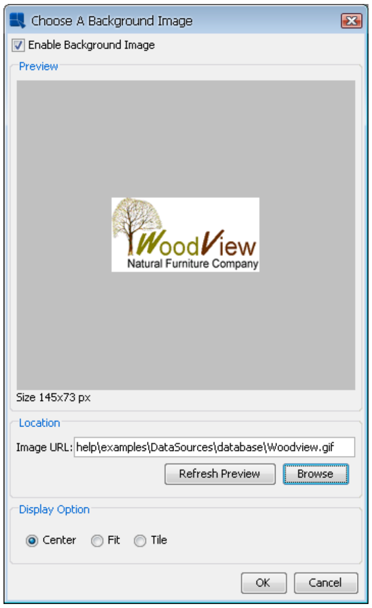
3.5.5.1.1. Background Images

You can add an image as the background of the chart instead of having a plain or a colored canvas.



Radar Chart with Background Image

You can add a background image by selecting Insert → Background or by clicking the *Background* button on the toolbar. This will bring up a dialog allowing you to specify the background image for the chart.



☑ Enable Background Image

Size 145x73 px

Image URL: help\examples\DataSources\database\Woodview.gif

● Center ○ Fit ○ Tile

Add Background Image Dialog

To insert a new image, select the *Enable Background Image* option. If you want to remove an existing background image and use simple background color instead, unselect this option.

There are two ways of inserting background images: either locate the image on the hard drive or retrieve it from an URL.

To locate an image on the hard drive, click on the *Browse* button.

To retrieve image from an URL, enter the URL in the *Image URL* text field. After that, click on the *Refresh Preview* button to verify the URL. If the image from the URL appears in the *Preview* section, the URL is correct.

If you add a background image and save the chart as TPL or CHT, the image itself is not stored with the chart. Only the path or URL is saved. If you move a TPL or CHT chart, you need to be sure that it can still access the image along the path specified. If you save the chart as PAC, the background image will be stored in the PAC file along with the chart.

3.5.5.2. Moving and Sizing Chart Elements

You can select any element in the chart by clicking on it. The status bar at the bottom of the Chart Designer window will indicate which object has been currently selected. Clicking and dragging on an object will move it around the chart canvas. Note that some objects like axis or data top labels move in tandem, while other objects like legends move independently.

You can move the entire chart plot by clicking in the plot area and dragging the mouse. This will move the entire chart around the canvas. To resize a chart click the plot area and 8 points will appear in the corners and on the sides of the plot area. Drag any of those points using the mouse to enlarge or shrink the plot area. You can also resize three-dimensional charts by using the zoom control in the navigation panel.

3.5.6. Adding Chart Elements

In addition to the default chart elements, ChartDesigner provides a number of additional elements that you can add to a chart.

3.5.6.1. Adding Text

There are two ways to add text to a chart: as titles or as plain text elements.

Adding Titles: To add titles to a chart, select Insert → Titles. This will bring up a dialog prompting you to enter titles for the chart.

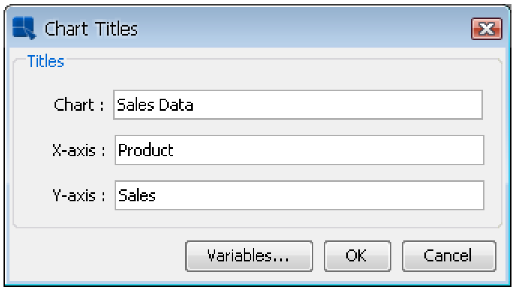


Chart : Sales Data

X-axis : Product

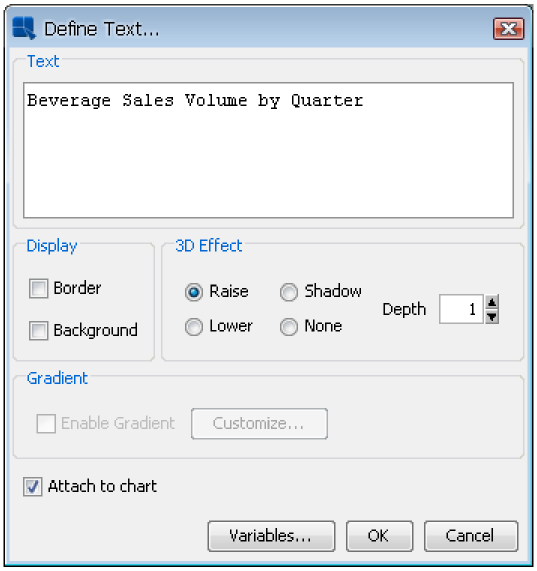
Y-axis : Sales

Add Titles Dialog

The dialog allows you to specify a main title for the chart and a title for each of the axes. Pie and dial charts, which do not have axes, prompt you for the chart title only. After you have finished specifying the titles click *OK* and they will be added to the chart. Titles are placed and sized automatically. However, they can be moved and the fonts can be changed.

Adding Text:

To add individual text fields to the chart, select Insert → Text or click the *Add Text* button on the toolbar. This will bring up a dialog prompting you to add text to the chart.



```
Beverage Sales Volume by Quarter
```

□ Border

□ Background

● Raise ○ Shadow

○ Lower ○ None

Depth 1

□ Enable Gradient

☑ Attach to chart

Add Text Dialog

From this dialog you can specify the text and configure some display options. You can specify whether to draw a border around the text or around a background. You can also specify what effect you want to apply to the background.

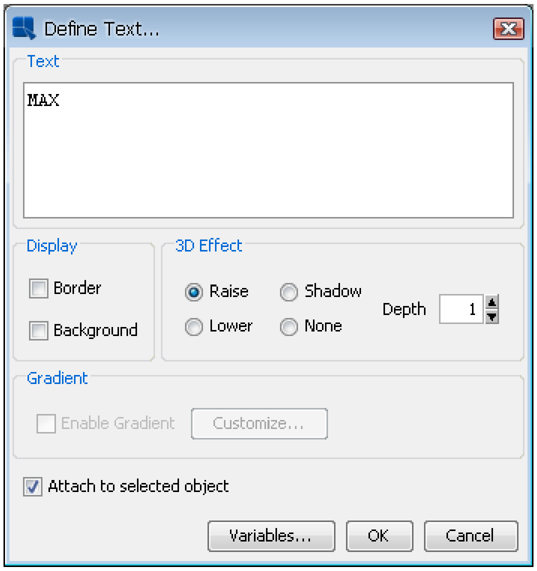
On this dialog, you can also set up gradient background for the text label. The gradient settings are the same as in the *Rendering options* described in the Section 3.5.1.3 - Format Menu.

Once you have finished specifying the text, click *OK*. You will then be able to place the text in the chart. A small rectangle will follow your pointer around the chart canvas. Click the mouse where you would like to place the text.

Annotation Text:

ChartDesigner also supports annotation. Annotation allows you to attach labels or text fields to a particular element, such as a line or the chart plot. For example, you can insert a control line showing the maximum value to a chart and attach a text label called **`MAX`** to this control line. Each time the maximum value changes, the label will adjust its position along with the control line. For more information about the control lines, please see Section 3.5.6.2.3 - Fixed Horizontal/Vertical Lines.

You can specify text to be annotation in two ways. To attach the text to the chart plot, select the *Attach to Chart* option when adding text. To attach the text to a specific object like a control line, first select the object, and then select Insert → Text or click the *Add Text* button on the toolbar. The option for *Attach to selected object* will be automatically checked. Leave it checked and any text you add will be automatically attached to the object.



MAX

□ Border
□ Background

● Raise ○ Shadow
○ Lower ○ None
Depth 1

□ Enable Gradient

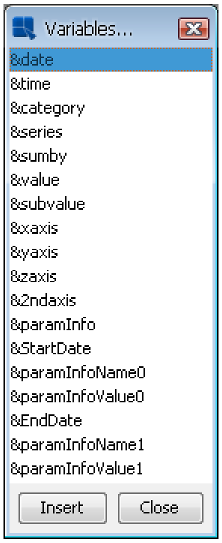
☑ Attach to selected object

Add Text (Annotation) Dialog

3.5.6.1.1. TextVariables

ChartDesigner allows you to specify certain variables within text that allow for run time substitution based on certain values/objects in the chart. For example, if your chart uses a parameterized query as the data source, you could use the **`&paramInfo`** variable to display which parameter value(s) were selected at runtime.

Both the insert titles dialog and the add text dialog have a button marked *Variables* at the bottom. This will bring up a dialog with a list of variables you can use and it will allow you to select one to add to the title or to the text.



&date
&time
&category
&series
&sumby
&value
&subvalue
&xaxis
&yaxis
&zaxis
&2ndaxis
&paramInfo
&StartDate
&paramInfoName0
&paramInfoValue0
&EndDate
&paramInfoName1
&paramInfoValue1

Variables Dialog

The following text variables are supported:

&drillInfo: This displays which data point is being drilled on for drill-down charts. This variable does not work for parameter drill-down.

&paramInfo: This displays the parameter value(s) that were selected. You can use this variable instead of &drillInfo for parameter drill-down charts.

&date: This displays the date when the chart was last drawn/redrawn.

&time: This displays the time when the charts were last drawn/redrawn.

&category: This displays the name of the category column.

&series: This displays the name of the data series column.

&sumby: This displays the name of the sum-by column.

&value: This displays the name of the value column

&subvalue: This displays the name of the secondary value column

&xaxis: This displays the name of the column that is mapped to the X-axis. This is for charts that map a value instead of a category to the X-axis like scatter or bubble charts.

&yaxis: This displays the name of the column that is mapped to the Y-axis. This is for scatter, bubble, and surface charts.

&zaxis: This displays the name of the column that is mapped to the Z-axis. This is for scatter, bubble, and surface charts.

&2ndaxis: This displays the name of the column that is mapped to the 2nd-axis.

&paramInfoName<index>: If the chart contains parameters, this displays the name of the parameter for the selected index.

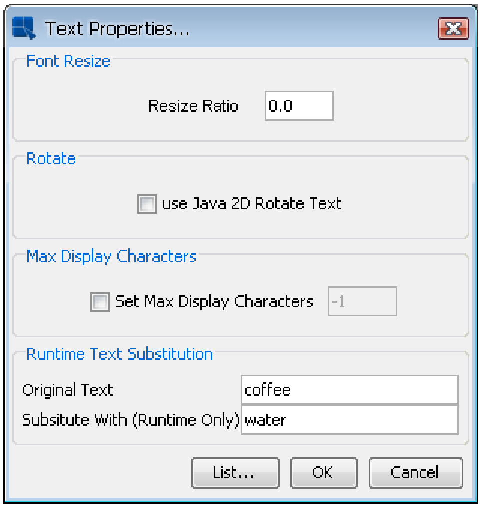
&paramInfoValue<index>: If the chart contains parameters, this displays the supplied parameter value for the selected index.

&<paramName>>: If the chart contains a parameter with this name, this will display the value selected for that parameter.

3.5.6.1.2. Text Replacement

ChartDesigner allows you to overwrite a particular piece of text in a chart. This can be useful if the data source does not use particularly intuitive column names. Note that this feature will replace all instances of the text. For example, if you have a column chart without a series that displays a column name for both the X-axis label and the legend item, you cannot use text replacement to change only the label and not the legend item. The text replacement feature will also change only whole strings and not instances where there is a partial match.

To use text replacement, select Format → Text Properties. This will bring up a dialog allowing you to specify replaced text.



Resize Ratio 0.0

□ use Java 2D Rotate Text

□ Set Max Display Characters -1

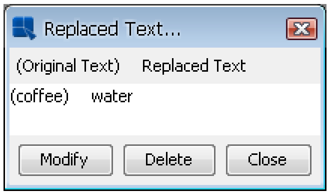
Original Text coffee

Subsitute With (Runtime Only) water

Text Properties Dialog

Please note that when making successive changes to the same text, the original text must be used. For example, assume you replaced the word “coffee” with “water”. Now if you want to change “water” to “soft drink” the text replacement should have the original text, which is “coffee” and then “soft drink” as the replacement. To remove any text replacement, simply replace the original string with itself. Hence, using the same example, you would replace “coffee” with “coffee”.

You can see a list of all the original text and the replacements by clicking on the *List* button. This will bring up a dialog listing all of the text replacement in the chart.



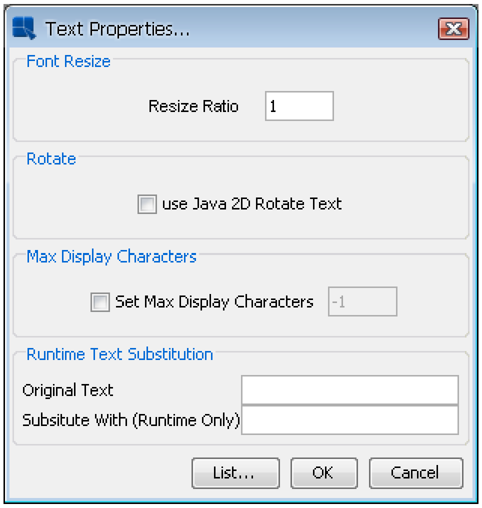
(Original Text)	Replaced Text
(coffee)	water

Replaced Text List

From this dialog you can select any of the replaced text and modify or undo the replacement by clicking the buttons at the bottom of the dialog.

3.5.6.1.3. Automatic Text Resizing

ChartDesigner has the ability to automatically adjust the font size of the text in a chart as it is resized. This is useful if you're using the same chart template to produce a number of charts in different sizes. You can specify a ratio for the font size to adjust based on changes in the chart canvas. To specify a resize ratio, select Format → Text Properties.



Resize Ratio 1

□ use Java 2D Rotate Text

□ Set Max Display Characters -1

Original Text

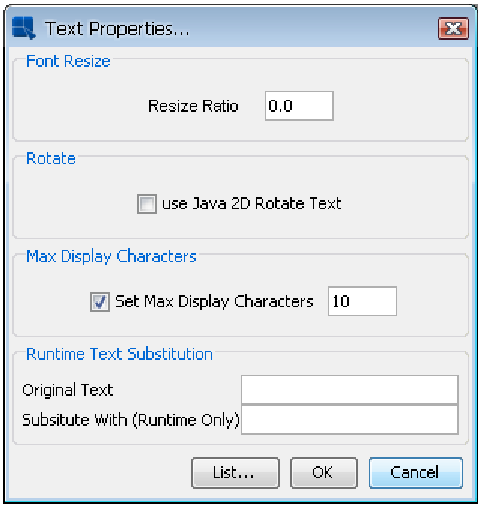
Subsitute With (Runtime Only)

Text Properties Dialog

The ratio dictates the relative percent that the font should resize in regards to the canvas. For example, say you resize a chart from 500 x 500 pixels to 250 x 250. With a resize ratio of 1 then text with 12 point font would decrease to 6 point, decreasing by the same percent as the canvas. However, with a resize ratio of 0.5 the font would decrease half as much as the canvas so our 12 point font would only decrease to 9 point.

3.5.6.1.4. Text Cropping

Long labels or text in a chart can sometimes take up too much space in the chart plot, leaving little room for the actual chart. For situations like this, ChartDesigner offers a text cropping feature for chart text. Text that is longer than a user-supplied threshold will be truncated with "...". The hint box for the chart will show the whole label. To specify text cropping, select Format → Text Properties.



Resize Ratio 0.0

□ use Java 2D Rotate Text

☑ Set Max Display Characters 10

Original Text

Subsitute With (Runtime Only)

Text Properties Dialog

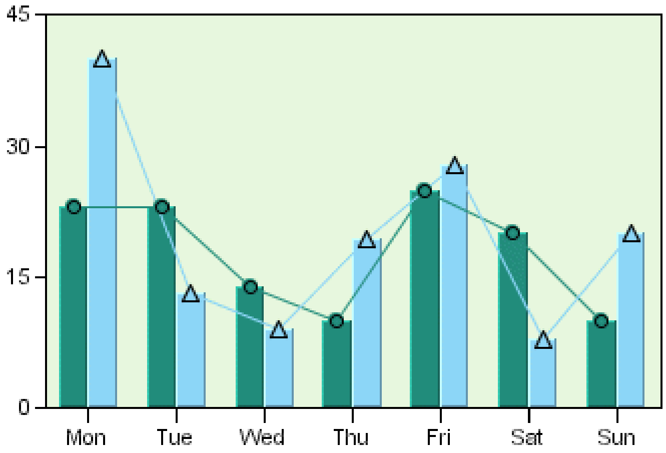
To enable text cropping, check the *Set Max Display Characters* option and specify the maximum character length in the dialog. Any text longer than the specified number of characters will be truncated.

3.5.6.2. Adding Lines

ChartDesigner allows you to add and format a number of different types of lines for charts.

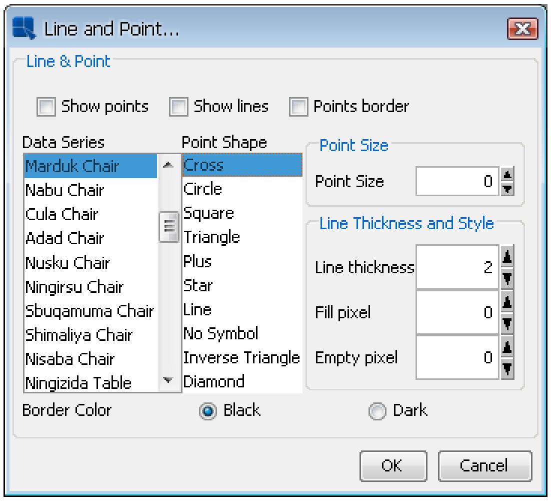
3.5.6.2.1. Line and Point Formatting

You can choose to display lines and points for all the data points in the chart for any two dimensional chart type. Note that some chart types already use this representation (i.e. line or scatter charts).



Column Chart with Lines and Points Defined

Line and point display is controlled by selecting Format → Line and Point, or click the *Line and Point* button on the toolbar. This will bring up a dialog presenting several options.


Line and Point...

Line & Point

□ Show points □ Show lines □ Points border

Data Series: Marduk Chair (selected), Nabu Chair, Cula Chair, Adad Chair, Nusku Chair, Ningirsu Chair, Sbuqamuma Chair, Shimaliya Chair, Nisaba Chair, Ningizida Table

Point Shape: Cross (selected), Circle, Square, Triangle, Plus, Star, Line, No Symbol, Inverse Triangle, Diamond

Point Size

Point Size 0

Line Thickness and Style

Line thickness 2

Fill pixel 0

Empty pixel 0

Border Color ● Black ○ Dark

Line and Point Dialog

The first three options allow you to specify whether you would like to show lines, points, and a points border for the chart. For radar, scatter, and polar charts you also have the option of showing areas. For radar and polar charts the area option will fill in the areas enclosed by the data points. For scatter charts it will draw columns from the X axis to the data points. The remaining options allow you to customize the line and point displays for each element in the data series.

For each data series element, you can specify the point shape that you would like to use. You can also control the size of the points. The default point size is 0. You can specify point sizes of -1, -2, & -3 which represent sizes of 0.75, 0.5, and 0.25 respectively. At -3 (0.25), the point will be drawn as a dot regardless of the selected point shape.

For lines you can specify the line thickness, as well as customize a dash pattern. The dash pattern is created by specifying the number of filled pixels and the number of empty pixels (between 0 and 255). The line is then drawn by dividing into segments - the number of filled pixels followed by the number of empty pixels. Setting 0 for both will result in a solid line. Setting 255 for both will result in no line being drawn.

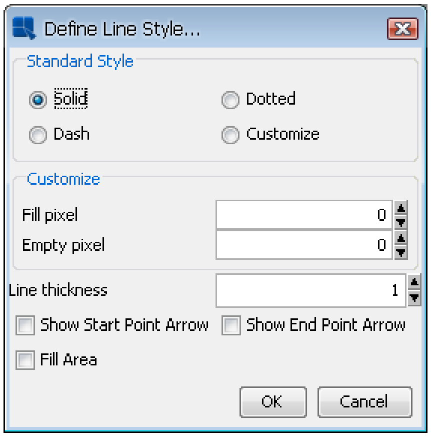
The last option allows you to change data point symbol border color to black or darker shade of symbol color.

3.5.6.2.2. Floating Lines

Floating lines are free-form lines that can be arbitrarily added to any place on the chart canvas. Often floating lines are used to point to a specific element in a chart. To add a floating line select Insert → Line, or click the *Insert Line* button on the toolbar.

When you select this option, your mouse pointer will change to a cross. Click within the chart canvas where you would like the line to begin. Each additional click will add a point to the line, allowing you to add another segment. This way you can use floating lines to draw shapes as well. Once you have finished, right-click to stop drawing. The line will then be added.

Once a line has been placed on the canvas, it cannot be moved individually. It will move with the chart plot, like annotation text. To specify options for a floating line, first select it, and then select Format → Line and Point or click the *Line and Point* button on the toolbar. This will bring up a dialog allowing you to format various properties for the floating line.



Define Line Style...

Standard Style

● Solid ○ Dotted

○ Dash ○ Customize

Customize

Fill pixel 0

Empty pixel 0

Line thickness 1

□ Show Start Point Arrow □ Show End Point Arrow

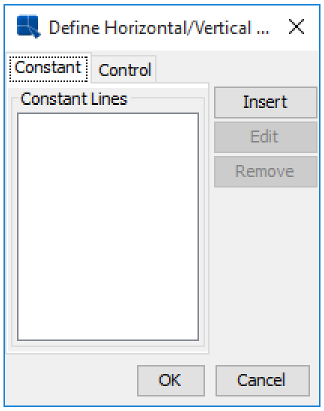
□ Fill Area

Line and Point Dialog for Floating Lines

The dialog allows you to specify a standard line style or to create a custom dash pattern in the same manner as line and point formatting. You can also specify the line thickness in pixels. The checkboxes at the bottom of the dialog allow you to place an arrowhead at the start and/or end of the line, as well as fill the area enclosed by the line to create a solid shape.

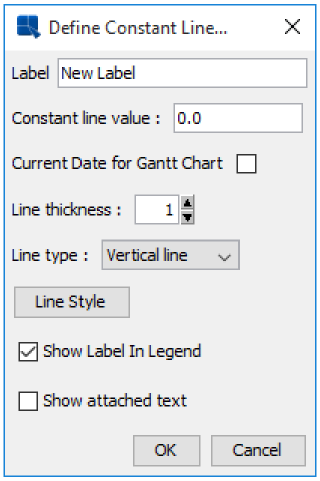
3.5.6.2.3. Fixed Horizontal/Vertical Lines

Fixed horizontal or vertical lines are lines that are drawn on one of the chart axes. These lines can also be drawn on three-dimensional charts where they appear as planes. There are two types of fixed lines: constant lines and control lines. Constant lines are lines that are fixed to a certain value in an axis. Control lines are drawn based on computed values that allow you to spot data points that are outside of certain value ranges. To add either type of line to a chart, select *Horz\Vert Line* from the *Insert* menu. This will bring up a dialog showing the list of existing horizontal/vertical lines, allowing you to edit the selected line, remove the selected line, and/or create a new line.



Define Horizontal/Vertical Lines Dialog

Clicking on *Insert* or clicking on *Edit* when an existing line is selected, respectively, will bring up a dialog allowing you to configure the selected line.



Label New Label

Constant line value : 0.0

Current Date for Gantt Chart □

Line thickness : 1

Line type : Vertical line

☑ Show Label In Legend

□ Show attached text

Constant Line Dialog

For constant lines you need to specify a label for the line, as well as the numeric value to use for the line. Note that for the category axis, the data points start with 0.5. You can also specify the line thickness and whether the line is horizontal or vertical. The last two options allow you to add an item to the chart legend for the line and whether to display any annotation for the line.

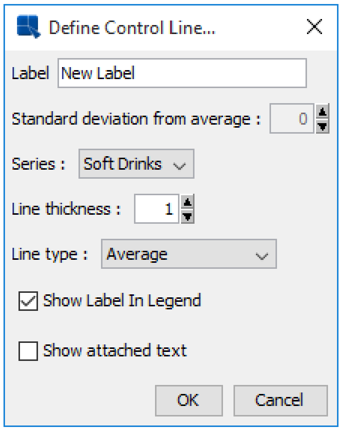
For Gantt charts, there is one extra option called *Current Date for Gantt Chart*. If you choose this option, the *Constant line value* field will deactivate and the current date will be used as the line value (the line position will be updated every time you run the chart).

For radar charts, the horizontal/vertical option is disabled. Radar charts draw constant/control lines at the same point around all the chart axes (in a similar manner to the radar grid). In addition, for radar charts, an additional option named *Circular Style* is present. By default, lines in radar charts are drawn in a segmented fashion - straight lines connect the points on each axis. Selecting this option will draw the constant/control as a circle.

Note

If you have not specified any annotation for the line then none will appear if you select the last option. For more on adding annotation to a chart, please see Section 3.5.6.1 - Adding Text.

To add a control line, click on the *Control Line* tab in the dialog.



Label New Label

Standard deviation from average : 0

Series : Soft Drinks

Line thickness : 1

Line type : Average

☑ Show Label In Legend

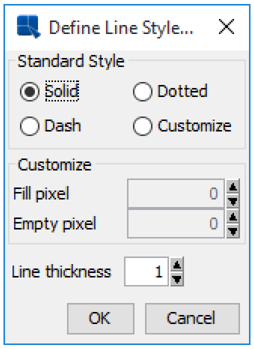
□ Show attached text

Control Line Dialog

For control lines you need to specify which series element you want to compute the value for (this option does not appear if no series is present) and how to compute the value. Options for control lines are average, minimum, maximum, and multiples of standard deviation.

After you have specified all of the options, the line will be added to the chart or the selected line will be changed, respectively. To edit any of the properties specified in the previous dialogs, you can select Insert → Horz/Vert Line again and select the line from the list. You can also double-click on the line that you want to modify.

You can change the appearance of fixed lines by first selecting the line and then selecting Format → Line and Point, or clicking the *Line and Point* icon on the toolbar. This will bring up a dialog allowing you to customize the line.



● Solid ○ Dotted

○ Dash ○ Customize

Fill pixel 0

Empty pixel 0

Line thickness 1

Line and Point Dialog for Fixed Lines

This dialog allows you to specify a standard line style or to create a custom dash pattern in the same manner as line and point formatting.

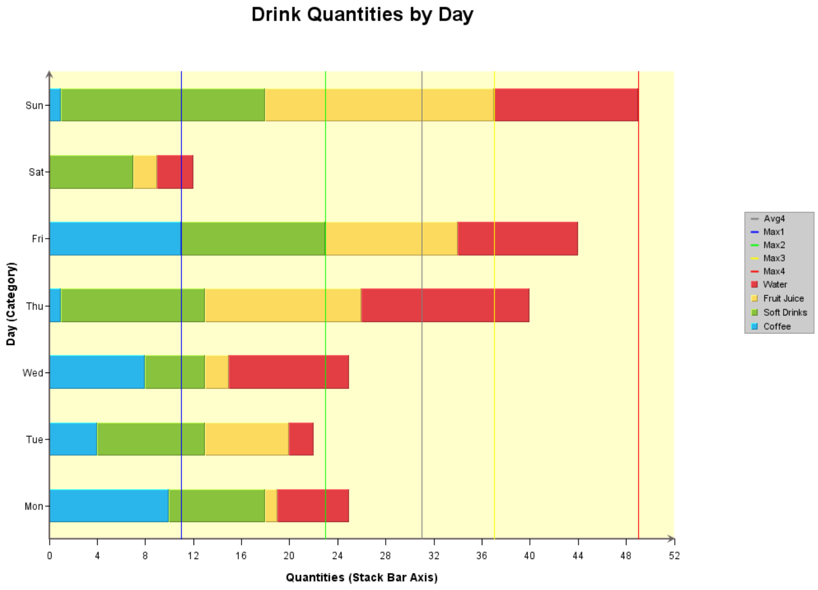
3.5.6.2.3.1. Multiple control lines for Stack Type Charts

For stack type chart, i.e., Stack Column, Stack Bar and Stack Area, we have API function:

```
...
newControlLine(int linetype, String label, int level);
...
```

can draw control line with indicated stack level. For more information about this option, see Section 3.9.6.2.5 - Adding Multiple Control Lines to Stack Type Chart. If a stack column chart with the combo chart of stack area, the API can only works on the main axis data, i.e., stack column data; In this case, the API code cannot get the secondary value to draw control lines on stack area as combo.

You can also set the control line color with this API code. The following image shows the muliple control lines in different color, and display the meaning of them in the legend of the chart.



Stack Chart with Multiple Control Lines

3.5.6.2.4. Trend Lines

A powerful feature of ChartDesigner is the ability to add trend lines to charts. Trend lines can help to show more details of a chart's data by exposing and highlighting certain trends.

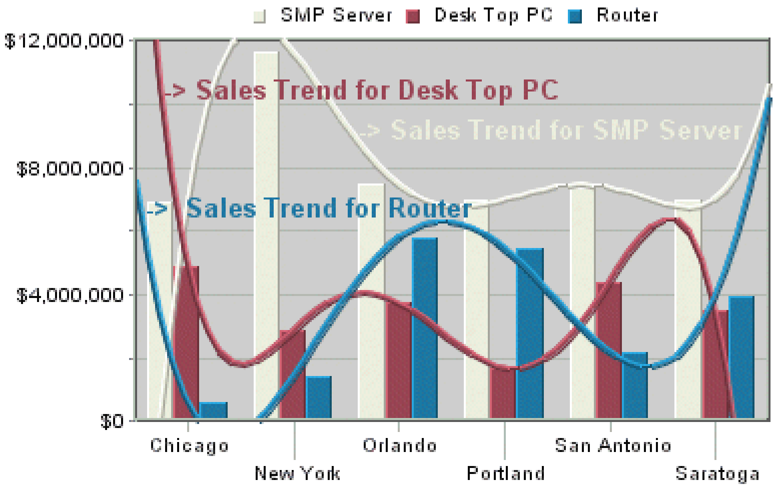
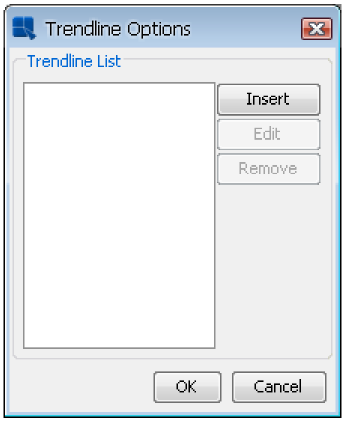


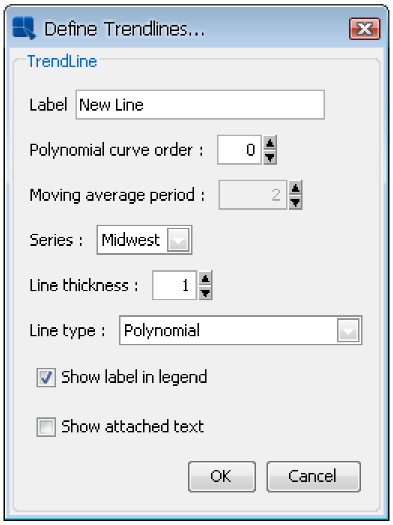
Chart with Trend Lines

To add a trend line to a chart, select Insert → Trendline. This will bring up a dialog showing the list of existing trend lines, allowing you to edit the selected trend line, remove the selected trend line, and/or create a new trend line.



Trend Line Options Dialog

Clicking on *Insert* or clicking on *Edit* when an existing trend line is selected, respectively, will bring up a dialog allowing you to configure the selected trend line.



Label New Line

Polynomial curve order : 0

Moving average period : 2

Series : Midwest

Line thickness : 1

Line type : Polynomial

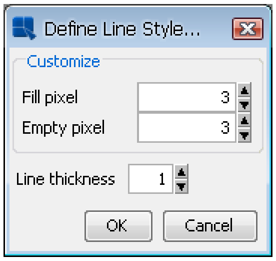
☑ Show label in legend

□ Show attached text

Define Trend Lines Dialog

In this dialog you can specify a label for the line, as well as what element of the data series to base the calculation on. The following types of trend lines are supported: a polynomial of any degree (please note that a linear trend line is a polynomial trend line of the 1st degree, i.e. the *Polynomial curve order* option has to be set to *1*), a power, exponential, logarithmic, a moving average, an exponential moving average, an triangular moving average, cubic B-spline, and a normal distribution curve. For moving averages you will need to specify the average period and for a polynomial you will need to specify the curve order. You can also specify the thickness of the line and configure whether a label in legend and the attached text should be shown. In case the chart has data series, you can configure the trendline for a specific series.

After you have specified all of the options, the trend line will be added to the chart or the selected trend line will be changed, respectively. To edit any of the properties specified in the previous dialog, you can select 'TrendLine' from the Insert menu again, and select the line from the list. You can change the appearance of the trend line by first selecting it, and then selecting Format → Line and Point, or clicking the *Line and Point* button on the toolbar. This will bring up a dialog allowing you to customize the lines.


Define Line Style...

Customize

Fill pixel: 3

Empty pixel: 3

Line thickness: 1

Line and Point Dialog for Trend Lines

This dialog allows you to create a custom dash pattern in the same manner as line and point formatting.

3.5.6.2.4.1. Normal Distribution Curve

A special type of trend line that allows you to draw a normal distribution curve for the data in the chart. In order to plot a normal distribution curve, the chart must either be a two-dimensional column or bar chart, it cannot have a data series, and the category should be numeric. Assuming these conditions are met, you can specify a normal distribution curve as one of the trend line options.

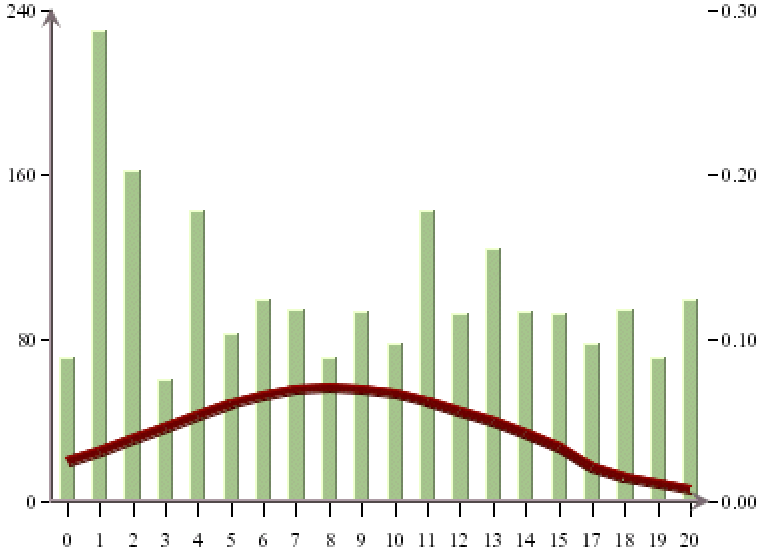
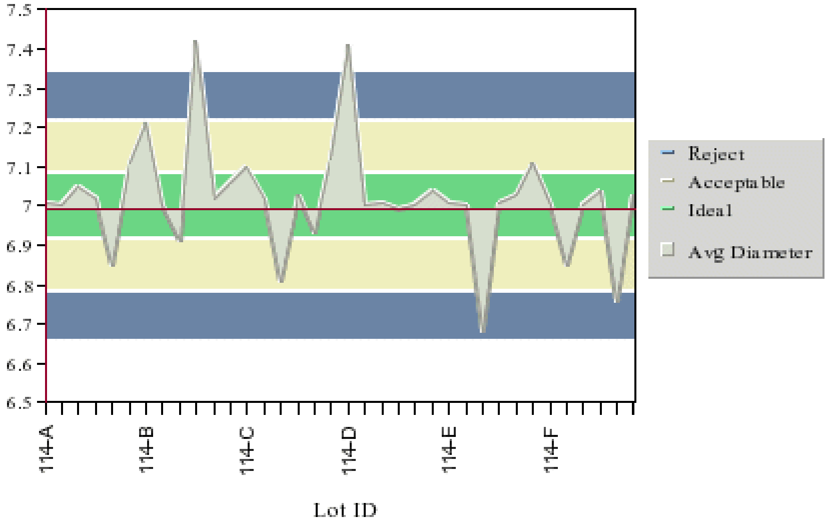


Chart with Normal Distribution Curve

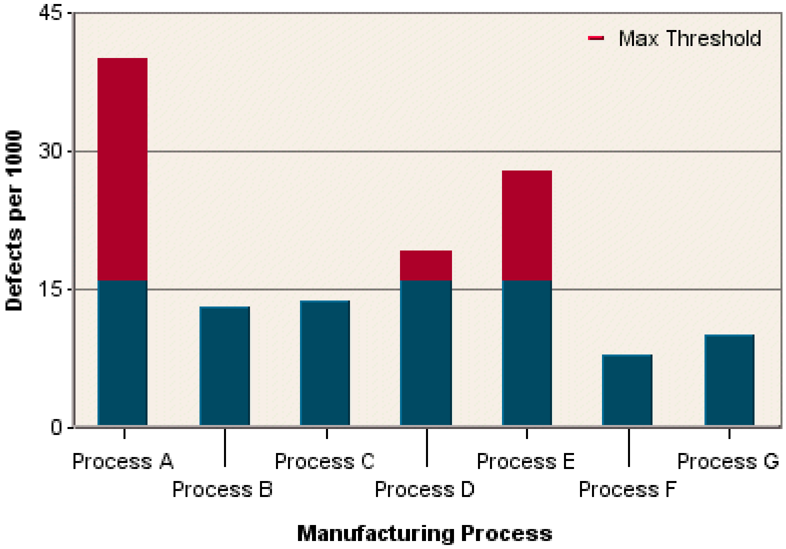
Since the scale for the curve is usually different than the scale for the value axis, the curve is shown on a secondary axis. You can modify the scale by changing the scale for the secondary axis.

3.5.6.3. Adding Control Areas

Control areas are useful for comparing the chart data against a certain range of data. For most two-dimensional charts, control areas are drawn as filled areas on the chart plot between a range of values on the chart's value axis and/or category axis. The data points are then drawn over top of the control areas giving you a quick visual reference to see which data points fall within the designated range. Instead of drawing a background area on the plot, the control areas can also be shown only where the data points intersect the control area. This feature gives users a clear visual reference when specific threshold values are reached.

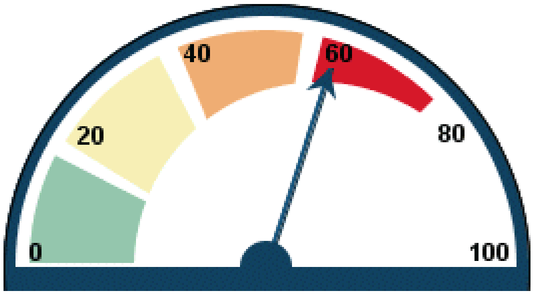


Two-Dimensional Area Chart with Control Areas



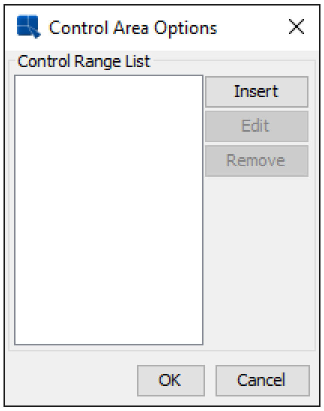
Column Chart with Control Area Drawn for Data Points

A special instance of control areas can be used for dial charts. For dial charts control areas are drawn as arcs on the face of the dial, allowing you to see if the dial hands (data points) fall within the range. Note that control areas are not available for radar and pie charts.



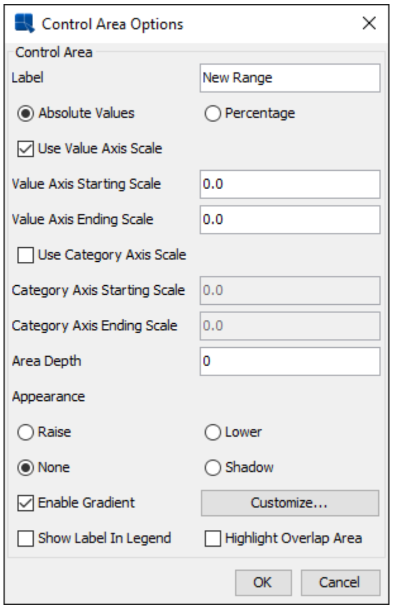
Dial Chart with Control Areas

To add a control area to a chart, select Insert → Control Area. The following dialog will appear showing the list of existing control areas, allowing you to edit the selected control area, remove the selected control area, and/or create a new control area.



Control Area List

Clicking on *Insert* or clicking on *Edit* when an existing control area is selected, respectively, will bring up a dialog allowing you to configure the selected control area. If your chart is not a dial chart, the following dialog will open.



Label: New Range

● Absolute Values ○ Percentage

☑ Use Value Axis Scale

Value Axis Starting Scale: 0.0

Value Axis Ending Scale: 0.0

□ Use Category Axis Scale

Category Axis Starting Scale: 0.0

Category Axis Ending Scale: 0.0

Area Depth: 0

Appearance

○ Raise ○ Lower

● None ○ Shadow

☑ Enable Gradient

□ Show Label In Legend □ Highlight Overlap Area

Control Area Configuration Dialog

The following options are provided for control areas:

Label: This option allows you to specify a label for the control area.

Absolute Values: This option allows you to specify the scale in absolute values.

Percentage: This option allows you to specify the scale in percentage.

Use Value Axis Scale: This option allows you to specify whether the control area should be bounded by values on the value axis of the chart.

Value Axis Starting Scale: This is the lower bound for the control area on the value axis.

Value Axis Ending Scale: This is the upper bound for the control area on the value axis.

Use Category Axis: This option allows you to specify whether the control area should be bounded by values on the category axis of the chart.

Category Axis Starting Scale: This is the lower bound for the control area on the category axis.

CategoryAxis Ending Scale: This is the upper bound for the control area on the category axis.

Area Depth: This option specifies the depth for any of the appearance styles. If no style is selected, the depth will have no effect.

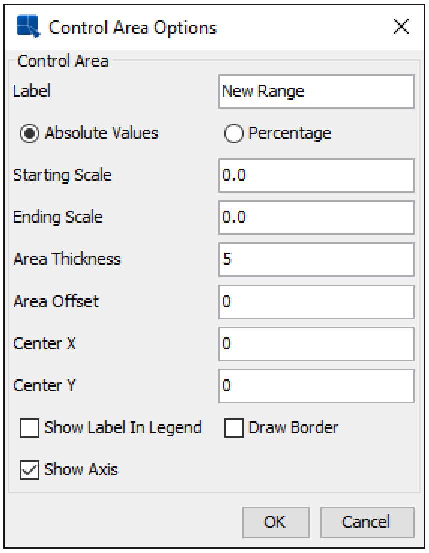
Appearance: This option allows you to specify a 3D or shadow effect for the control area. If the area depth is specified as zero, the appearance will not take effect.

Enable Gradient: Enable color gradient for the control. Gradient settings are described in Section 3.5.1.3 - Format Menu

Show Label In Legend: This option specifies whether or not to show the control area label in the chart legend.

Highlight Overlap Area: This option will only show the control area where the data points overlap the control area boundaries.

If your chart is a dial chart, then a different dialog will appear when you select Insert → Control Area and then click on the *Insert* or *Edit* button.



Label: New Range

● Absolute Values ○ Percentage

Starting Scale: 0.0

Ending Scale: 0.0

Area Thickness: 5

Area Offset: 0

Center X: 0

Center Y: 0

□ Show Label In Legend □ Draw Border

☑ Show Axis

Control Area Dialog for Dial Charts:

The following options are provided for dial chart control areas:

Label: This option allows you to specify a label for the control area.

Absolute Values: This option allows you to specify the scale in absolute values.

Percentage: This option allows you to specify the scale in percentage.

Starting Scale: This is the value where the control area begins.

Ending Scale: This is the value where the control area ends.

Area Thickness: This option allows you to set the thickness for the control area

Area Offset: This option allows you to specify the offset in pixels from the edge of the dial chart

Center X: This sets the X coordinate for the center of the control area. 0 shares the same center point as the dial face. You can specify a new number (either negative or positive) pixels to specify an offset position from the center of the dial.

Center Y: This sets the Y coordinate for the center of the control area. It works in the same way as the previous option.

Show Label in Legend: Specifies whether to show the control area label in the chart legend.

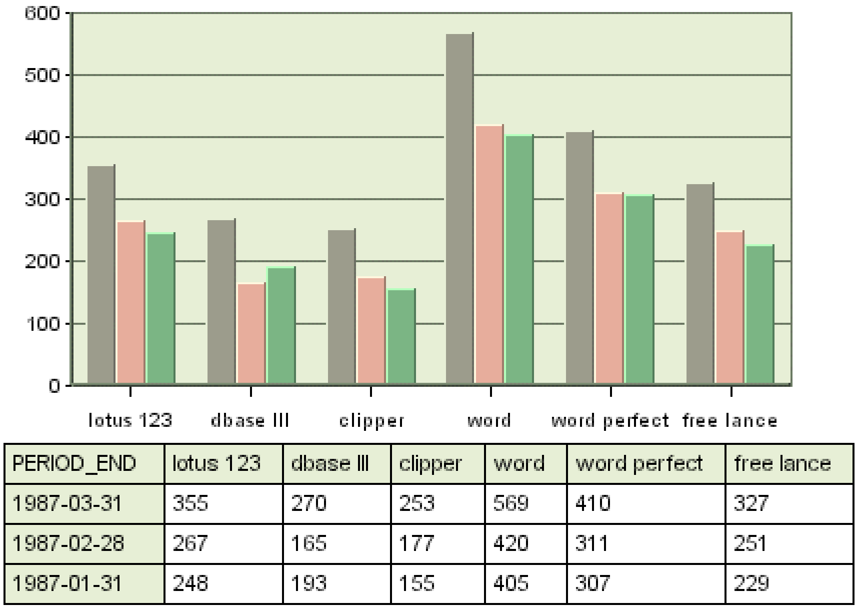
Draw Border: This option allows you to draw a border around the control area.

Show Axis: This option allows you to show or hide the axis for the remaining area not covered by the control range.

After you have specified all of the options, the control area will be added to the chart or the selected control area will be changed, respectively. To edit any of the properties specified in the previous dialog, you can select Insert → Control Area again, and select the control area from the list. You can also double-click on the control area that you want to modify.

3.5.6.4. Adding Tables

In addition to displaying charts, you can also display a table showing the data points displayed in the chart. The table can be placed below or to the right of the chart plot.



PERIOD_END	lotus 123	dbase III	clipper	word	word perfect	free lance
1987-03-31	355	270	253	569	410	327
1987-02-28	267	165	177	420	311	251
1987-01-31	248	193	155	405	307	229

Chart with Table

To add a table to a chart, or to modify the various display options for a table, select Format → Table. This will bring up a dialog allowing you to customize various options for the table display.

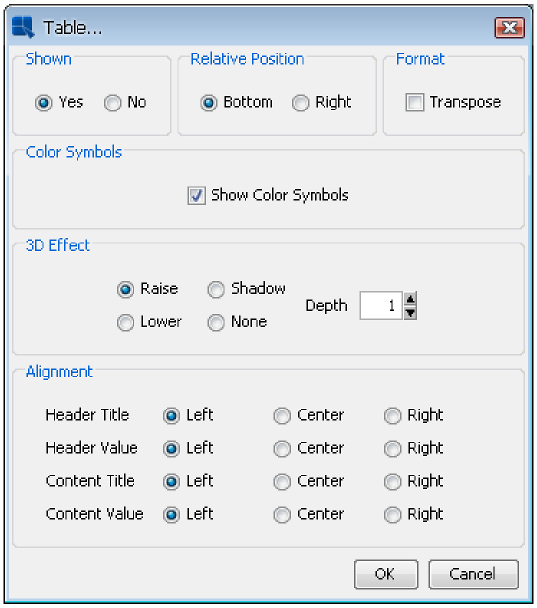


Table...

Shown: ● Yes ○ No

Relative Position: ● Bottom ○ Right

Format: □ Transpose

Color Symbols: ☑ Show Color Symbols

3D Effect: ● Raise ○ Shadow ○ Lower ○ None Depth 1

Alignment

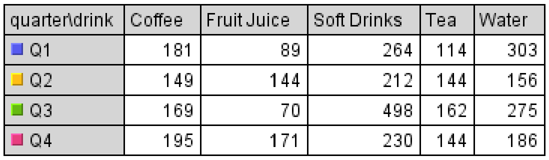
Header Title	● Left	○ Center	○ Right
Header Value	● Left	○ Center	○ Right
Content Title	● Left	○ Center	○ Right
Content Value	● Left	○ Center	○ Right

Format Table Dialog

From this dialog you can specify whether or not to display the table, as well as what relative position to give the table either to the bottom or right-hand side of the chart plot. You can also specify a 3D effect for the table and its depth.

The *Transpose* checkbox allows you to swap the columns and rows of the table. By default, the category elements are drawn as columns and the data series elements as rows.

The *Show Color Symbols* option allows you to show/hide color boxes for data series in the table.



quarter\drink	Coffee	Fruit Juice	Soft Drinks	Tea	Water
🟦 Q1	181	89	264	114	303
🟨 Q2	149	144	212	144	156
🟩 Q3	169	70	498	162	275
🟥 Q4	195	171	230	144	186

Chart table with color boxes

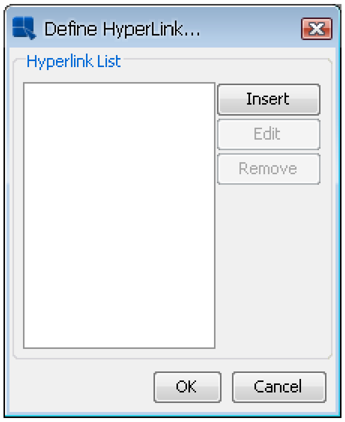
The *Alignment* options allows you to specify horizontal alignment of the text in the table cells, either left, center, or right. The alignment can be set for row headers, column headers, and inner table cells.

Note

If there isn't enough room in the chart canvas, not all data points will be displayed in the table. The table size adjusts with the canvas size and also with the font size in the table cells.

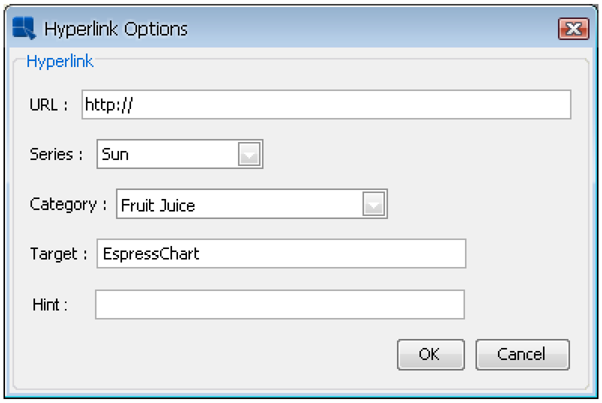
3.5.6.5. Adding Hyperlinks

ChartDesigner has an capability to add hyperlinks to any data point in a chart. Links can be specified for either single data points or multiple elements. Any added hyperlinks will be applied to both the data points on the chart and to their respective fields on the legend. To add a hyperlink to a chart, select Insert → Link or right-click on a data point and select *Insert Link* from the pop-up menu. This will bring up a dialog showing a list of existing hyperlinks which you can configure, remove, and/or create new ones.



Insert Link Dialog

Clicking on *Insert* or clicking on *Edit* when an existing hyperlink is selected will bring up a dialog allowing you to configure the selected hyperlink.



URL : http://

Series : Sun

Category : Fruit Juice

Target : EspressChart

Hint :

Define Link Dialog

The URL field allows you to specify a Web page that you want to link.

The *Series* and *Category* drop down menus allows you to select an element in the data series and category elements for the hyperlink. You can also link to all data series elements or all category elements.

You can specify the `Target` parameter recognized by HTML when specifying a hyperlink to be attached to a data point or data series. This can be used to determine whether the new HTML page should open in a new browser window, in the same browser window, or whether the new page should occupy the same portion of the page as the current page.

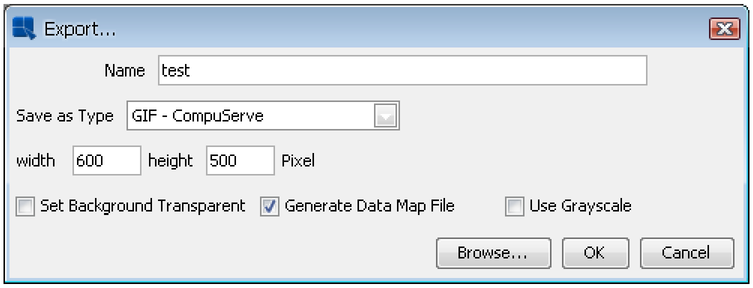
The *Hint* field allows you to enter text that will pop-up when you move your mouse cursor over a data point. If you want to create pop-up labels without hyperlinks, you can leave the URL field blank and only specify the hint.

3.5.6.5.1. Viewing Hyperlinks

If you specify hyperlinks for charts, they will be active only when the chart is exported to Flash format*. For most image formats such as PNG, JPG, GIF, etc, an image map file containing information for the link will be automatically generated when you export the chart. You can insert the image and image map into an HTML file to view the image with clickable links.

Note

* For Flash export, when the user clicks on the hyperlink, there will likely be a warning message prompted by Flash saying that there was a potentially unsafe operation. You can turn this warning off by clicking on settings and adding the chart into the list of trusted locations.



Name test

Save as Type GIF - CompuServe

width 600 height 500 Pixel

□ Set Background Transparent ☑ Generate Data Map File □ Use Grayscale

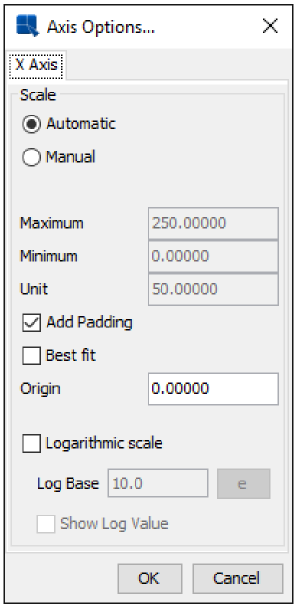
Export Dialog

3.5.7. Formatting Chart Axes

ChartDesigner provides a number of extensive formatting capabilities for the chart axes. Users can customize everything from the axis scale to the way how axis labels should be displayed.

3.5.7.1. Axis Scale

By default, the scale of any value axes in the chart is calculated to provide a 'best fit' for the data being plotted. This is often a useful feature if the data being displayed can change radically. However, you may often want to set the scale of the axes manually. To modify the axis scale, select Format → Axis Scale, or click the *Axis Scale* button on the toolbar. This will bring up a dialog allowing you to format the scale for any value axes in the chart.



● Automatic

○ Manual

Maximum 250.00000

Minimum 0.00000

Unit 50.00000

☑ Add Padding

□ Best fit

Origin 0.00000

□ Logarithmic scale

Log Base 10.0

□ Show Log Value

Axis Scale Dialog

The following options are provided:

Automatic:	This turns on automatic scaling for the axis. This is the default option.
Manual:	This turns on manual scaling, allowing you to set the axis scale to your preference.
Maximum:	This is the highest value on the axis scale.
Minimum:	This is the lowest value on the axis scale.
Unit:	This is the step interval between successive labels.
Add Padding:	This will raise the highest value of the axis to create a cushion between the max value of the data and the top of the chart.

Best fit: This will automatically place the origin of the chart based on the minimum and maximum values of the data.

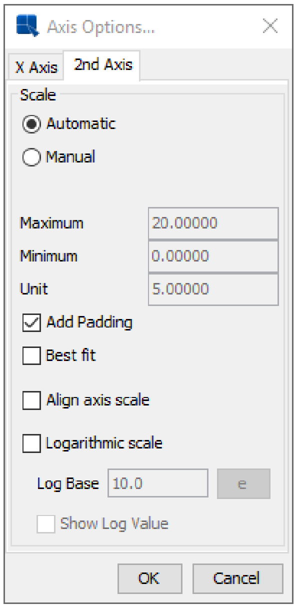
Origin: This allows you to specify where the X and Y axes should intersect. This is usually set to zero.

Logarithmic Scale: This option creates a logarithmic scale for the given axis. It's valid only if the axis in question contains positive values.

Log Base: This allows you to specify the base for the log value.

Show Log Value: This specifies whether to show log values in the axis labels or not.

The axis scale dialog will have a tab for each axis in the chart. There's a unique option available for secondary axes which allows you to align the axis scale. It will apply all options from the primary axis to the secondary axis, giving both axes the exact same scale.



Scale

● Automatic

○ Manual

Maximum 20.00000

Minimum 0.00000

Unit 5.00000

☑ Add Padding

□ Best fit

□ Align axis scale

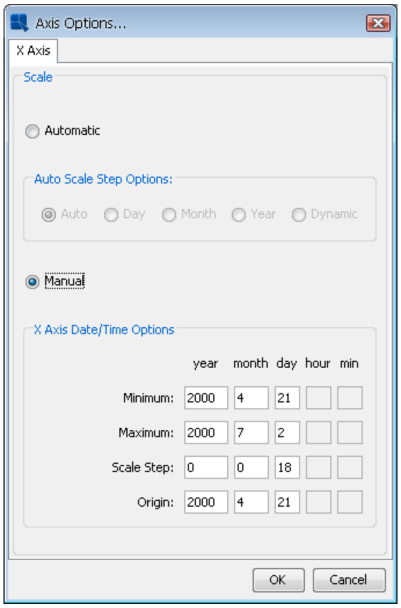
□ Logarithmic scale

Log Base 10.0 e

□ Show Log Value

Axis Scale Dialog for Secondary Axis

The axis scale dialog is different for a Gantt chart. You can still select the *Automatic* option to have the scale configured automatically or the *Manual* option to set the scale manually. The *Automatic* axis scale has a few Auto Scale Step Options: Auto, Day, Month, Year, and Dynamic. The *auto* option always finds a best fit. The *dynamic* option also finds a best fit, but unlike the *auto* option, it uses standard step intervals only (for example: 1 month, 1 year etc...). When the *Manual* axis scale is chosen, the *Maximum*, *Minimum*, *Unit* and *Origin* are replaced with *Maximum Date*, *Minimum Date*, *Scale Step* and *Origin Date* respectively and these new settings take in a Date/Time (represented by yyyy, MM, dd, hh, mm).



Scale

○ Automatic

Auto Scale Step Options:

● Auto ○ Day ○ Month ○ Year ○ Dynamic

● Manual

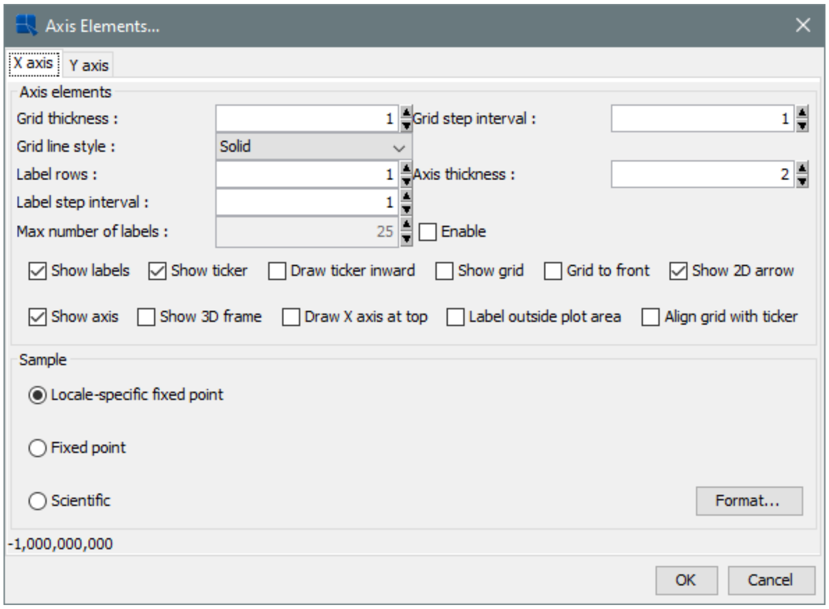
X Axis Date/Time Options

	year	month	day	hour	min
Minimum:	2000	4	21		
Maximum:	2000	7	2		
Scale Step:	0	0	18		
Origin:	2000	4	21		

Axis Scale Dialog for Gantt Chart

3.5.7.2. Axis Elements

The appearance properties of the axes and the axis labels are controlled through the axis elements dialog. To invoke the axis elements dialog, select Format → Axis Elements or click the *Format Value Elements* button on the toolbar. This will bring up the following dialog, allowing you to customize elements in all chart axes. You can also customize the appearance of dial and pie chart labels from this dialog.



Axis elements

Grid thickness : 1
Grid step interval : 1
Grid line style : Solid
Label rows : 1
Axis thickness : 2
Label step interval : 1
Max number of labels : 25 □ Enable

☑ Show labels ☑ Show ticker □ Draw ticker inward □ Show grid □ Grid to front ☑ Show 2D arrow

☑ Show axis □ Show 3D frame □ Draw X axis at top □ Label outside plot area □ Align grid with ticker

Sample

● Locale-specific fixed point

○ Fixed point

○ Scientific

-1,000,000,000

Axis Elements Dialog

A tab will appear in this dialog for each axis in the chart. The dialog allows you to perform the following options. Note that some options are only available for certain chart types, certain data types, and on certain axes.

Grid thickness: This allows you to specify the thickness of any grid lines along the axis.

Grid step interval: This option allows you to set the grid step interval for any grid lines along the axis.

Grid line style: This option allows you to select the grid line style (solid, dotted, dash).

Label rows: This option allows labels to be displayed in alternating rows. This can prevent overlapping. This option is only available for X-axis.

Axis thickness: This option allows you to set the thickness of the axis in pixels. Note that this setting is applied to all axes in the chart.

Label step interval: This option allows you to set the label step interval for the data. For example, setting this to 2 will draw the label for every other data point in the chart.

Label interval unit: This option allows you to select the unit to be used when sorting and representing time-based data (date, time, or timestamp). Selecting tickers will use the data as it is read by Chart Designer. You can also select Dynamic for time-based data and that will choose an appropriate scale (depending on number of data points and range of data).

Max number of labels: This option allows you to select the maximum number of labels and tickers displayed on the axis. If the number of labels exceeds the max count, the label step will be re-calculated.

Ascending/Descending: This option allows you to order and filter time-based data. You can sort the data in ascending or descending order, as well as specify the starting (or ending) point for the data.

Show labels: This option allows you to remove or display the labels for each axis.

Show ticker: This option allows you to remove or display the axis tickers.

Draw ticker inward: This option will draw the axis tickers inside the plot area instead of outside (default).

Show sub-tickers: This feature is only available for the value axis when the axis scale is set to logarithmic with a log base of 10. This feature will draw interval tickers (non-uniform) between the points on the value axis.

Show grid: This option allows you to remove or display the grid for each axis.

Grid to front: This option allows you to draw the grid lines on top of the data elements in the chart. By default the data points are drawn on top of the grid.

Show 2D arrow: This option allows you to remove or display the arrowhead at the end of the axis. Note that this option applies to all chart axes and it is only available for two-dimensional charts.

Show axis: This option allows you to remove or display the axis (for two-dimensional charts) or the wall (for three-dimensional charts).

Show 3D frame: This option allows you to remove or display a frame around the chart. Note that this option applies to all chart axes and it is only available for three-dimensional charts.

Label outside plot area: This option sets the labels to be placed outside of the plot area, irrespective of where the axis is. This feature can be useful for category axis labels if you're plotting data with both positive and negative values.

Align grid with ticker: This option aligns the grid line with the ticker instead of placing it between tickers. This places the ticker and the corresponding grid line along the same line. This option only applies to the category axis of column-type charts.

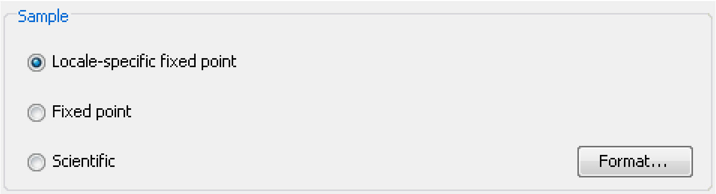
Swap Y-axis position: This option will swap the primary and secondary value axes. This option can only be found under the 2nd Axis tab.

Draw X-axis at top: This option allows the X-axis to be positioned at the top of the chart instead of the default bottom position. This option is available for two-dimensional column, bar, scatter, high-low, HLCO, bubble, and Gantt charts.

3.5.7.2.1. Axis Label Formatting

The axis elements dialog also allows you to format the appearance of the axis labels, depending on what type of data is plotted on the axis. The *Format Options* portion of the dialog contains the label formatting dialog.

Formatting Numeric Data For numeric data there are three primary options for display formatting: locale-specific fixed point, fixed point, and scientific. Additional options will be displayed if you click on the *Format* button.



Sample

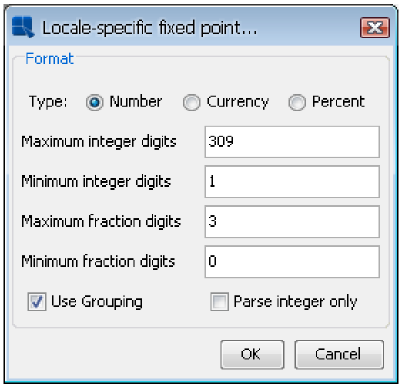
● Locale-specific fixed point

○ Fixed point

○ Scientific

Numeric Data Format Options

Locale-Specific Fixed Point This will change the format of the data depending on the locale in which it is being viewed. Additional formatting for this option allows you to specify whether the data should be displayed as a number, currency, or percentage. In addition, you can set the maximum and minimum number of integer digits and fraction digits. Other display attributes will vary depending on locale.



Format

Type: ● Number ○ Currency ○ Percent

Maximum integer digits: 309

Minimum integer digits: 1

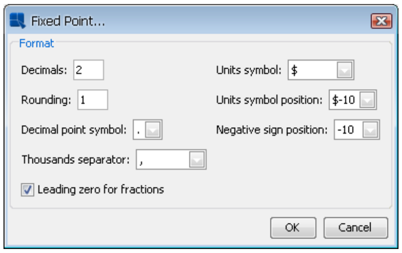
Maximum fraction digits: 3

Minimum fraction digits: 0

☑ Use Grouping ☐ Parse integer only

Locale-Specific Formatting Options

Fixed Point: This will keep the data format consistent, regardless of locale. Additional formatting for this option allows you to set the number of decimals, rounding for digit number, unit symbols, negative sign position, decimal and thousands separator, and enable leading zeros for fractions



Decimals: 2

Units symbol: $

Rounding: 1

Units symbol position: $-10

Decimal point symbol: .

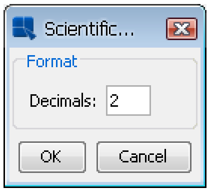
Negative sign position: -10

Thousands separator: ,

☑ Leading zero for fractions

Fixed Point Formatting Options

Scientific: This will display the data in scientific notation. Additional formatting for this option allows you to set the number of decimals.

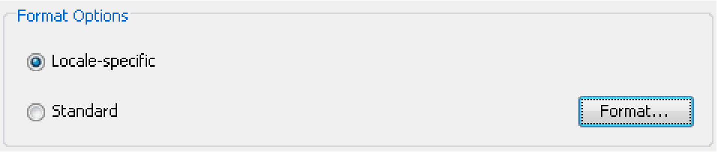


Decimals: 2

Scientific Formatting Options

Formatting Date/Time Data

For date/time data there are two primary options for display formatting: locale specific and standard. Additional options will be displayed if you click on the *Format* button. The available options will vary depending on the nature of your data. Date, time, and timestamp data will bring up date, time, and date & time options respectively.

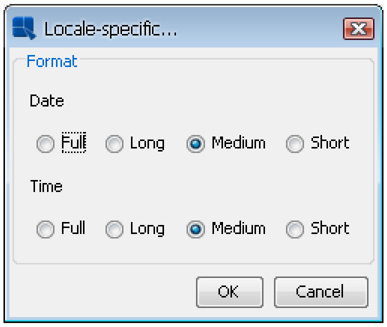


● Locale-specific

○ Standard

Date/Time Data Format Options

Locale-Specific: This will change the format of the data depending on the locale in which it is being viewed. Additional formatting for this option allows you to select full, long, medium, or short notations for date and time information. Other display attributes will vary depending on locale.



Date

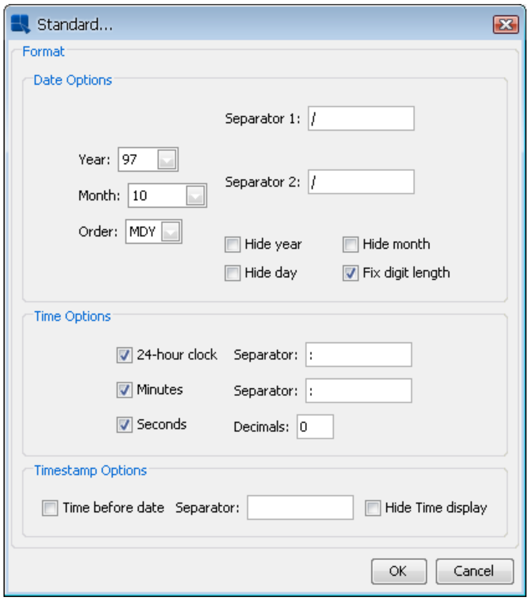
○ Full ○ Long ● Medium ○ Short

Time

○ Full ○ Long ● Medium ○ Short

Locale-Specific Formatting Options

Standard: This will keep the data format consistent, regardless of locale. Additional formatting for this option allows you to select year and month displays, as well as the order in which month, day, and year information is presented. You can also select the characters that you want to be used as separators. Time options allow you to display hours, minutes, and/or seconds and also to select the separators between them. For timestamp data, you can select to display the time before or after the date, as well as the separator to be used between them.



Date Options

Separator 1: /

Year: 97

Separator 2: /

Month: 10

Order: MDY

☐ Hide year ☐ Hide month

☐ Hide day ☑ Fix digit length

Time Options

☑ 24-hour clock Separator: :

☑ Minutes Separator: :

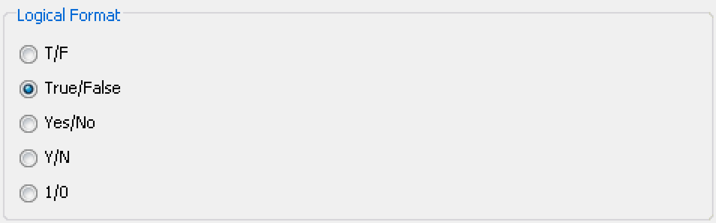
☑ Seconds Decimals: 0

Timestamp Options

☐ Time before date Separator: ☐ Hide Time display

Standard Formatting Options

Formatting Logical Data: There are five options available for displaying logical or Boolean data: T/F, True/False, Yes/No, Y/N, and 1/0.



Logical Format

○ T/F

● True/False

○ Yes/No

○ Y/N

○ 1/0

Logical Data Formatting Options

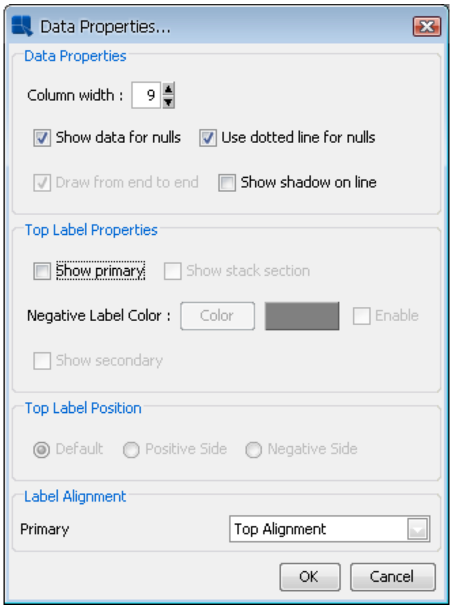
Any changes you make to the data formatting will take effect after you click on the *OK* button in the axis elements dialog. Note that there are no additional formatting options for string data.

3.5.8. Formatting Plot/Data Elements

ChartDesigner provides a number of ways to customize and configure the way data points are drawn and annotated on the chart, as well as the chart plot itself.

3.5.8.1. Data Properties

Many of the data display options are controlled through the data properties dialog. From this dialog you can control the size of bars/columns, set display options for null values, and specify options for data labels. To invoke the data properties dialog, select Format → Data Properties or click the *Data Properties* button on the toolbar. This will bring up the following dialog:



Data Properties...

Data Properties

Column width : 9

☑ Show data for nulls ☑ Use dotted line for nulls

☑ Draw from end to end ☐ Show shadow on line

Top Label Properties

☐ Show primary ☐ Show stack section

Negative Label Color : Color ☐ Enable

☐ Show secondary

Top Label Position

● Default ○ Positive Side ○ Negative Side

Label Alignment

Primary: Top Alignment

Data Properties Dialog

This data properties dialog contains the following options:

Column width: This specifies the ratio of the bar/column width with respect to the gap between successive bars in the chart. Each unit represents $1/10_{th}$ of the space between data points. Therefore, entering **`9`** would leave 10% of the space between data points blank, while **`10`** would eliminate all space between bars/columns. This option only pertains to two-dimensional bar, column, stack bar, stack column, high-low, HLCO, and Gantt charts. To control the column thickness in three-dimensional charts, you can use the thickness of shape slider in the navigation panel.

Show data for nulls: This option will connect lines when null data is present. For example, if you have three points and the value of point 1 is 4, point 2 is null, and point 3 is 6, then a line will be drawn from 4 to 6. This option is only available for line charts and other two-dimensional charts with lines. All other chart types will not plot null data.

Use dotted lines for nulls: You can use this option to replace the full line with a dotted line. Like the show data for nulls option, this property is only available for lines.

Draw from end to end: This option allows you to draw two-dimensional line and area charts across the entire plot area, rather than offsetting to the first and last data points on the chart.

Show shadow on line: This option allows you to use shading on two-dimensional lines. However, the line must be thicker more than one pixel.

Show primary: This option will display data top labels for the primary values in the chart.

Show stack section: This option will display individual labels for each stack section for stack bar, stack column, and stack area charts.

Negative Label Color: This option will display the top labels with a value smaller than that of the origin in a different color that can be selected using the *Color* button after enabling the feature.

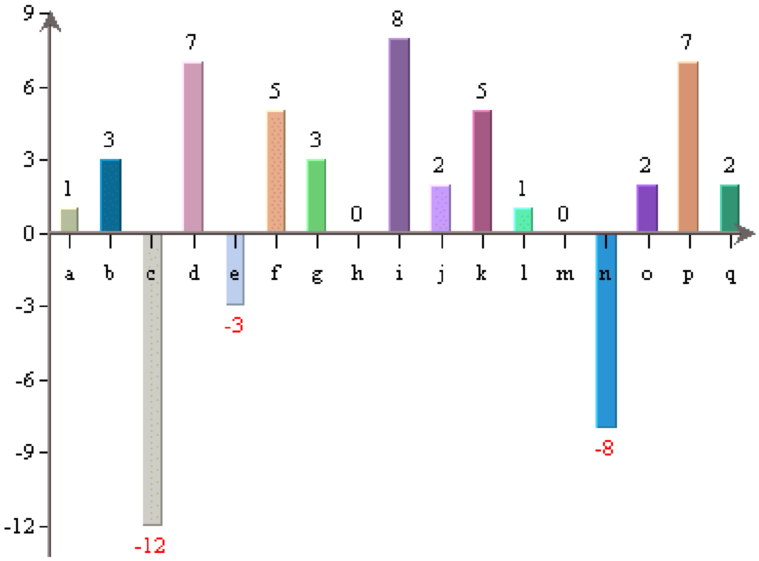


Chart with Colored Negative Top Labels

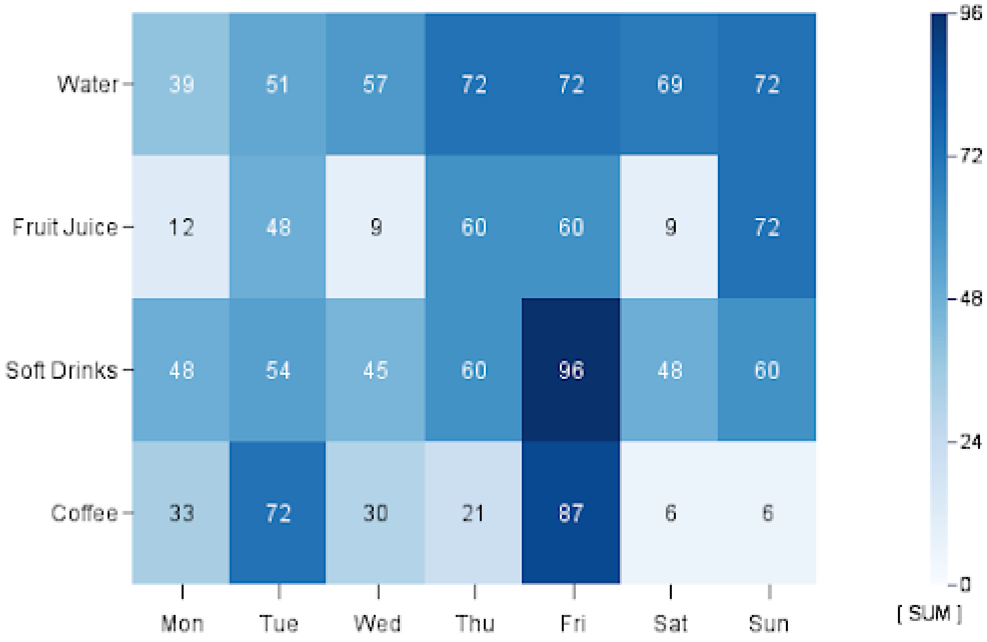
Show secondary: This option will display data top labels for the secondary values in the chart.

Top label position: This option allows you to specify where the data top labels should be drawn. By default, they are drawn above data points if they are positive and below data points if they are negative. Other options allows you to draw the labels to either positive or negative side.

Label Alignment: This option allows you to set the alignment for the data top labels. You can draw them at the top, bottom, or middle of the data points. In addition, you can select to draw the label inside the data point at the top or bottom. An additional option stack charts offers you to set the alignment for stack section labels.

3.5.8.1.1. Data Properties for Heatmap Chart

For heatmap charts, the Data Properties dialog allows users to configure the values displayed in the cells in the matrix.



Data Properties for Heatmap Chart

3.5.8.2. Date/Time Based Zooming

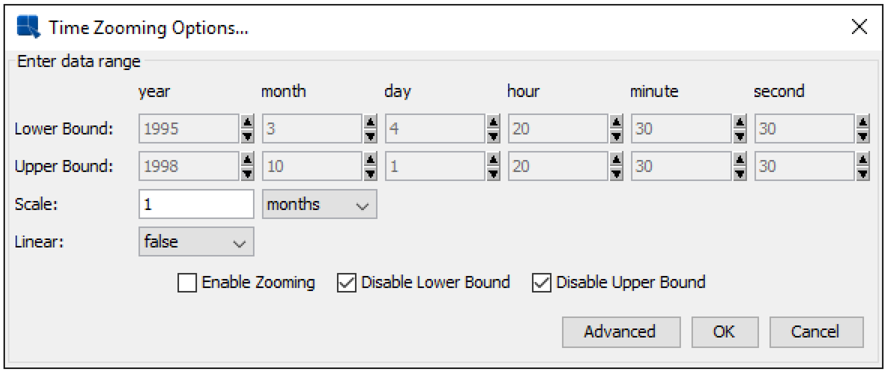
For charts displaying date or time data on the category axis, ChartDesigner provides a unique feature allowing users to perform date/time based zooming. Using this feature, you can group the category elements into user-defined intervals and aggregate the points in each group. You can also filter the data by specifying upper and lower bounds for the results.

For example, suppose your data contains daily sales volume for the past two years. Using zooming, you could aggregate the data to look at average volume per month, quarter, or year. Using the upper and lower bounds, you could narrow the range to look at weekly sales volume within a specific quarter.

Zooming is available for all chart types except scatter, surface, box, dial, polar, radar, bubble, and Gantt.

3.5.8.2.1. Adding Zooming

When you create a new chart with date, time, or timestamp data in the category axis, you can specify zooming options by selecting Format → Time Zooming Options.


Time Zooming Options...

Enter data range

	year	month	day	hour	minute	second
Lower Bound:	1995	3	4	20	30	30
Upper Bound:	1998	10	1	20	30	30

Scale: 1 months

Linear: false

□ Enable Zooming ☑ Disable Lower Bound ☑ Disable Upper Bound

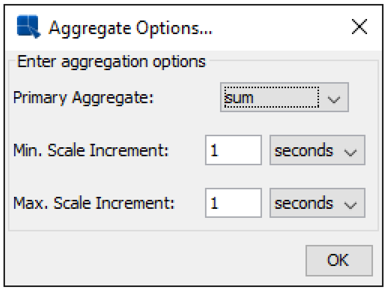
Zoom Options Dialog

This dialog allows you to specify a lower and upper bound for the data, as well as the interval by which you would like to group the data. The scale specified here must be within the maximum and minimum scale specified in the aggregation options dialog.

This dialog also allows you to preserve a linear scale for the chart. By setting the *Linear* option to true, the chart will always display points for the grouped intervals, even if there is no data associated with a particular group. For example, say again that you are measuring sales volume over a three month period - April, May, and June. If the input data has no records for May and you set the *Linear* option to true, a point will be drawn for May with a value of zero. If you set the *Linear* option to false, the data point for April will be immediately followed by June.

You can disable/enable zooming, as well as the lower and upper bound restrictions by using the checkboxes at the bottom of the dialog.

If you enable zooming (if you check *Enable Zooming* option), the dialog *Aggregate Options* will appear, prompting you to specify aggregation options for the grouped data points.



Aggregate Options...

Enter aggregation options

Primary Aggregate: sum

Min. Scale Increment: 1 seconds

Max. Scale Increment: 1 seconds

Aggregation Options Dialog

In this dialog, you can specify the Primary Aggregation, as well as the maximum and minimum scale increments that can be used when zooming the data. After you have specified your desired options, click on the *OK* button to return to zooming options.

Once you have finished specifying all the options, click on the *OK* button and the zooming will be applied to the chart.

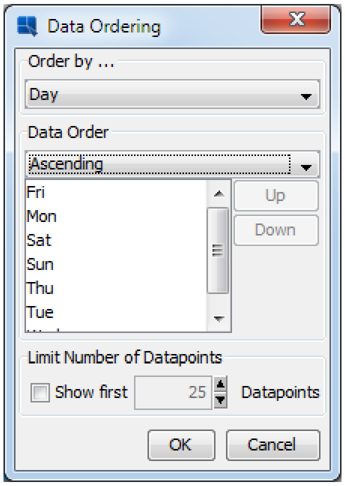
3.5.8.2.2. Zooming In Chart Viewer

When deploying charts using Chart Viewer, end users can perform dynamic zooming. To perform a time-series zoom in the Chart Viewer, **Ctrl**+**Click** on a point on the chart and drag it to another point in the chart. This will automatically zoom in based on the lower and upper bounds selected using the mouse. The aggregation is performed according to the options that were set during design time. You can undo the zoom by **Ctrl**+**Right-Click**.

The scale interval is chosen automatically, depending on the data and chosen bounds (as long as minimum 2 data points can be shown). The scale interval can also be changed in the Chart Viewer by pressing **Alt**+**Z**. This will bring up a dialog allowing the user to change the zoom settings.

3.5.8.3. Data Ordering

ChartDesigner allows you to change the order of the category and series elements. To modify the ordering, select Data → Ordering or click the *Change Data Ordering* button on the toolbar. This will bring up the following dialog:



Order by ...

Day

Data Order

Ascending

Fri
Mon
Sat
Sun
Thu
Tue

Limit Number of Datapoints

□ Show first 25 Datapoints

Data Ordering Dialog

There is an *Order By* list which contains category element, series element, and an option marked *VALUES*. You have the following options for the category and series elements:

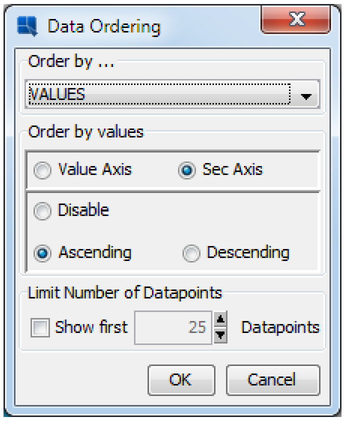
DataSource Order: This option turns the ordering off. The categories/series order will depend on the data source only and will not be altered by the EspressReport at all.

Ascending: This option will arrange the categories/series elements in ascending order. For example, if the category elements are strings, they will be arranged alphabetically.

Descending: This option will arrange the categories/series elements in descending order.

Customize: This option allows you to customize the categories/series order. To customize the order, select an item from the list of Categories/Series items and then move it upwards or downwards in the list by clicking on the *Up* or *Down* button (the buttons are inactive until you select the *Customize* option).

You can also sort the category elements based on their corresponding values. To do this, select the *VALUES* option in the data ordering dialog.



Order by ...

VALUES

Order by values

○ Value Axis ● Sec Axis

○ Disable

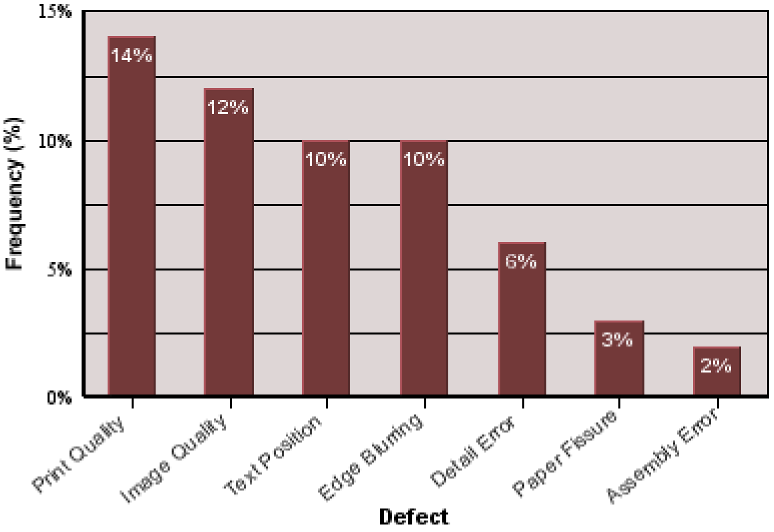
● Ascending ○ Descending

Limit Number of Datapoints

□ Show first 25 Datapoints

Value Data Ordering Dialog

If you choose the *VALUES* option, the entire dialog changes. From this dialog, you can specify to sort the category elements based on their corresponding values in the value, or secondary value axis. You can also specify whether to sort them in ascending or descending order. This type of ordering is called a *Pareto* chart and is often used in process control applications.



Pareto Chart

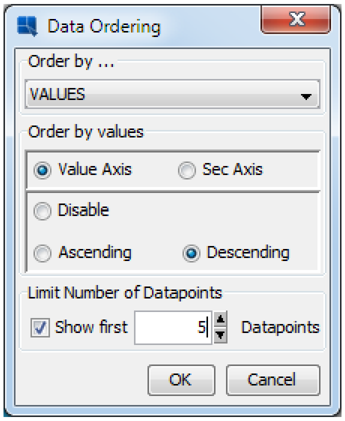
Please note that any sorting set will be re-applied if the chart is refreshed and/or if the data changes.

3.5.8.3.1. Top/Bottom N Charts

Sometimes you want to plot only a few highest or lowest values. To do that, you can use the *Top/Bottom N* function.

To enable this feature, choose Data → Ordering, or click the *Change Data Ordering* button on the toolbar.

If the chart doesn't have any data series, the *Ordering* dialog will show the *Limit Number Of DataPoints* option.



Data Ordering

Order by ...

VALUES

Order by values

● Value Axis ○ Sec Axis

○ Disable

○ Ascending ● Descending

Limit Number of Datapoints

☑ Show first 5 Datapoints

This option can only be enabled if you sort the chart by categories or values, or by the values in ascending or descending order. If you have such chart, you can enable this function by selecting the *Show first* option. Then you can specify the maximum number of items that will be shown in the chart. If the data source returns more items than you specify in this option, excessive items will not be shown in the chart as if they didn't exist.

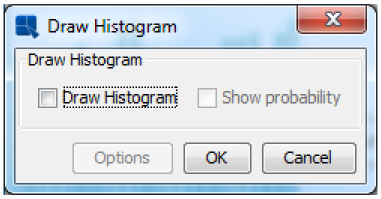
3.5.8.4. Histograms

Histogram is a useful analysis tool that allows you to track how often events occur and when and how a set of data falls into specific ranges. ChartDesigner allows you to plot histograms based on the category elements in a chart. You can plot histograms for all category data types except time-based data (date, time, or timestamp).

Histograms are calculated by counting data points or instances of each category element. For numeric categories, you can further specify upper and lower bounds, as well as the number of bins or bin width to create ranges for the frequency counts.

To create a histogram, you must start with a 2D column chart, bar chart, line chart or area chart. In the Data Mapping dialog, make sure DataSeries is set to *None* and the field you want to plot in the histogram is set in the Category axis.

Once you click *Done* in the Data Mapping dialog and the chart is shown on the canvas, select Format → Histogram Options. A dialog will appear allowing you to select a histogram plot. By default, the histogram is displayed as frequency count. You can choose to change it to display probability by checking the *Show probability* check box.

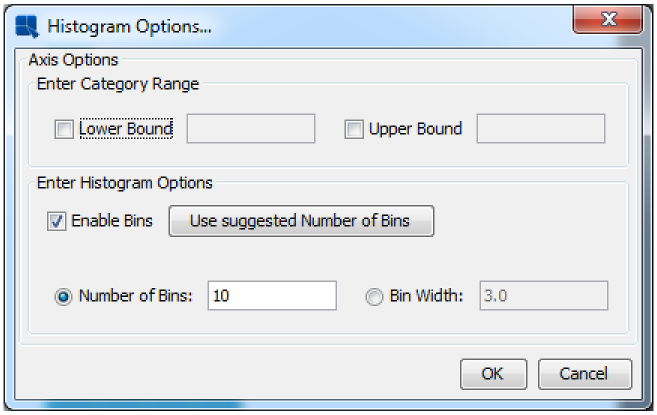

Draw Histogram

Draw Histogram

□ Draw Histogram □ Show probability

Select Histogram Dialog

If the values in the Category axis is numeric, you will see that the *Options* button is enabled. When you click Options button, another dialog will appear, allowing you to specify options for the histogram plot.


Histogram Options...

Axis Options

Enter Category Range

□ Lower Bound □ Upper Bound

Enter Histogram Options

☑ Enable Bins

● Number of Bins: 10 ○ Bin Width: 3.0

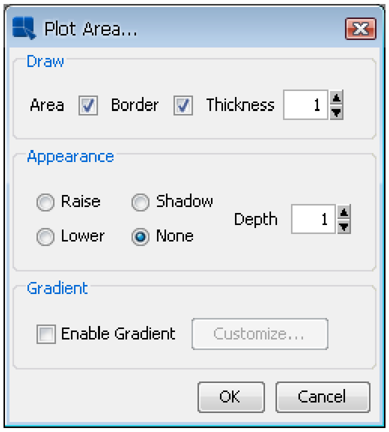
Histogram Options Dialog

From this dialog, you can set a lower or upper bound for that data being plotted. When you place bound restrictions, the histogram will not count data that falls outside of the range specified by the upper and lower bounds. If you select *Enable Bins*, you can specify the number of bins or set width of each bin. The default number of bins is 10. You can also click *Use suggested Number of Bins* if you wish to use a value calculated by the system. If *Enable Bins* is deselected, the frequency count will be performed on each Category value instead of a range of values for a bin.

If you enable scale, you can specify the number of bins or set width of each bin.

3.5.8.5. Formatting Plot Area

The plot area is the plane on which the data points are drawn for two-dimensional charts. You can customize the appearance of the plot area by selecting Format → Plot Area. Assuming the current chart is a two-dimensional chart, the following dialog will appear.



Draw

Area ☑ Border ☑ Thickness 1

Appearance

○ Raise ○ Shadow Depth 1
○ Lower ● None

Gradient

☐ Enable Gradient

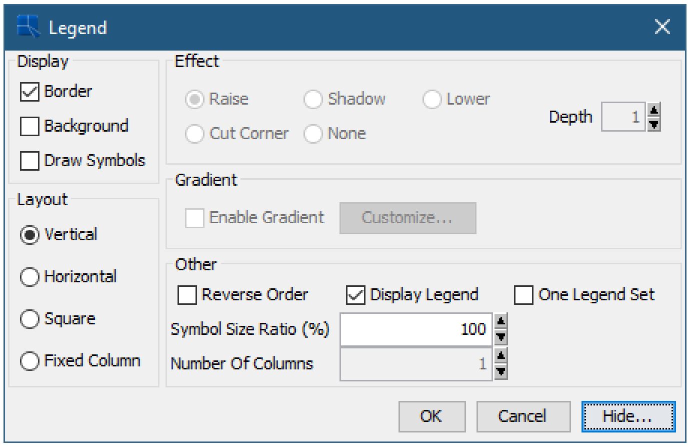
Plot Area Dialog

This dialog allows you to draw a border around the plot area, or fill it with a background color. If you fill the area, you can also specify certain 3D effects like raising, lowering or shadow.

On this dialog, you can also set up gradient background for the plot area. The gradient settings are the same as in the *Rendering options* described in the Section 3.5.1.3 - Format Menu.

3.5.8.6. Formatting Chart Legend

You can control and modify the display of the chart legend either by selecting Format → Legend, or by clicking on the *Format Legend* button on the toolbar, or by selecting *Legend properties* from legend pop-up menu (which pops up when you right-click on the legend). This will bring up the following dialog, allowing you to customize the legend properties.



Display

☑ Border
☐ Background
☐ Draw Symbols

Layout

● Vertical
○ Horizontal
○ Square
○ Fixed Column

Effect

● Raise ○ Shadow ○ Lower Depth 1
○ Cut Corner ○ None

Gradient

☐ Enable Gradient

Other

☐ Reverse Order ☑ Display Legend ☐ One Legend Set

Symbol Size Ratio (%) 100

Number Of Columns 1

Format Legend Dialog

The dialog contains the following options:

Display: These options allow you to turn on or off the legend border and background. This also allows you to display the point symbols instead of lines or blocks in the legend.

Effect: This allows you to add a 3D effect to the legend. You can raise it, lower it, or draw a shadow. In addition to the 3D effects, you can also display the legend with cut corners.

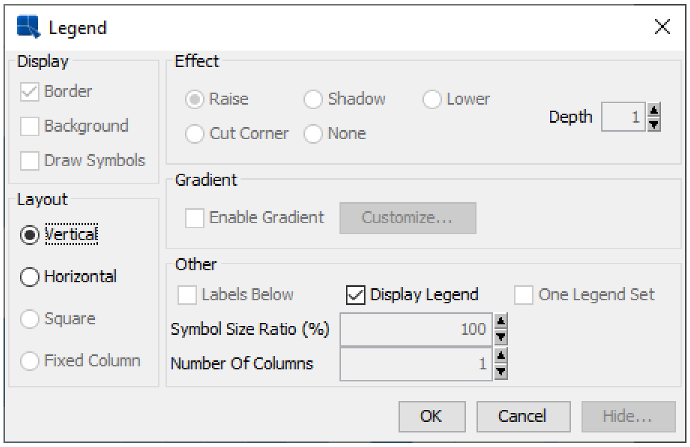
Layout: This allows you to change the legend from vertical, horizontal, square, or fixed column layout.

Gradient: Allows you to configure gradient for the legend background. The gradient settings are the same as in the *Rendering options* described in the Section 3.5.1.3 - Format Menu.

Other: This allows you to choose whether or not to display the legend, or to draw the legend in reverse order. You can set the fixed number of columns in legend in the *Number of columns* field. This field will be active only if you choose the *Fixed columns* layout option in the *Layout* section. You can set trend/control lines and chart data legend drawn as *One Legend Set*. You can also change the size of the symbols in the legend.

Additionally, you can remove specific category/series elements from the legend by clicking on the *Hide* button. This will bring up a list of the legend items, where you can select which elements you would like to hide.

The legend for heatmap charts differs from those of other charts, and some legend options in the *Legend* dialog are not applicable to heatmap charts. By default, the legend for heatmap charts is displayed vertically.



Legend

Display
☑ Border
□ Background
□ Draw Symbols

Layout
● Vertical
○ Horizontal
○ Square
○ Fixed Column

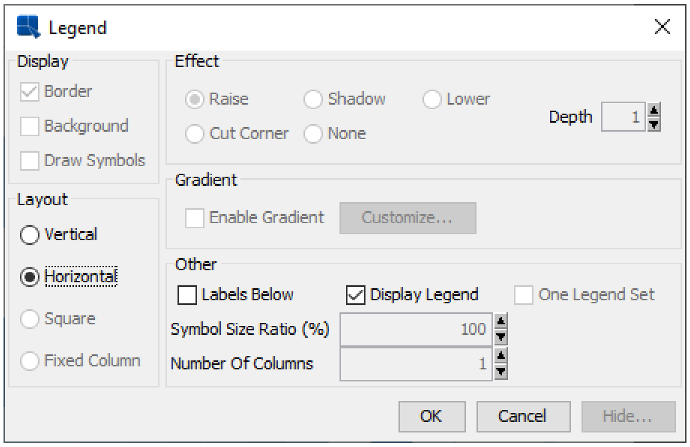
Effect
● Raise ○ Shadow ○ Lower
○ Cut Corner ○ None
Depth 1

Gradient
□ Enable Gradient

Other
□ Labels Below ☑ Display Legend □ One Legend Set
Symbol Size Ratio (%) 100
Number Of Columns 1

Heatmap Vertical Legend

In the horizontal layout, you can choose to display tickers and labels below the legend by selecting the *Labels Below* option.



Legend

Display
☑ Border
□ Background
□ Draw Symbols

Layout
○ Vertical
● Horizontal
○ Square
○ Fixed Column

Effect
● Raise ○ Shadow ○ Lower
○ Cut Corner ○ None
Depth 1

Gradient
□ Enable Gradient

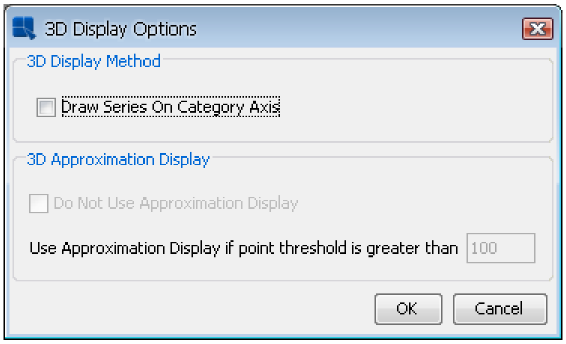
Other
□ Labels Below ☑ Display Legend □ One Legend Set
Symbol Size Ratio (%) 100
Number Of Columns 1

Heatmap Horizontal Legend

3.5.8.7. 3D Display Options

ChartDesigner renders three-dimensional charts in true 3D, allowing light source modification, panning, zooming, and rotation. However, 3D rendering can be very memory and CPU intensive. When charts have a lot of data points (like 3D scatter and surface charts), it's possible to run out of memory when generating the chart. To solve this problem, a rendering approximation feature is provided. Using this algorithm the chart is not rendered perfectly, but it's usually acceptable when a lot of points have to be shown.

By default, approximation is turned on at a threshold value of 100 points. This means that if a 3D chart has more than 100 data points, the approximation will be used. You can turn this feature off, or change the threshold value by selecting Format → 3D Display Options. This will bring up the following dialog.



3D Display Options

3D Display Method

□ Draw Series On Category Axis

3D Approximation Display

□ Do Not Use Approximation Display

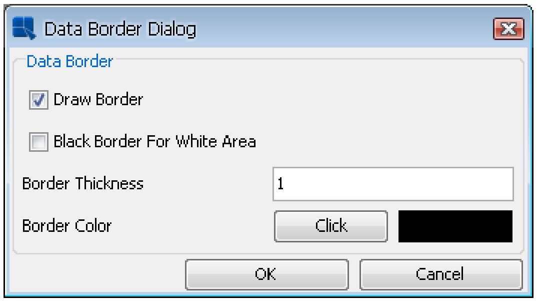
Use Approximation Display if point threshold is greater than 100

3D Display Options Dialog

The two options for 3D approximation allows you to turn on/off the approximation and set the threshold value. The other option in this dialog allows you to draw the series in-line (the same option is in navigation panel).

3.5.8.8. Data Border

For column, bar, stack column, stack bar and HLCO charts, ChartDesigner allows you to configure a border around the columns. To set the border option, select Format → Data Border. This will bring up a dialog allowing you to set border options.



Data Border Dialog

Data Border

☑ Draw Border

□ Black Border For White Area

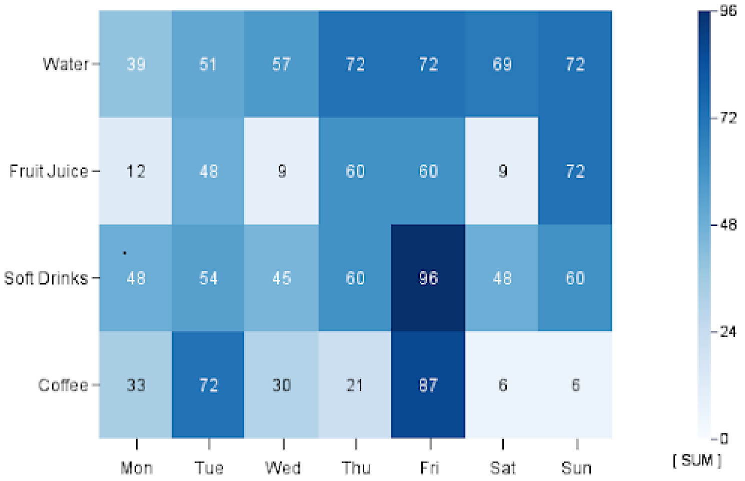
Border Thickness: 1

Border Color: ■

IData Border Dialog

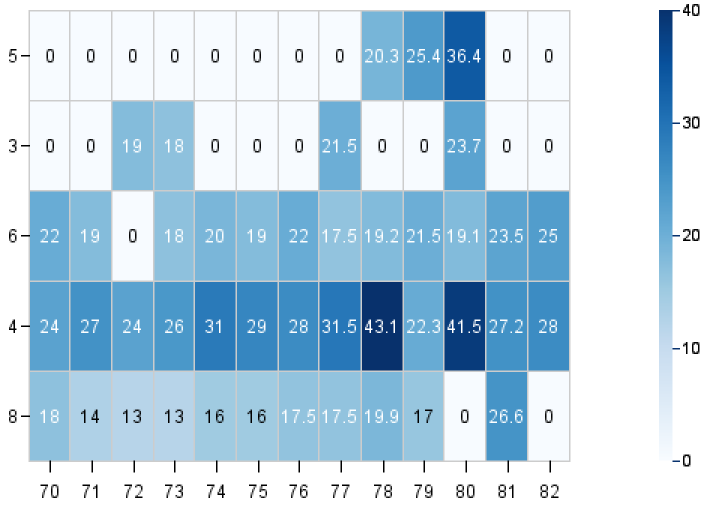
The first option allows you to turn on/off the data border. The second option allows you to set a black border for any white areas in the chart. Please note that the border is black only if the first option is unchecked and will only appear around white areas in the chart. The third and fourth options allows you to set border thickness and border color. If you click on the *Click* button, a new dialog will appear allowing you to select or enter a new color.

For heatmap charts, you can configure a border around the data rectangles. This is particularly useful when there are many light-colored cells.



Heatmap Data Border

The following is an example heatmap chart with border thickness set to one.

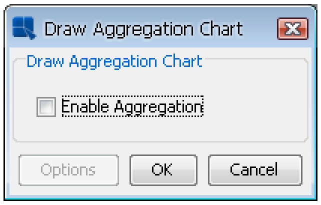


Heatmap chart with light gray colored border

3.5.8.9. Aggregation

ChartDesigner allows you to aggregate data if there is more than one data point associated to a given category (and its series and/or stack, if a series and/or stack is present). This allows for a broader look at the data rather than just a single data point (out of many).

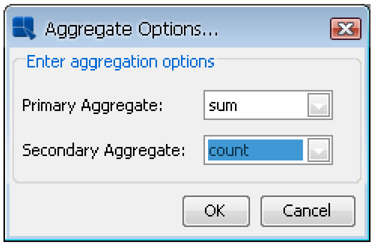
To aggregate the data, select Format → Aggregation Options. A dialog will appear allowing you to enable aggregation.



☐ Enable Aggregation

Select Aggregation Dialog

When you select *Enable Aggregation*, a second dialog box will appear, asking you which type of aggregation should be applied. You can choose from minimum, maximum, average, sum, count, first, last, sumsquare, variance, stddev, and countdistinct for the aggregates. You can specify a primary aggregate (aggregate applied to the column mapped to the primary axis) as well a secondary aggregate (aggregate applied to the column mapped to the secondary axis), if a secondary axis exists.


Aggregate Options...

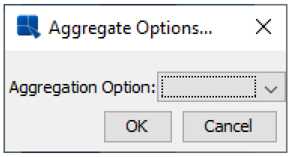
Enter aggregation options

Primary Aggregate: sum

Secondary Aggregate: count

Aggregate Options Dialog

For heatmap charts, the aggregation option can be used when there are multiple data values at a single (x, y) point.


Aggregate Options...

Aggregation Option:

Aggregate Options

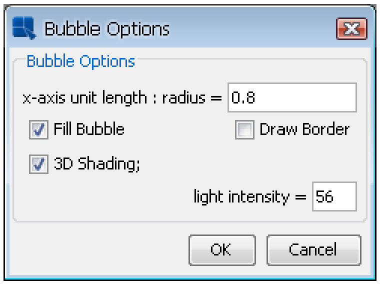
By default, no aggregation is applied to heatmap charts, and the first data value is always selected if there are multiple values at the same (x, y) point.

3.5.9. Chart-Specific Options

There are a number of formatting options that are unique to certain chart types. These options can be modified by selecting Format → Chart Options, or clicking the *Chart Options* button on the toolbar. This will bring up a dialog that varies depending on the type of the current chart. Some chart types have no additional options.

3.5.9.1. Bubble Charts

For bubble charts, the following dialog is displayed:


Bubble Options

Bubble Options

x-axis unit length : radius = 0.8

☑ Fill Bubble ☐ Draw Border

☑ 3D Shading;

light intensity = 56

Bubble Options Dialog

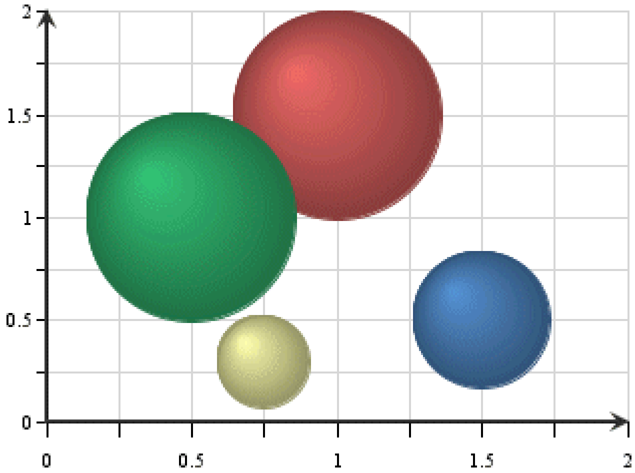
The following options are available for bubble charts:

x-axis unit length: radius This option specifies the ratio of X-axis unit length to the radius of the bubble.

Fill Bubble: This option allows you to specify whether or not to fill the bubble area.

Draw Border: This option allows you to specify whether or not to draw a border around the bubbles.

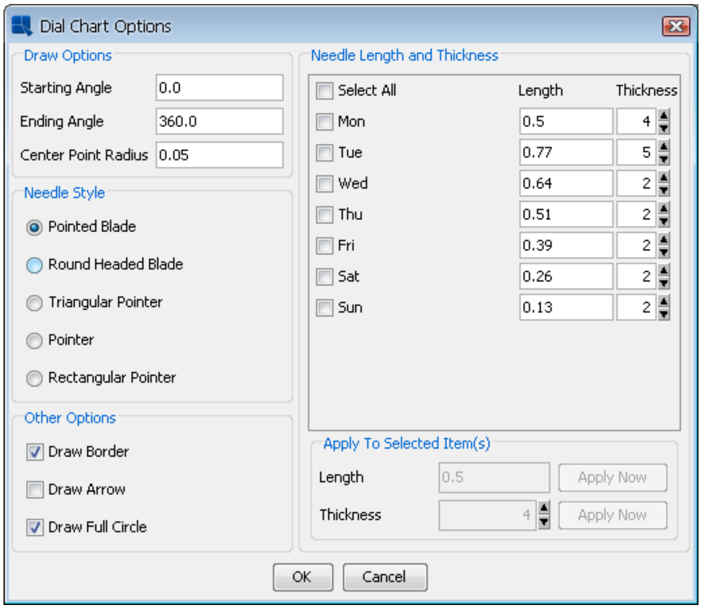
3D Shading: This option allows you to add 3D shading to the bubbles. You can also specify the light intensity for the shading.



Bubble Chart with 3D Shading

3.5.9.2. Dial Charts

For dial charts, the following dialog is displayed.



Draw Options

Starting Angle: 0.0

Ending Angle: 360.0

Center Point Radius: 0.05

Needle Style

● Pointed Blade
○ Round Headed Blade
○ Triangular Pointer
○ Pointer
○ Rectangular Pointer

Other Options

☑ Draw Border
☐ Draw Arrow
☑ Draw Full Circle

Needle Length and Thickness

☐ Select All	Length	Thickness
☐ Mon	0.5	4
☐ Tue	0.77	5
☐ Wed	0.64	2
☐ Thu	0.51	2
☐ Fri	0.39	2
☐ Sat	0.26	2
☐ Sun	0.13	2

Apply To Selected Item(s)

Length: 0.5 [Apply Now]

Thickness: 4 [Apply Now]

[OK] [Cancel]

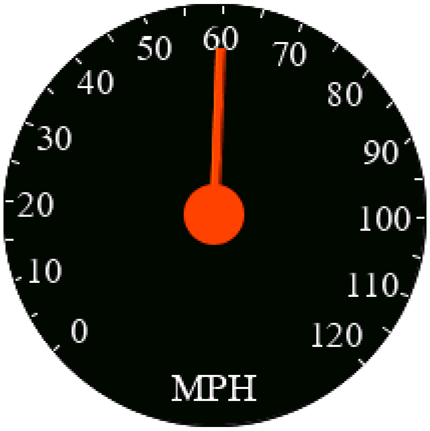
Dial Options Dialog

The following options are available for dial charts:

Starting Angle: This option specifies the angle where the first axis label is to be set. This property also determines where the border and dial area will start if the *Draw Full Circle* option is unchecked. The angle is represented in degrees and is 0 by default. Assuming the dial chart is a clock face, 0 degrees is 12 o'clock.

Ending Angle: This option specifies the angle where the last axis label should be set. This property also determines where the border and dial area will end if the *Draw Full Circle* option is unchecked. The angle is represented in degrees and is 360 by default. Hence by default the labels (and data points) encompass the entire circumference of the dial.

Center Point Radius: This option specifies the radius for an inner circle, which starts from the center of the dial chart. The radius is specified as a ratio to the radius of the dial plot. Hence, a value of 1 will make the inner circle encompass the entire dial. If the *Draw Full Circle* option is unchecked, only the portion of the center point that is within the starting and ending angles will be shown.



Dial Chart with Center Point Radius

Needle Style: This option specifies the type of needle to draw.

The Pointed Blade is a smooth line with a thick base. It becomes slightly thinner as it extends outwards, but finishes with a very sharp pointed tip.

The Round Headed Blade is similar to the Pointed Blade except for a round tip.

The Triangular Pointer is sharp and resembles a very thin triangle.

The Pointer is a step ladder pointer with three segments.

The Rectangular Pointer (as shown above on the picture) is a simple straight needle.

Default needle is Pointed Blade.

Draw Border: This option specifies whether or not to draw a border around the dial.

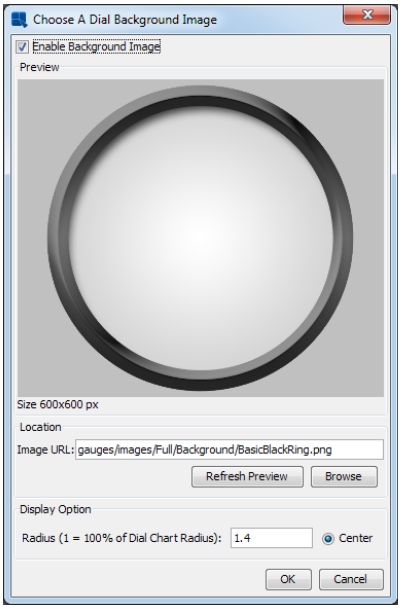
Draw Arrow: This option specifies whether or not to draw arrowheads at the end of the dial hands.

Draw Full Circle: This option specifies whether to draw the dial as a complete circle (360 degrees) or only draw the portion of the circle determined by the starting and ending angles.

Needle Length and Thickness: This option specifies the length and thickness of the needle. The needle length is measured from the center of the dial. The range is from 0 (center of the dial) to 1 (the end of the dial). The thickness determines the width of the needle, larger values results in a wider pointer. When creating a chart, the needle length is randomly generated. Default thickness is 2. Each category element is represented by a different needle. You can either change properties individually or change multiple categories at once. To change the property of each needle individually, directly change the values to the right of the category. To make changes to multiple needles, check each category or check the *Select All* option, set the properties in the lower right corner and then click on the *Apply Now* button for each property changed. This will modify all checked categories to new values.

3.5.9.2.1. Gauge Images

Dial charts have an additional option to display a foreground or background image for the dial plot area. To add a foreground or background image, select Insert → Dial Foreground... or Insert → Dial Background.... These options are only enabled for dial charts.



Choose A Dial Background Image

☑ Enable Background Image

Preview

Size 600x600 px

Location

Image URL: gauges/images/Full/Background/BasicBlackRing.png

Display Option

Radius (1 = 100% of Dial Chart Radius): 1.4 ● Center

Dial Chart Background Image Dialog

Selecting an image works in the same way as the background image dialog, see Section 3.5.5.1.1 - Background Images. In the dial chart image dialog, there is also an option to specify the radius of the image. Specifying 1 for the radius will make the image the same width and height as the plot area. Increasing or decreasing this value will enlarge or shrink the images.

3.5.9.3. Overlay Charts

For overlay charts, the following dialog is displayed:

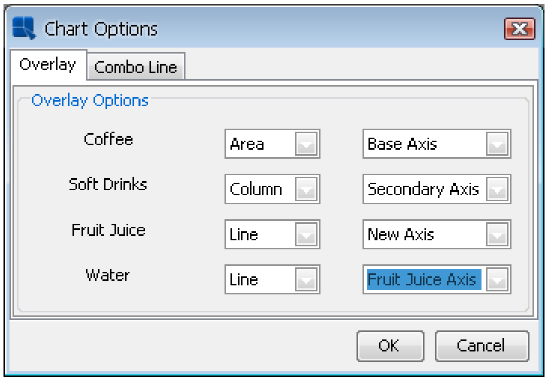


Chart Options

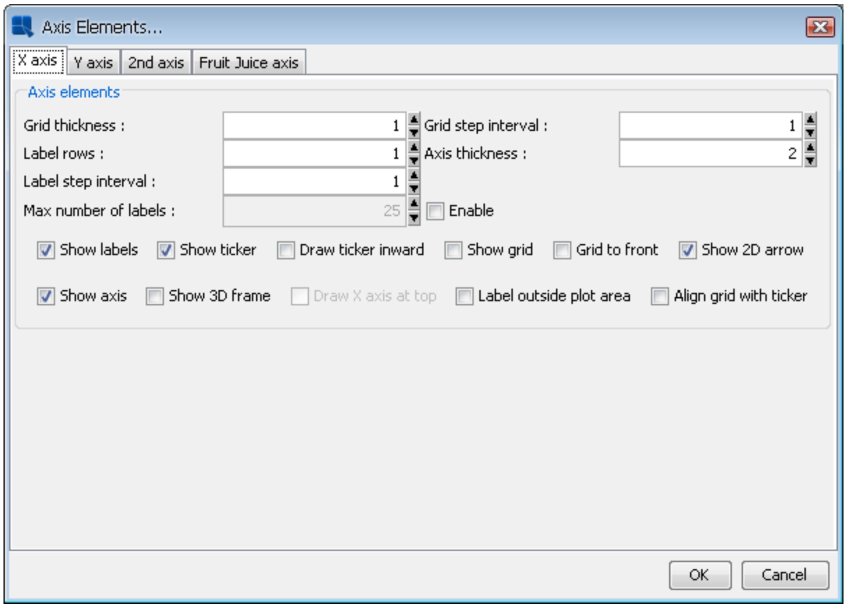
Overlay Options

Coffee	Area	Base Axis
Soft Drinks	Column	Secondary Axis
Fruit Juice	Line	New Axis
Water	Line	Fruit Juice Axis

Overlay Options Dialog

From this dialog, you can specify which chart type you would like to use for each element of the data series. Available chart types for the series elements in an overlay chart are column, area, and line. You can also choose not to display certain series elements.

From this dialog, you can also specify which axis you would like to use to plot a series element. You can place elements on the primary or secondary axes, or you can create new value axes for the series elements. To create a new value axis, select *New Axis* from the drop-down menu. Once you specify to use a new axis, a new option will be added to the drop-down menus for the other data series elements, allowing them to be drawn on the same axis that you previously specified. Using a variety of axes allows you to precisely tune the scale that the different series elements use. Each of these axes will have its own tab in the Axis Elements window, where you can change the label step interval for each axis independently of the others.

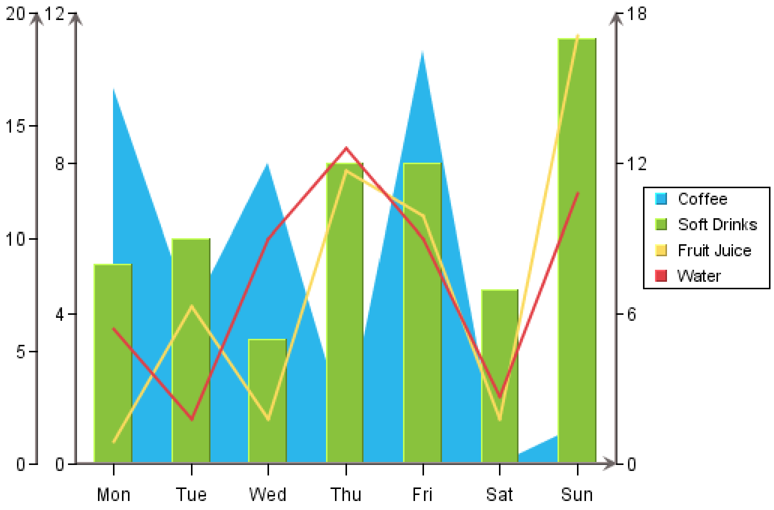


Grid thickness : 1
Label rows : 1
Label step interval : 1
Max number of labels : 25
Grid step interval : 1
Axis thickness : 2
☐ Enable

☑ Show labels ☑ Show ticker ☐ Draw ticker inward ☐ Show grid ☐ Grid to front ☑ Show 2D arrow

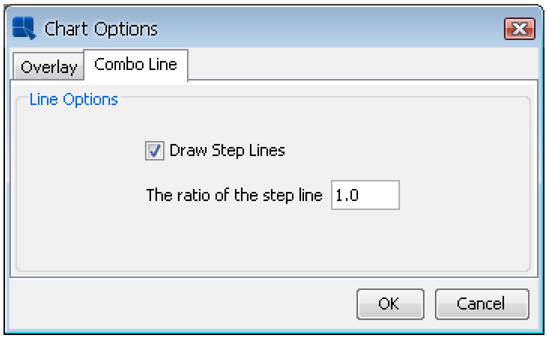
☑ Show axis ☐ Show 3D frame ☐ Draw X axis at top ☐ Label outside plot area ☐ Align grid with ticker

Axis Options Dialog for Multiple Value Axes



Overlay Chart with Multiple Value Axes

If one of the chart types used by the overlay chart is a line, you can specify whether or not to draw the line as a step line using the *Combo Line* tab. For more about step lines see Section 3.5.9.5 - Line Charts:



Line Options

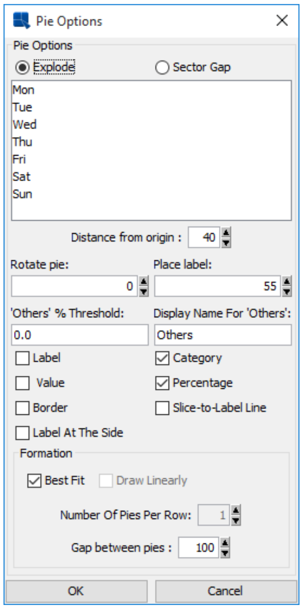
☑ Draw Step Lines

The ratio of the step line 1.0

Combo Line Options for Overlay Charts

3.5.9.4. Pie Charts

For pie charts, the following dialog is displayed. (Note that different options will appear/disappear depending on whether the chart has a series, and if it's a 2D or 3D chart.) Here are the options available for a 2D chart with series:



Pie Options

● Explode ○ Sector Gap

Mon
Tue
Wed
Thu
Fri
Sat
Sun

Distance from origin : 40

Rotate pie: 0

Place label: 55

'Others' % Threshold: 0.0

Display Name For 'Others': Others

□ Label ☑ Category

□ Value ☑ Percentage

□ Border □ Slice-to-Label Line

□ Label At The Side

Formation

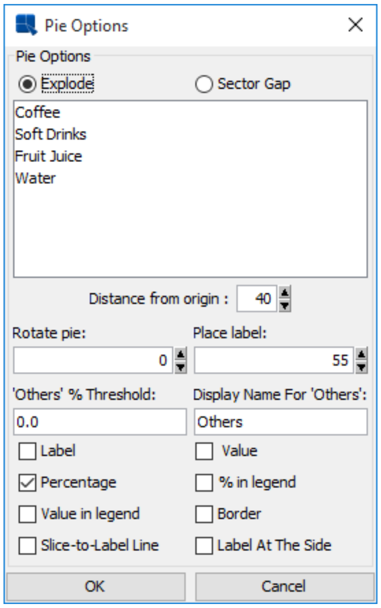
☑ Best Fit □ Draw Linearly

Number Of Pies Per Row: 1

Gap between pies : 100

Pie Options Dialog (With Series)

Here are the options available for a 2D chart without series. Notice that the formation options are removed and the *% in legend* and *Value in legend* options are added:



Pie Options

Pie Options

● Explode ○ Sector Gap

Coffee
Soft Drinks
Fruit Juice
Water

Distance from origin : 40

Rotate pie: 0

Place label: 55

'Others' % Threshold: 0.0

Display Name For 'Others': Others

□ Label □ Value

☑ Percentage □ % in legend

□ Value in legend □ Border

□ Slice-to-Label Line □ Label At The Side

Pie Options Dialog (Without Series)

The following options are available for pie charts:

Explode: This option allows you to pick one or more category/series elements whose sections are to be drawn at a certain distance away from the center of the pie.

Sector Gap: This option allows you to pick one or more category/series elements whose sections are drawn at a certain distance away from the center of the pie and still maintain the same distance between pie slices and a circular boundary.

Distance from origin: This option allows you to specify how far the exploded/sector gap sections are to be drawn away from the center. This number, represented as a percentage of the radius, indicates the distance between the center and the tip of the pie slice to be exported.

Rotate pie: This option allows you to specify the number of degrees that the chart should be rotated in a clockwise direction. Available values are between 0 and 360

Place label: This option indicates the distance of the labels from the center of the pie. The position of an individual label can also be adjusted by dragging the text.

“Others” % Threshold: This feature is useful for pie charts that have a large number of small categories. Rather than draw a slice for each category, users can select a threshold value. Any category whose percentage of the value column is less than the threshold value will be lumped into an “Others” slice.

Display Name for “Others”: This option allows you to set the display name for the “Others” slice that is created for categories that fall below the supplied threshold value. This label will appear in the legend, and/or for the slice label.

Label: This option determines whether a category/series label should be drawn for each pie slice. By default, these only appear as legend items. Note that the label will not appear if the data for the slice is 0 or null.

Value: This option allows you to specify whether to display the actual value of each pie slice. Note that the value will not appear if the data for the slice is 0 or null.

Category: This option allows you to specify whether to display the category of each pie slice.

Percentage: This option allows you to display the percentage for each pie slice. The percentages are calculated by dividing the value of each section by the sum of all the values. Note that the percentage will not appear if the data for the slice is 0 or null.

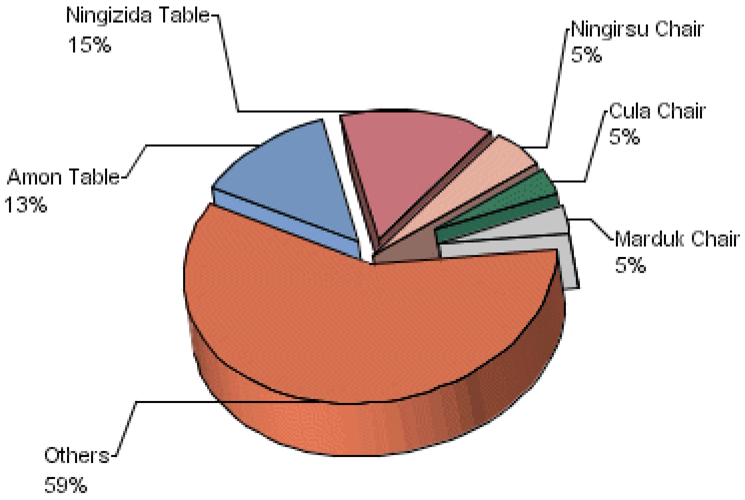
% in legend: This option allows you to display the percentage represented by each slice in the pie in the legend. This can be a preferable presentation if the pie slices become too thin. This option is only available if the pie chart does not have a data series.

Value in legend: This option allows you to display the value represented by each slice in the pie in the legend. This can be a preferable presentation if the pie slices become too thin. This option is only available if the pie chart does not have a data series.

Border: This option specifies whether to draw a border around each pie slice. This option is only available for two-dimensional pie charts. For three-dimensional pies, you can use the border drawing option on the navigation panel. Note that the border will not appear if the data for the slice is 0 or null.

Slice-to-Label Line: This option will draw a line from any label(s) to it's corresponding pie slice. Note that the slice-to-label line will not appear if the data for the slice is 0 or null.

Label at the Side: This option will place labels for the pie chart away from the plot around the outside of the chart. When used with the *Slick-to-Label Line* option, it gives users a way to display the pie labels for charts with many small categories without any text overlapping. Note that the label will not appear if the data for the slice is 0 or null.



Pie Chart with Side Labels and Lines

Best Fit: This option will arrange multiple pies in best configuration to fit the chart canvas. It's only available for pies with data series.

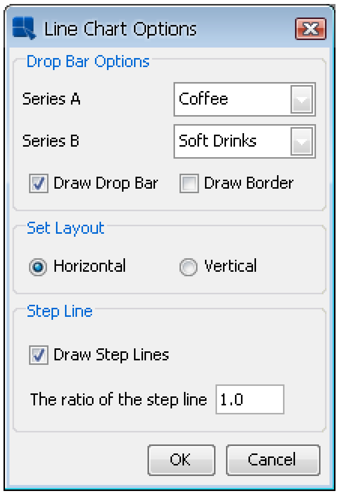
Draw Linearly: This option will arrange multiple pies in a straight horizontal line. It's only available for pies with data series.

Number of Pies Per Row: This option allows you to create a custom arrangement of multiple pies, by specifying the number of pies to draw in each row of the arrangement.

Gap between pies: This option allows you to specify the gap between the multiple pies. The number is a multiple of the pie radius, so the gap will adjust with the size of the chart plot.

3.5.9.5. Line Charts:

For two-dimensional line charts, one of two different dialogs will be displayed depending on whether the chart has a data series or not. If the chart has a data series then the following dialog is displayed.



Line Chart Options

Drop Bar Options

Series A: Coffee

Series B: Soft Drinks

☑ Draw Drop Bar ☐ Draw Border

Set Layout

● Horizontal ○ Vertical

Step Line

☑ Draw Step Lines

The ratio of the step line: 1.0

Line Options Dialog (with series)

Line charts with data series have a specific option that allows you to draw drop bars between two series elements. The dialog options are as follows:

Series A: This option specifies the first series element for the drop bar.

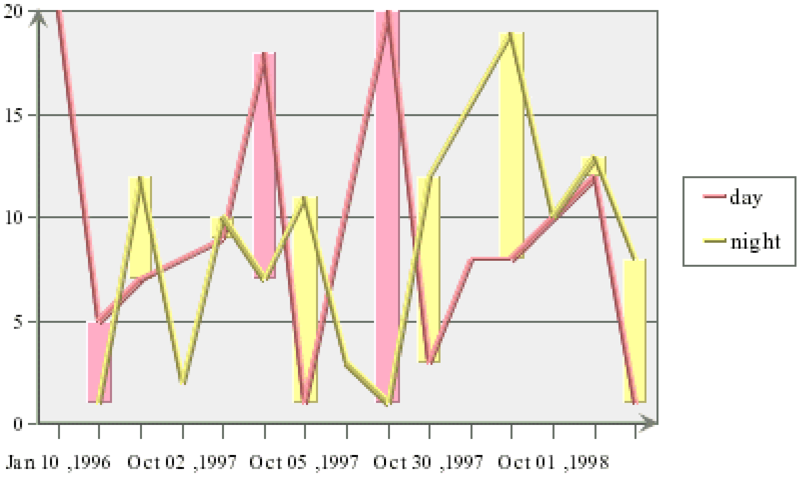
Series B: This option specifies the second series element for the drop bar.

Draw Drop Bar: This option specifies whether or not to draw drop bars.

Draw Border: This option specifies whether or not to draw a border around drop bars.

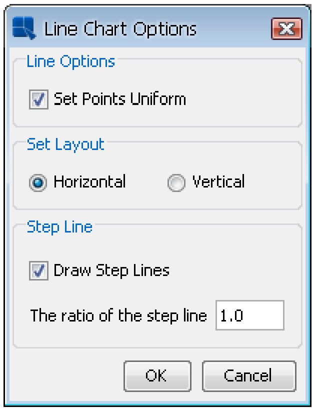
Set Layout: This option specifies whether to draw a line chart in vertical or horizontal orientation.

Step Line: This option allows you to draw line chart as a step line. You can also specify the step line ratio to use.



Line Chart with Drop Bars

Note that the color of the drop bar will vary depending on which series has the higher value for a given point. If the line chart does not have a series, then the following dialog will appear.



Line Chart Options

Line Options

☑ Set Points Uniform

Set Layout

● Horizontal ○ Vertical

Step Line

☑ Draw Step Lines

The ratio of the step line 1.0

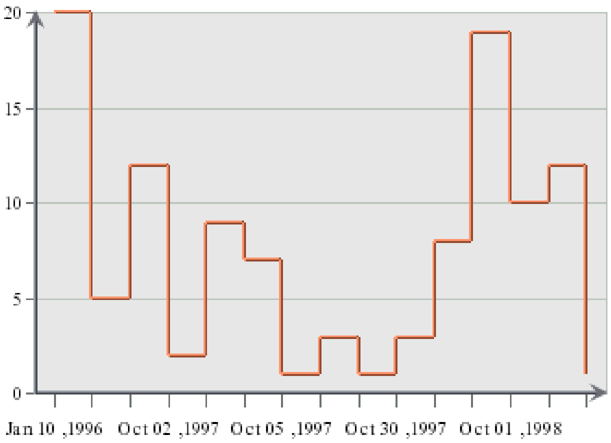
Line Options Dialog (without series)

The dialog options are as follows:

Set Points Uniform: This option specifies whether the point shapes and colors are uniform or not. Un-checking this option allows you to set multiple colors and point shapes for the data points. The points can be customized in the line and point dialog.

Set Layout: This option specifies whether to draw the line chart in vertical or horizontal orientation.

Step Line: This option allows you to draw the line chart as a step line. You can also specify the step line ratio to use.



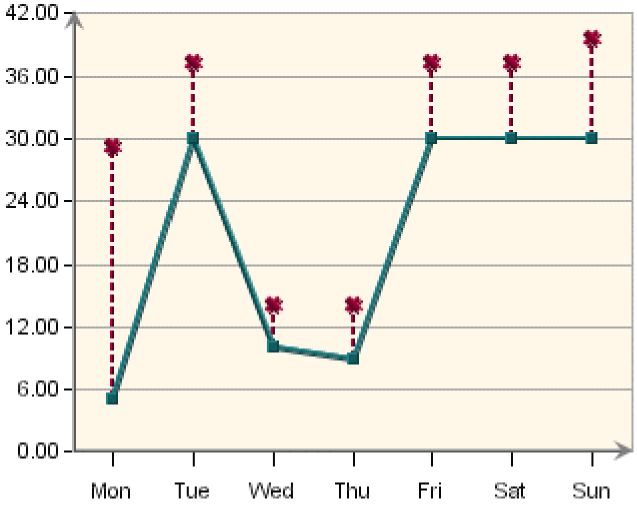
Line Chart with Step Lines

The step line ratio allows you to specify how far between points the horizontal portion of the step line should be drawn. A ratio of 1 draws the line horizontally to the next point on the chart and then draws vertically to that point, while a ratio of 0.5 draws the horizontal portion halfway between the two points. A ratio of 0 will result in the vertical portion of the line drawn first and then connect to the next point horizontally. Values for the ration are between 0 and 1.

There are no additional options for three-dimensional line charts.

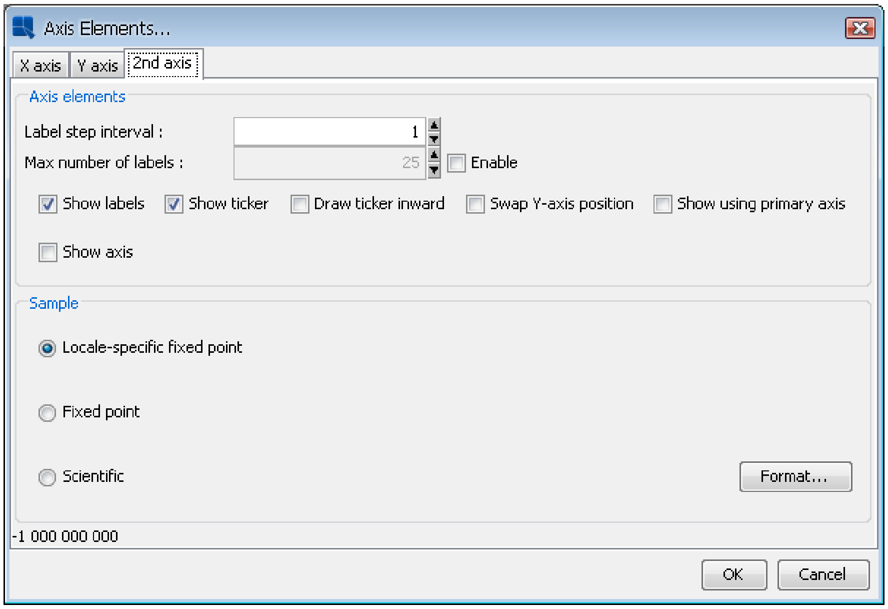
3.5.9.5.1. Double Value Line Charts

ChartDesigner contains a special option for line charts that allows you to have two values shown for the same line. Here the secondary axis is used to plot the second value (as in a line-line combination) and then combined with the line on the primary axis.



Double Value Line Chart

To create a double value line chart, design a line-line combination chart (a line chart with primary and secondary values). Then select Format → Axis Elements. This will bring up the axis elements dialog.



Label step interval : 1

Max number of labels : 25 □ Enable

☑ Show labels ☑ Show ticker □ Draw ticker inward □ Swap Y-axis position □ Show using primary axis

□ Show axis

● Locale-specific fixed point

○ Fixed point

○ Scientific

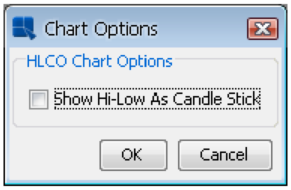
-1 000 000 000

Axis Elements Dialog for Line-Line Combination Charts

Under the *2nd Axis* tab, there is a checkbox marked *Show using primary axis*. Check this box and click on the *OK* button. Your chart will now be drawn as a double value line chart.

3.5.9.6. HLCO Charts

For HLCO charts, the following dialog is displayed:

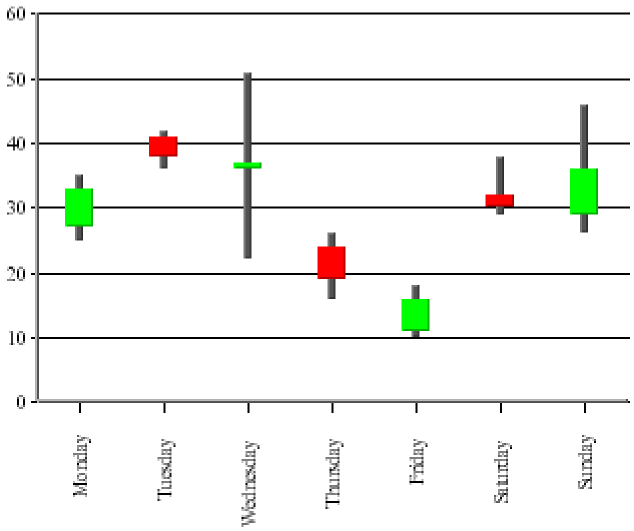


HLCO Chart Options

☐ Show Hi-Low As Candle Stick

HLCO Options Dialog

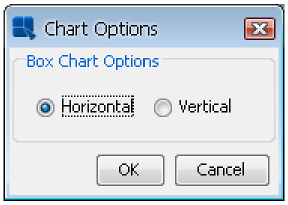
The *Show Hi-Low As Candle Stick* option will turn the HLCO chart into a candle representation. A candle HLCO chart blends high, low, close, and open data into a single object that resembles a candlestick.



HLCO Candlestick Chart

3.5.9.7. Box Charts

For box charts, the following dialog is displayed:



Box Chart Options

● Horizontal ○ Vertical

Box Options Dialog

This dialog allows you to specify whether to display the box chart in a horizontal or vertical orientation.

3.5.9.8. Stack Area Charts

For stack area charts, the following dialog is displayed:

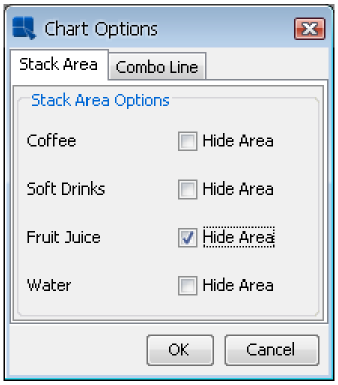


Chart Options

Stack Area | Combo Line

Stack Area Options

Coffee ☐ Hide Area

Soft Drinks ☐ Hide Area

Fruit Juice ☑ Hide Area

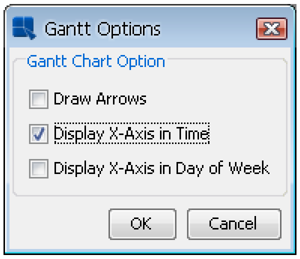
Water ☐ Hide Area

Stack Area Options Dialog

You can choose to hide any of the stacks in the chart by clicking on the corresponding check box. The *Combo Line* tab allows you to specify step lines if the chart is a line stack area combination. There are no additional options for three-dimensional stack area charts.

3.5.9.9. Gantt Charts

For Gantt charts, the following dialog is displayed:



Gantt Options

Gantt Chart Option

☐ Draw Arrows

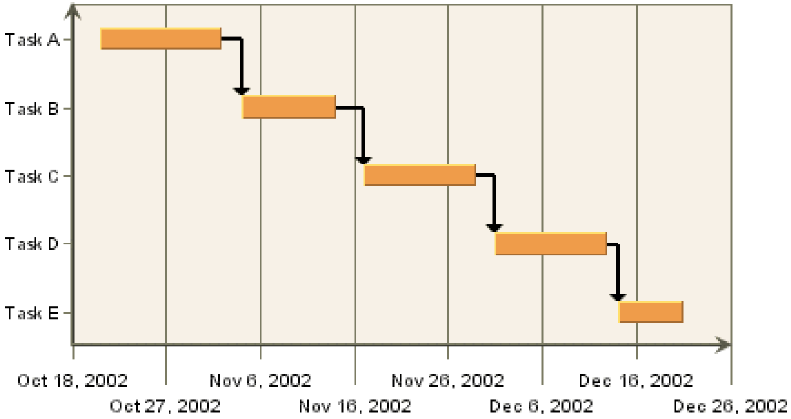
☑ Display X-Axis in Time

☐ Display X-Axis in Day of Week

Gantt Options Dialog

The following options are available for Gantt charts:

Draw Arrows: This option will draw connecting arrows between category elements of the Gantt chart. This allows you to illustrate a sequence between scheduled events. The arrows are drawn in the order the category elements appear in the data source.



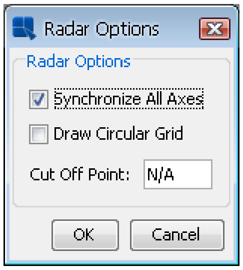
Gantt Chart with Arrows

Display X-Axis in Time: This option shows the ticker labels as time values instead of numeric values for the X-axis.

Display X-Axis in Day of Week: This option shows the ticker labels as days of the week with the date for each Sunday shown as well.

3.5.9.10. Radar Charts

For radar charts, the following dialog is displayed:



☑ Synchronize All Axes

□ Draw Circular Grid

Cut Off Point: N/A

Radar Options Dialog

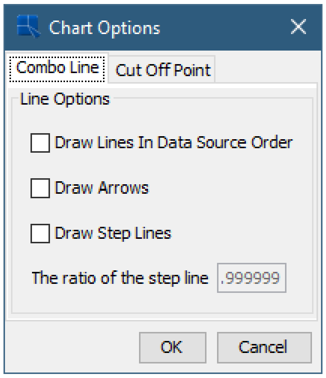
By default, the scale is same for all axes in the radar chart. Unchecking the *Synchronize All Axes* option will allow each axis in the radar chart to be scaled independently. You can select to use auto-scaling for each axis, or you can set the scales manually by invoking the axis scale dialog.

The second option allows you to set how the grid is drawn for the radar chart. By default, if the grid is enabled, it is drawn in straight lines that connect the tickers on each axis. Enabling the *Draw Circular Grid* option will draw the grid in a circle, similar to the polar chart grid.

The third option allows you to specify a cut-off point for the data points (areas) in the radar chart. You can enter a maximum value that should be shown in the chart. The areas bounded by the data points will not be drawn beyond the specified cut-off point.

3.5.9.11. Scatter Charts

For scatter charts, the following dialog is displayed:



□ Draw Lines In Data Source Order

□ Draw Arrows

□ Draw Step Lines

The ratio of the step line .999999

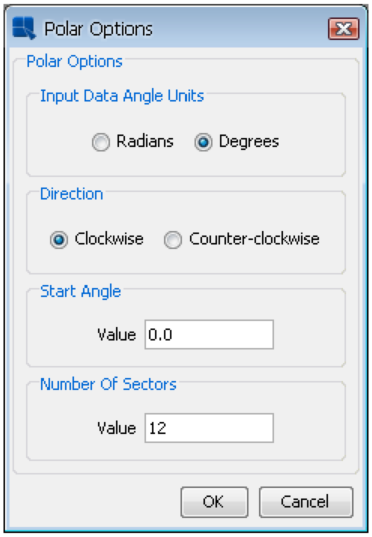
Scatter Options Dialog

The *Combo Line* tab allows you to set Draw Lines In Data Source Order, Draw Arrows on lines or Draw the connecting lines as a Step Lines, as well as specify the step lines ratio.

The *Cut Off Point* tab allows you to specify a maximum value for the Y point of the scatter coordinates. Any coordinates that fall beyond this threshold will not be plotted. Connecting lines will draw up to the edge of the threshold and continue to the next data point.

3.5.9.12. Polar Charts

For polar charts, the following dialog is displayed:



Polar Options

Input Data Angle Units

○ Radians ● Degrees

Direction

● Clockwise ○ Counter-clockwise

Start Angle

Value 0.0

Number Of Sectors

Value 12

Polar Options Dialog

Scale: This option allows you to specify whether the input data for the angle (θ) portion of the data points is in radians or degrees. The chart will always display angles from 0 to 360. If the input data is in radians, it will be displayed as degrees.

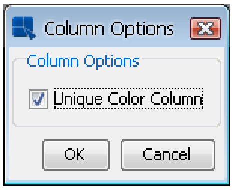
Direction This option allows you to specify whether the circular plot should be drawn clockwise or counter-clockwise.

Start Angle: By default, the top of the polar chart plot is 0 degrees. This option allows you to specify a different angle for the top of the plot. The argument for this angle is supplied in degrees or radians, depending on the scale you have chosen.

Number of Sectors: This option allows you to select number of sectors you'd like to show in the chart. Sectors are created by drawing additional polar axis lines at specified angle intervals. By default, four sectors are shown.

3.5.9.13. Column Charts with Series

For column charts with data series, the following dialog is displayed:



Column Options

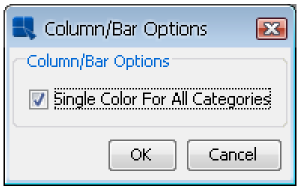
☑ Unique Color Columri

Column Options Dialog

Normally, when column charts have data series, each series has it's own color that is applied for every category in the chart. If you want to assign different colors to the columns in the chart regardless of the series, you can enable the *Unique Color Column* option in this dialog. When it's turned on, you can set color for each column in the chart individually.

3.5.9.14. Column/Bar Charts without Series

For column/bar charts without data series, the following dialog is displayed:



Column/Bar Options

Column/Bar Options

☑ Single Color For All Categories

Column/Bar Options Dialog

Normally, when column/bar charts don't have data series, all categories in the chart have single color. If you want to assign different colors to the categories in the chart, uncheck the *Single Color For All Categories* option in this dialog.

3.5.9.15. Two-Dimensional Line Combination Charts

For any other two-dimensional line combination chart, the following dialog is displayed:

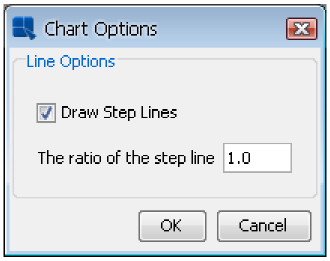


Chart Options

Line Options

☑ Draw Step Lines

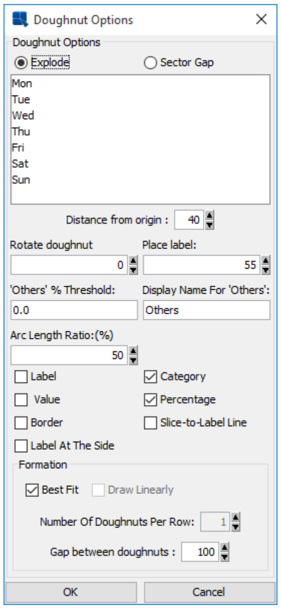
The ratio of the step line 1.0

Line Combination Chart Options

This dialog allows you to specify whether to draw the combo line as a step line, as well as specify the step line ratio.

3.5.9.16. Doughnut Charts

The chart options for a doughnut chart are almost exactly the same as the options for a pie chart. You can refer to the options under Section 3.5.9.4 - Pie Charts for more details.



Doughnut Options

Doughnut Options

● Explode ○ Sector Gap

Mon
Tue
Wed
Thu
Fri
Sat
Sun

Distance from origin : 40

Rotate doughnut 0

Place label: 55

'Others' % Threshold: 0.0

Display Name For 'Others': Others

Arc Length Ratio:(%) 50

□ Label ☑ Category

□ Value ☑ Percentage

□ Border □ Slice-to-Label Line

□ Label At The Side

Formation

☑ Best Fit □ Draw Linearly

Number Of Doughnuts Per Row: 1

Gap between doughnuts : 100

Doughnut Chart Options

The only option unique to doughnut charts is the Arc Length Ratio (%) which specifies the size of the hole at the center of the chart. A higher number results in a smaller hole.

3.6. Drill-Down

Like with reports, it is also possible to add layers of drill-downs to charts. All of the drill-downs added will be applied to both data points on the plot area and their respective fields in the legend. Using chart drill-down, you can create a top-level chart that displays summarized data and allows users to click through specific data points to see underlying details.

The charting engine in EspressReport contains several different built-in drill-down mechanisms that allows you to create only one chart template for each level of drill-down. All of the drill-down options are available from the Drill-Down menu in Chart Designer.

Warning

It is important to note that you can only use drill-downs in charts when the chart is deployed independently from the report (in the Chart Viewer, its own servlet, etc). If a chart template that includes drill-down is placed in a report, only the top-level chart will display. Users will not be able to click through to the lower-level charts.

3.6.1. Data Drill-Down

Data drill-down allows you to group and display information that is based on a single data source. The advantage to this form of drill-down is that it works with any data source. However, it does not allow you to display loosely related information because all levels of drill-down will share the same value column from the input data.

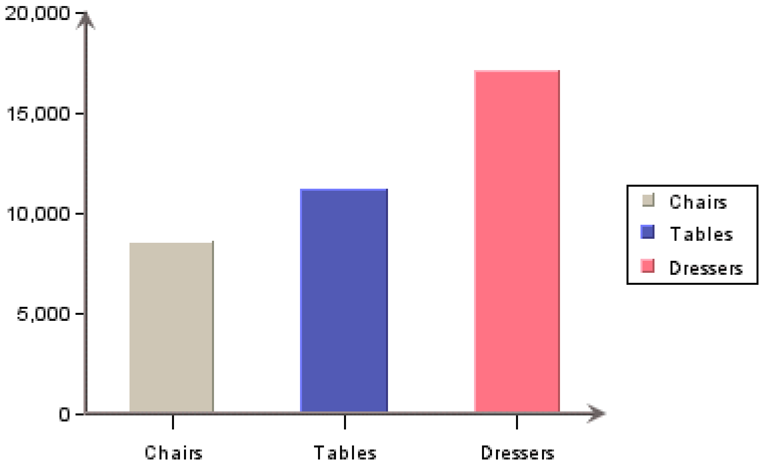
For example, assume you have the following table as the chart data:

Category	Product	Sales
Chairs	Elm Arm Chair	$8,216
Chairs	Pine Side Chair	$7,611
Chairs	Redwood Arm Chair	$8,625
Tables	Elm Round Table	$10,241
Tables	Pine Oval Table	$9,663
Tables	Oak Oval Table	$11,261
Dressers	Oak Single Dresser	$16,442
Dressers	Elm Double Dresser	$17,148

Using this data, you could create a top-level chart that displays total sales for each distinct product category and then create a lower-level chart that shows individual sales for each product in a category. The number of drill-down levels available depends on the number of groupings that are present in the input data.

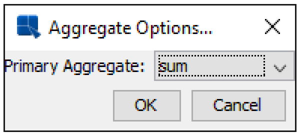
3.6.1.1. Adding Data Drill-Down

To add layers of data drill-down to a chart, you must first create a top-level chart. In the above example, we would create a chart with “Category” mapped to the category axis and “Sales” mapped to the value axis. In a column chart, it would look like this:



Top-Level Chart

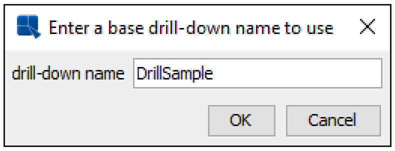
To add a layer of drill-down, select Drill-Down → Add. This will bring up a dialog prompting you to specify the aggregation you want to use for the value axis. For example, selecting sum will display the total for each category in the top-level chart. Available aggregate functions are minimum, maximum, average, sum, count, first, last, sumsquare, variance, stddev, and countdistinct.


Aggregate Options...

Primary Aggregate: sum

Aggregation Dialog

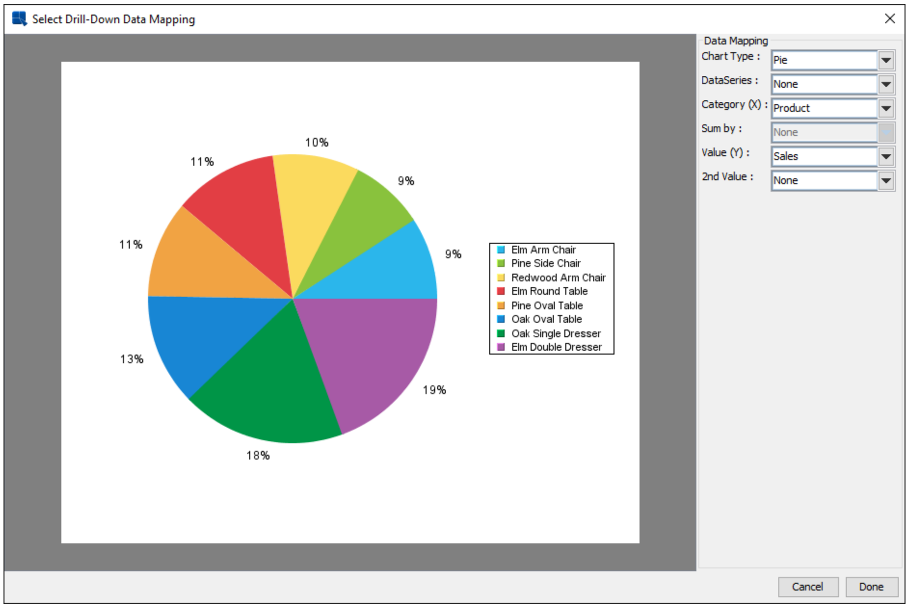
After you select the aggregation you want to use, click on the *OK* button. A new dialog will appear prompting you to specify a name for the drill-down chart. This name will be assigned to the chart templates that are created by the drill-down process. Note that all sub-level drill-down templates are saved in the `/drilltemplates/` directory.


Enter a base drill-down name to use

drill-down name DrillSample

Drill-Down Name Dialog

Once you have specified a name you want to use, click on the *OK* button. You will then be prompted to specify the chart type and data mapping for the sub-level chart.



Data Mapping

Chart Type : Pie

DataSeries : None

Category (X) : Product

Sum by : None

Value (Y) : Sales

2nd Value : None

Data Drill-Down Mapping Dialog

The options in the dialog are similar to those for normal data mapping. The major difference is that the first option allows you to select a chart type. Available chart types for data drill-down are column, bar, line, stack column, stack bar, pie, area, doughnut, overlay, radar, and dial. The data mapping options will change depending on the type of chart that you select.

Once you have finished specifying the mapping options, click on the *Done* button and you will go back to the Chart Designer where you can customize and modify your sub-level chart. You can add additional layers of drill-down by selecting Drill-Down → Add again.

You can navigate to a particular drill-down chart by selecting Drill-Down → Previous or Drill-Down → Next or by double clicking a data point. The *Previous* selection takes you up a level, while the *Next* selection goes down a level using the left-most category as the data to be drilled on. You can also go from any drill-down chart to the top-level chart by selecting Drill-Down → Go To Top Level. You can customize (i.e. assign colors, add title, axis labels, background image, ...etc.) the drill-down chart in the same way as the top-level chart. Everytime you change and leave a level, you will be prompted to save the chart. Make sure that you answer *yes* if you want the attributes of the drill-down level to be saved, otherwise you will you lose all the changes.

You can insert a drill-down chart between two other drill-down charts by going to the higher level drill-down chart and selecting Drill-Down → Add. The drill-down chart that you create will be inserted between the two previous drill-down charts.

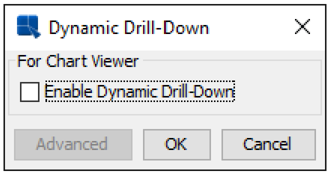
You can remove a level of the drill-down by navigating to that level and selecting Drill-Down → Remove This. The entire drill-down chart can also be deleted by selecting Drill-Down → Remove All. Selecting Drill-Down → Previous brings the drill-down chart to a higher level, which is the same as right clicking and selecting *Back* from the pop-up menu. The *Next* selection brings the drill-down chart to a lower level, which has the same function as double clicking on an item in the chart.

Once you have finished creating and editing all the levels of the drill-down chart, you can save it by navigating to the top-level chart and selecting File → Save or File → Save as.

3.6.2. Dynamic Data Drill-Down

Usually, mapping and drill options for data drill-drown are fixed during design time. Dynamic drill-down is an additional option that allows you to select the mapping. The only thing specified during design time is the top-level chart and aggregation.

To create a chart with dynamic drill-down, first create a chart you want to use as the top-level chart (for example, you can create the same top-level chart as before) a then select Drill-Down → Dynamic. A dialog will appear allowing you to enable dynamic drill-down.

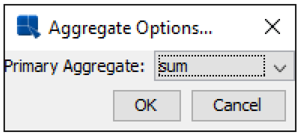

Dynamic Drill-Down

For Chart Viewer

□ Enable Dynamic Drill-Down

Enable Dynamic Drill-Down Dialog

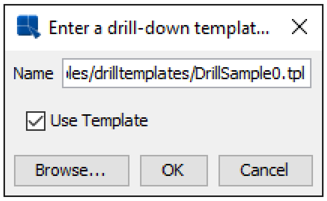
Check the *Enable Dynamic Drill-Down* box and a new dialog will pop-up prompting you to select the aggregation.


Aggregate Options...

Primary Aggregate: sum

Aggregation Dialog

Options for this dialog are the same as for regular data drill-down. Once you specify the aggregate you want to use, click on the *OK* button. Another dialog will pop-up, allowing you to specify a template you want to use for the sub-level charts.


Enter a drill-down templat...

Name ıles/drilltemplates/DrillSample0.tpl

☑ Use Template

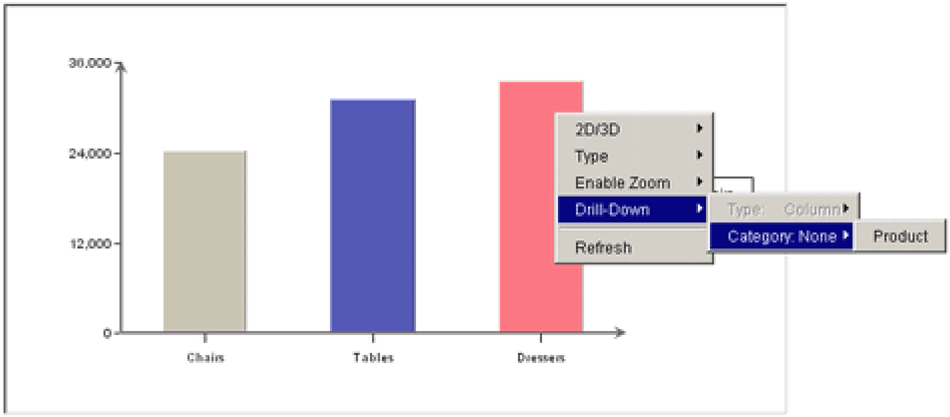
Select Dynamic Drill-Down Template Dialog

If you do not select to use a template, the sub-level charts will be generated using default appearance properties. After you finish selecting these options, your chart will change to display aggregated data on the value axis. You can change the option by selecting Drill-Down → Dynamic again and then clicking on the *Advanced* button.

Dynamic drill-down charts can only be viewed in the Chart Viewer JNLP (More about Applets in JNLP: Section 1.2.7 - Run Applets in WebStart with JNLP file). There you can configure setting of the next drill-down chart by right clicking on the desktop area which will bring up a pop-up menu. If the dynamic drill-down is enabled and the chart still has some unused fields (columns) to plot, the item *Drill-Down* will be displayed in the pop-up menu. Upon selecting *Drill-Down* in the pop-up menu, you can view the current setting for next drill-down. If *Category* is **None**, it means you have not configured the next drill-down chart yet. To configure the drill-down chart for next level, you must select *Category* (*Type*, *Series* and *SumBy* can be selected once *Category* has been set). After you create a drill-down chart, you can navigate to different levels in the Chart Viewer by either left clicking on a data point if you want to go down one level, or by right clicking and selecting *Back* from the pop-up menu if you want to go up one level, or by repeating the previous step to create another drill-down chart for next level.

Tip

After traversing to a lower-level of drill-down, right clicking on a data point will navigate you back to the top-level chart. To bring up the pop-up menu, right click on the chart canvas away from the chart data points.



Dynamic Data Drill-Down in Chart Viewer

3.6.3. Parameter Drill-Down

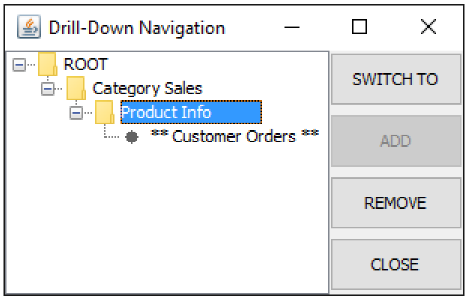
The third type of a drill-down you can use for charts is a parameter drill-down. Parameter drill-down is the most flexible implementation because you can relate the data between drill-down levels in any way you like. Specifically, you do not need to use the same value element for each level. Instead, parameter drill-down uses parameterized query feature to relate various chart levels and since it uses queries, the data source for the sub-level charts must be a database or parameterized class file.

For example, using our previous scenario, rather than always looking at sales, you want your top level chart to look at aggregated sales by category (same as before), then on the next level, you want to look at sales volume for each product, and from there, you want to look at inventory levels for each product by region. This can be accomplished with parameter drill-down.

With parameter drill-down, the category value of an element you click to drill on will be passed to the sub-level chart as the parameter value. Hence, if you click on the “Chairs” column, the value of “Chairs” would be passed to the query. Therefore, anything in the database that could be filtered by category name could be retrieved.

3.6.3.1. Adding Parameter Drill-Down

To add and edit various layers of a parameter drill-down, select Drill-Down → Parameter Drill-Down. This will bring up a navigation window that shows various levels of drill-down.



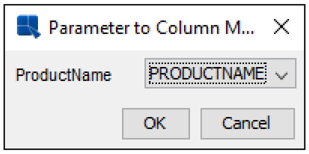
Parameter Drill-Down Navigation Window

Left-hand side of the navigation window displays the hierarchy of drill-down levels. The “ROOT” node is the top-level chart. The level you are currently editing is marked with **. To edit a different level, select it and click the *SWITCH TO* button on the right-hand side. The chart will then open in the Designer. Levels of drill-down can also be removed by selecting the node in the Navigation window and then clicking on the *REMOVE* button.

To add a new level of drill-down, select the chart under which you would like to add the layer (if you have only the top-level chart, then only the “ROOT” node will be visible), and click *ADD*. You will then be prompted to create a new chart or to use an existing chart for the drill-down layer. You can use any existing chart; however, any chart for a drill-down layer must have a parameterized query as the data source. If you select to create a new chart, the

Data Source Manager will reopen, allowing you to select a data source for the chart and to continue through the Chart Wizard steps.

The next step is to map the category and/or series columns from the top-level chart to the query parameter(s) in the sub-level chart. You will be prompted to do this when you select an existing chart for the drill-down layer or when you select the data source for a new chart for the drill-down layer. In either case, a dialog will appear prompting you to map the fields.



ProductName: PRODUCTNAME

Parameter Mapping Window

Available options in the drop-down menus are based on data type. For example, if your parameterized query has string as the parameter, only fields containing string data can be mapped. Hence, it is important to consider what type of data will be passed from the top-level chart to the lower-level charts because if your category or series are not of the correct data type or if you do not have enough fields to pass to the lower-levels, the drill-down will not work correctly.

Once you correctly specify the parameter mapping, you will be prompted to specify a display name for the drill-down level. After you select a name, the sub-level chart will appear in the Designer, allowing you to customize it. You can navigate through the layers of drill-down using the Navigation window, or like data drill-down, you can double click on a data point to go down a level and right click and select *Back* from the pop-up menu to go up a level.

3.7. Saving & Exporting Charts

After you finish designing a chart, you can save the definition as a chart or as a chart template, and provide XML definitions for both. You can also generate a number of static image exports of the chart or create a JSP/JNLP page with the Chart Viewer JNLP (More about Applets in JNLP: Section 1.2.7 - Run Applets in WebStart with JNLP file).

3.7.1. Saving Charts with Report Data

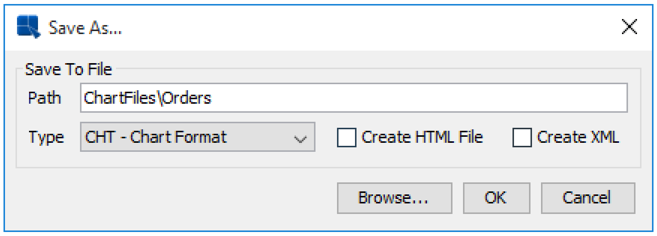
When you design a chart using report data, the chart cannot be run/viewed any place except within the report. Because of this, most of the file operations are disabled in the Chart Designer. When you opt to save a chart it will be automatically saved with the same name as the report in the `/chart/` directory of the installation. For example, if your report is named *SalesReport*, any charts in the report that use report data would be saved in the following convention `SalesReport_0.tpl, SalesReport_1.tpl,` etc

It is possible to have these charts in a different relative location when running reports. Runtime API options allow you to change the chart path.

Although most of the file options are disabled when you use report data for the chart, you can apply a template to carry over the appearance properties of another chart. Chart templates are covered in Section 3.7.2.1 - Working with Templates.

3.7.2. Saving Charts without Report Data

If your chart uses an independent data source from the report, it can be saved in any place with any filename. To save the current chart, select File → Save or File → Save As, or click on the 💾 *Save* button on the toolbar. Assuming you have not saved the chart before, or selected File → Save As, the following dialog will appear:



Path: ChartFiles\Orders

Type: CHT - Chart Format

□ Create HTML File

□ Create XML

Save As Dialog

The first option allows you to specify a name and file path for the saved chart. You can browse to the appropriate file path by clicking on the *Browse* button at the bottom of the dialog. The second option allows you to specify which format you would like to use when saving the chart. As detailed in Section 3.2 - Charting Basics, there are two principle ways in which chart definitions can be saved.

Chart format: Chart files save the chart in a binary file called `filename.cht`. A chart file stores both the definitions of the chart (type, dimension, etc), as well as the data that was used to create the chart.

Template format: Template files save the chart in a binary file called `filename.tpl`. A template file only stores chart definitions. It doesn't store any chart data. Hence, any time a template file is opened, it will try to connect to the original data source to retrieve the data.

PAC format: PAC files save the chart in such a way that makes them ready for deployment. A `.PAC` file takes the chart and all the supplementary files associated with it, such as background images, dynamic drill-downs, or parametric drill-downs, and places them in a single binary file.

Once you select a format you want to use, click on the *OK* button and the chart will be saved.

Note

When your report is run, EspressReport will try to load the chart along a relative path from where ever the chart is saved. If the chart cannot be found, it will not display. However, you can specify a different path for the chart using options available in the API.

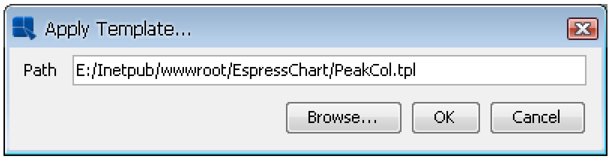
3.7.2.1. Working with Templates

Chart templates are a special format in which chart definitions can be saved. Unlike chart files (`.cht` format) which saves both the chart definitions as well as the data for the chart, templates only save the definitions (i.e. the chart attributes and the layout of the individual components), and the data source information. This means that templates store the location/connection information for the data source that was used to create the chart, but they store none of the actual data in the file. (Templates do keep 10 records of back-up data, allowing them to be opened when the data source is not present).

Template files can be used in two ways. First, it can be opened, viewed, or exported, allowing you to see a chart with fresh data. This way data for the chart are retrieved based on the data source specified in the file. If the original data source is not accessible, this method cannot be used. The second way in which templates can be used is to apply its attributes to other charts. In this scenario, the chart to which the template is applied will inherit appearance properties of the template (including colors, fonts, size of chart components, position of legends, etc). This feature allows you to produce a consistent look and feel among charts.

The second approach is very useful if you are using the API to generate charts programmatically, using JDBC code or some other data source. You can then use a pre-defined template to control the look and feel of the generated chart without having to define the appearance properties in code or relying on the default chart properties. For more on applying templates in the API, please see Section 3.9.5.2 - Applying a Chart Template.

You can apply a template in Chart Designer by selecting File → Apply Template. This will bring up a dialog prompting you to specify the template file you want to use.



Path E:/Inetpub/wwwroot/EspressChart/PeakCol.tpl

Apply Template Dialog

The dialog allows you to specify the template file and its location. You can browse to a file by clicking on the *Browse* button.

Note that when the chart and the template being applied are different sizes (i.e. chart canvas size), the resulting chart may not display correctly. This is because the text size will not change between template and chart to which it is applied. While the other components will adjust to the size of the new canvas, the font will not. To keep a consistent look, the size of your template should be close to the size of the chart to which it will be applied.

Hyperlinks, floating lines, floating text, and axis scales defined in the template files carry over. You may have to redefine them in the chart, if necessary. Chart type and dimension are not modified by the template. For example, a three-dimensional chart will not be changed to a two-dimensional chart if the template is a two-dimensional chart. Likewise, a pie chart remains a pie chart when a template of a bar chart is applied. Although the chart type does not change when you apply a template, some appearance properties will not translate very well from a template with another chart type. For best results, try to apply templates of the same type and dimension to a chart.

3.7.2.2. Saving XML Templates

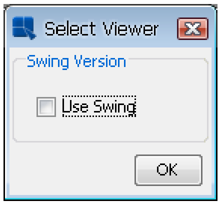
In addition to the two binary chart representations, you can also save the chart definitions in XML format. This option allows you to have a text-based chart template that can be used to modify chart properties outside of Chart Designer or the API.

To create an XML file when saving the chart, check the *Create XML File* box in the save as dialog when saving the chart. This will create an XML file for the chart. Note that the XML file is a representation of the `.tpl` format and not the `.cht` format.

3.7.2.3. Creating a Viewer Page

The other option to specify is whether you would like to create an HTML page to view the chart. The HTML page will redirect to a JSP which contains the Chart Viewer JNLP, which allows end users to view and manipulate the chart. The features of the Chart Viewer are discussed in detail in Section 3.8 - Chart Viewer

To create an HTML page when saving the chart, check the *Create HTML File* box in the save as dialog when saving the chart. This will create the HTML page with the same name as the chart file under the `html/`/ directory of your EspressReport installation. Before the file is written, you will be prompted to select which version of the Viewer that you would like to use.



□ Use Swing

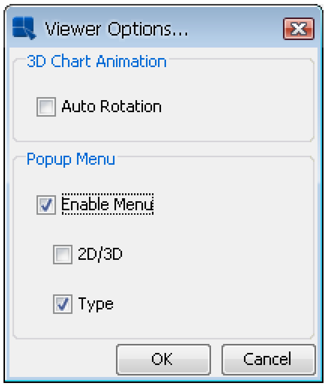
Select Viewer Dialog

EspressReport supports AWT and swing versions of Viewer. Note that if you use Viewer, the client will need to have the Java plug-in.

Once you have created the HTML page, you can point your browser to the page to view the chart. Note that in order for the chart to appear you must load the HTML page vial http protocol. It will not work it you attempt to load it over a file protocol (i.e. browsing to the file and opening it). This means that your EspressReport must be installed in a Web server, and you must access the page via http (i.e. `http://machinename:port/EspressReport/yourfile.html`) in order for it to load correctly.

3.7.2.3.1. Viewer Options

There are a number of Chart Viewer options that you can configure from within Chart Designer. These are properties that are saved with the chart, which will appear when the chart is displayed in the viewer. To modify the viewer options, select Format → Viewer Options. This will bring up the following dialog.



□ Auto Rotation

☑ Enable Menu

□ 2D/3D

☑ Type

Viewer Options Dialog

The following options are available:

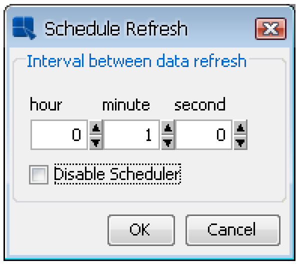
Auto Rotation: This will enable automatic rotation for three-dimensional charts. The charts will begin to rotate when the applet is loaded. The rotation can be stopped using the appropriate button on the navigation panel.

Enable Menu: This allows you to turn on or off the pop-up menu when users are viewing the chart in Chart Viewer.

2D/3D. This allows you to turn on or off the dimension toggle in the pop-up menu. If this is turned off, users will not be able to change the chart dimension.

Type: This allows you to turn on or off the type sub-menu in the pop-up menu. If this is turned off, users will not be able to change the chart type.

For `.cht` files, you can also set a scheduled refresh. This will allow the chart to periodically refresh the data when the chart is being viewed in Chart Viewer. To schedule a refresh rate, select Data → Schedule refresh. This will bring up a dialog, allowing you to set the schedule options.



hour 0 minute 1 second 0

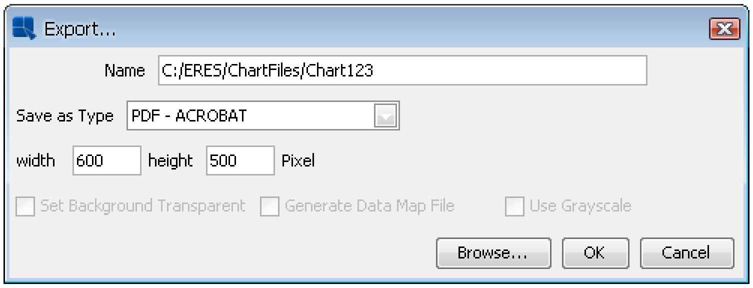
□ Disable Scheduler

Schedule Refresh Dialog

From this dialog you can set the hour, minute, and second for the refresh interval. If you check the *Disable Scheduler* box, the Scheduler will be turned off.

3.7.3. Exporting Charts

From the Chart Designer, you can generate a number of static image exports for your chart. To export the current chart, select File → Export, or click the *Export* button on the toolbar. This will bring up a dialog allowing you to specify options for the generated file.



Name C:/ERES/ChartFiles/Chart123

Save as Type PDF - ACROBAT

width 600 height 500 Pixel

□ Set Background Transparent □ Generate Data Map File □ Use Grayscale

Export Chart Dialog

The first option allows you to specify a name and the file path for the generated file. You can browse to the appropriate file path by clicking the *Browse* button at the bottom of the dialog. The second option allows you to select what type of export you would like. The following options are available:

GIF EspressReport can generate GIF images. GIF has 256 color limit which keeps image file sizes small.

JPEG JPEG is another popular image format. It is a higher resolution image format than GIF and it is not patent protected. When generating a JPEG file, you can specify quality/compression of the file. The higher the quality, the larger the file.

If you select JPEG as the export format, after clicking *OK* you will be prompted to specify the quality. The higher quality image you select, the larger the generated file size. It is recommended that with the JPEG export, you use specify a high-quality image, as the low-quality results will most likely be undesirable.

PNG PNG is an image format that is less popular, but can be displayed in most browsers. It is a high-quality image with a smaller file size than JPEG. It also provides transparent image background option.

SVG SVG (Scalable Vector Graphics) is a relatively new image format, which saves the image as vectors in an XML-based text format. Generally, you will need a browser plug-in to view these images.

SWF SWF is an Adobe Flash file. The flash format is vector based and allows the chart to be resized after export. Also, flash allows for high-resolution printing and produces a small file size. When selecting this export type, you can also specify the frame count (number of frames in the animation) and the frame rate (number of frames shown per second) or choose to disable the animation.

BMP This is a Windows bitmap format.

WMF WMF is the Windows Meta File format. This can be used for import/export into MS Office documents.

PDF This will generate the chart in Adobe Portable Document Format. The chart will be generated as a one-page PDF document.

XML This will generate an XML data file containing the chart's data. This will not generate an XML chart attribute file, only the data will be written into XML format.

XLS Will generate an XLS (MS Excel) file and insert the chart as an image to the first XLS sheet.

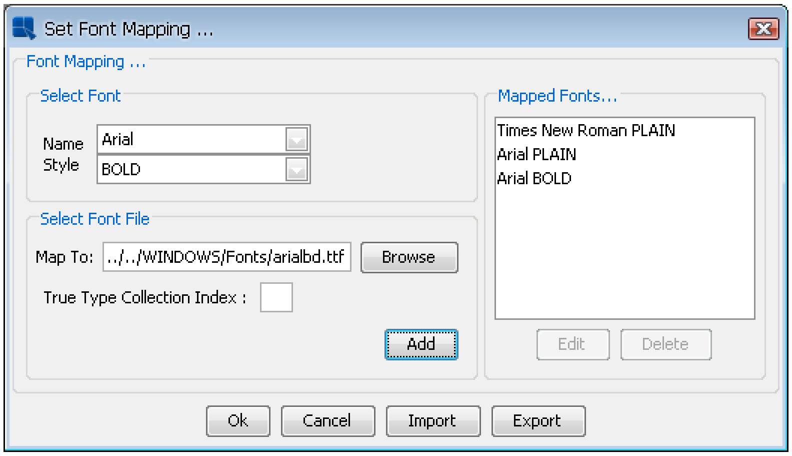
TXT This will generate a text data file containing the chart's data.

In addition, if your chart contains hyperlinks, you can choose to export a map file along with any of the generated images. The map file contains an HTML image map with the links for the generate chart. Map files are generated in the same location as the image file, and have the same file name. To generate a map file, check the *Generate Data Map File* box prior to exporting.

3.7.3.1. PDF Font Mapping

As with reports, you can use any font in the system for the labels, text, and titles in a chart. For most image exports, the text is written in the image and no additional configuration is necessary. For PDF export, however, you will need to specify a `.ttf` (true type font), `.ttc` (true type collection), `.pfb`, or `.afm` file for any system font you want to use in the chart.

To set font mapping for PDF export, select Format → Font Mapping. This will bring up a dialog allowing you to specify font files.



Name: Arial
Style: BOLD
Map To: ../../WINDOWS/Fonts/arialbd.ttf

Times New Roman PLAIN
Arial PLAIN
Arial BOLD

Font Mapping Dialog

For each font and style combination, you can select a specific `.ttf`, `.ttc`, `.pfb`, or `.afm` file for that font. You can either type the full path or browse to the font file. If you are using a `.ttc` file you will need to specify the font index in the box provided (`.ttc` files contain more than one font). Once you have specified the correct file, click the 'Add' button to save the mapping to the list. You can edit or delete existing mappings by selecting them in the list and then clicking the appropriate button.

Note

You only need to apply font mapping when exporting the chart by itself. If the chart is in a report, it will inherit the font mapping defined in the report.

3.7.3.1.1. PDF Font Mapping Import/Export

You can pass the font mapping from one chart to another using the Import/Export feature. You can export the font mapping by clicking on the *Export* button on the font mapping dialog. This will bring up a dialog box prompting you to specify a file name. The font mapping will then be saved as an XML file. You can load a font mapping XML file by selecting the *Import* option from the dialog. This will bring up a dialog box prompting you to specify the XML file that you would like to import. Click on *OK* and the mapping stored in the XML file will be applied to the current chart.

3.8. Chart Viewer

All charts created by Chart Designer can be saved in a range of file formats that may be pasted into documents. These formats include BMP, JPG, PNG, PDF, SVG, SWF, WMF, and GIF. In addition, Chart Designer provides the option of saving a chart as a `.cht`, `.tpl`, or `.xml` file.

Chart Viewer is a JNLP (More about Applets in JNLP: Section 1.2.7 - Run Applets in WebStart with JNLP file) that enables you to view and manipulate a chart dynamically through a web browser. Viewer reads the file (in .cht or .tpl format) as outputted by Chart Designer or the API, and then displays the chart. The small size of the data file makes it suitable to distribute the chart image over the web. The data is encrypted while being transferred from EspressManager to Chart Viewer and so lends a degree of security to sensitive information.

A `.cht` file may be viewed interactively by a user using the Chart Viewer JNLP (More about Applets in JNLP: Section 1.2.7 - Run Applets in WebStart with JNLP file). Chart Viewer enables a chart to be displayed and manipulated without changing the underlying data.

Files in the `.tpl` format can also be viewed using Chart Viewer. When a web page that contains a .tpl file is viewed using a browser, fresh data will be obtained automatically by Chart Viewer. Thus, a single chart template can be used to supply up-to-the-minute charts to users in real time.

Inside Chart Viewer, you can drag the chart, legend, title, or label to position the object. You can also resize the chart and drill-down on data points or a series of data. For three-dimensional charts, users can use the navigation panel to pan, zoom, rotate in each direction, and translate. Also, individual x, y, and z-axis scaling, thickness ratio adjustment, real time three-dimensional animation, etc can all be preformed easily. There are built-in callback mechanisms that let a user click on a data element to view the underlying data or to jump to a related URL. Chart Viewer is written in pure Java that runs on all platforms that support Java. Chart Viewer also supports scheduled refresh (where a chart's data is updated at regular intervals specified by the designer) and parameter serving where a chart's parameters are provided at load time.

You can embed a `.cht`, `.pac` or `.tpl` formatted file in a Web page using the following tag inserted into EspressViewer.jsp as shown below:

```
<applet-desc
      name="Chart Viewer"
      main-class="quadbase.chartviewer.Viewer"
      width="800"
      height="600">
        <param name="filename" value="help/examples/ChartAPI/data/test.tpl"/
>
        <param name="preventSelfDestruct" value="false"/>

</applet-desc>
```

The parameter `filename` specifies the file name of the file that contains the chart data and you can prefix it by `http://` for accessing a remote data file. When viewed by Chart Viewer, a chart saved in the chart format (`.cht`) will use the data stored in that file for plotting.

A chart saved in the template format (`.tpl`) allows Chart Viewer to dynamically fetch the data from a database or a data file depending on where you specify the data source of chart to be when using Chart Designer to create the template (the database name, user name, password, etc are all stored in the `.tpl` file).

3.8.1. The Chart Viewer Parameters

You can also pass data and chart viewing control information via parameters to the Chart Viewer JNLP (More about Applets in JNLP: Section 1.2.7 - Run Applets in WebStart with JNLP file) without using Chart Designer. That is, you can use the Chart Viewer to directly view a data file or pass the data directly in the form of lines of data, along with other control information, in the HTML code. By default, the parameter is true if the parameter is of a true/false type. The following is a list of parameters:

Chart Parameters (Common to Two and Three-Dimensional Charts):

mainTitle	the main title of the chart
xTitle	x-axis title
yTitle	y-axis title
zTitle	z-axis title
RefreshInterval	scheduled refresh interval in seconds
DragLegend	if false, the legend(s) can not be moved
DragChart	if false, the chart can not be repositioned or resized
ShowDataHint	if false, data information box will not be shown when left mouse click on chart data
ShowLinkHint	if false, hyperlink information box will not be shown when right mouse click on chart data
DataHintBgColor	set background color of data information box
LinkHintBgColor	set background color of hyperlink information box

DataHintFontColor set font color of data information box

LinkHintFontColor set font color of hyperlink information box

DataHintFont set font of data information box

LinkHintFont set font of link information box

DataHintOffsetX set x offset of the data information box

DataHintOffsetY set y offset of the data information box

LinkHintOffsetX set x offset of the link information box

LinkHintOffsetY set y offset of the link information box

Printing if false, this will disable the ability to export the chart (using **Ctrl**+**P** and/or **Ctrl**+**J**) in a browser

filename name of the template file to be applied to the chart

xAxisRuler if true, show the x-axis ruler (for 2D charts only)

yAxisRuler if true, show the y-axis ruler (for 2D charts only)

sAxisRuler if true, show the secondary-axis ruler (for 2D charts only)

ResizeChart if false, the chart cannot be resized

ResizeCanvas if false, the canvas cannot be resized

comm_protocol the protocol to be used, in case of a firewall

comm_url the url to connect to the EspressManager, in case of a firewall

RefreshData if false, the chart data cannot be refreshed

PopupMenu if false, the pop-up menu will not be displayed

TypeMenu if false, the type sub-menu will not be displayed in the pop-up menu

DimensionMenu if false, the dimension sub-menu will not be displayed in the pop-up menu

For Three-Dimensional Charts only

Toggle3Dpanel if false, the navigation panel can not be toggled to be visible or invisible

Drawmode set different mode of drawing 3D chart. Available draw modes are **`Flat`** (default), **`WireFrame`**, **`Flat Border`** (which draws a black border around the flat shading model), **`Gouraud`**, and **`Gouraud Border`**

NavColor set navigation panel color

navpanel if false, the Navigation Panel is not displayed when a 3D chart is being viewed. The Navigation Panel is never displayed when a 2D chart is being viewed

GouraudButton if false, the Gouraud shading button in the navigation panel is hidden

AnimateButton if false, the animation speed control in the navigation panel is hidden

SpeedControlButton if false, the speed control button in the navigation panel is hidden

Data Input Parameters

sourceDB	set the database information in order to generate the chart
sourceData	set the data information in order to generate the chart
sourceFile	set the datafile information in order to generate the chart
datamap	set the column mappings for the chart
TransposeData	set the data to be transposed before using it to generate the chart
chartType	set the chart type for the generated chart
EspressManagerUsed	Set the EspressManager to be used
ParameterServer	update the data in the chart dynamically
transposeData	if true, transpose the data
server_address	The IP address of the Espress Manager connection.
server_port_number	Port number of the Espress Manager connection.

*Example:*the parameters `mainTitle`, `xTitle`, `yTitle`, and `zTitle` are used to specify the main title and axis title of the chart, they will override the ones defined in the template:

```
<PARAM name="mainTitle" value="This is the main Title">
<PARAM name="xTitle" value="x axis title">
<PARAM name="yTitle" value="y axis title">
<PARAM name="zTitle" value="z axis title">
```

*Example:*With the parameter `RefreshInterval` you can specify:

```
<PARAM name="RefreshInterval" value="60">
```

In the above example, the JNLP (More about Applets in JNLP: Section 1.2.7 - Run Applets in WebStart with JNLP file) will fetch data from a database or data file (with the help of EspressManager) and redraw the chart every 60 seconds - all transparently. It is useful for accessing databases in which the data changes frequently.

3.8.2. Specifying the Data Source for Chart Viewer

Using parameters, you can specify a data source for the Chart Viewer in order to display different data with a chart template or create a chart from scratch.

3.8.2.1. Data Read From a Database

This is some sample JSP/JNLP code that uses the Chart Viewer to view a chart drawn using data extracted from a database.

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.0+" href="EspressViewer.jnlp">
<information>
<title>Espress Viewer</title>
<vendor>Quadbase Systems Inc.</vendor>
<offline-allowed/>
</information>
<resources>
<j2se version="1.8+" max-heap-size="1024m"/>
<jar href="lib/EspressViewer.jar"/>
</resources>
```

```
<security>
<all-permissions/>
</security>
<applet-desc
name="Chart Viewer"
main-class="quadbase.chartviewer.Viewer"
width="640"
height="480">

<PARAM name="sourceDB" value="jdbc:odbc:DataSource,
 sun.jdbc.odbc.JdbcOdbcDriver, username, password, select * from products">
<param name="dataMap" value="0 1 -1 3">
<param name="chartType" value="3D Column">

</applet-desc>
<update check="always" policy="always"/>
</jnlp>
```

The arguments of `dataMap` specify how the chart utilizes different columns from the input data to plot the chart. In case of a scatter chart, they are series, x-value, y-value, and z-value. For a high low open close or high low chart the numbers are series, category, high, low, open, and close. For all other charts, the arguments are series, category, sumBy, and value. If you would like more information,please see the chapter on Column Mapping in the Chart API reference. The argument `chartType` specifies the type of chart to be displayed and it can be one of:

- 2D Column
- 3D Column
- 2D bar
- 3D bar
- 2D stack bar
- 3D stack bar
- 2D stack column
- 3D stack column
- 2D area
- 3D area
- 2D stack area
- 3D stack area
- 2D line
- 3D line
- 2D pie
- 3D pie
- 2D scatter
- 3D scatter
- 2D High Low

- 3D High Low
- 2D HLCO
- 3D HLCO
- 2D 100% Column
- 3D 100% Column
- 3D Surface
- 2D Bubble
- 2D Overlay
- 2D Box
- 2D Radar
- 2D Dial
- 2D Gantt
- 2D Polar

3.8.2.2. Data Read From a Data File

This HTML/jsp code draws a chart using data from a data file:

```
<applet-desc code = "quadbase.chartviewer.Viewer.class" width=640
 height=480>

      <param name="sourceFile" value="http://.../test.dat">
      <param name="dataMap" value="-1 0 -1 1">
      <param name="chartType" value="3D Pie">
</applet-desc>
```

3.8.2.3. Data Read From an Argument

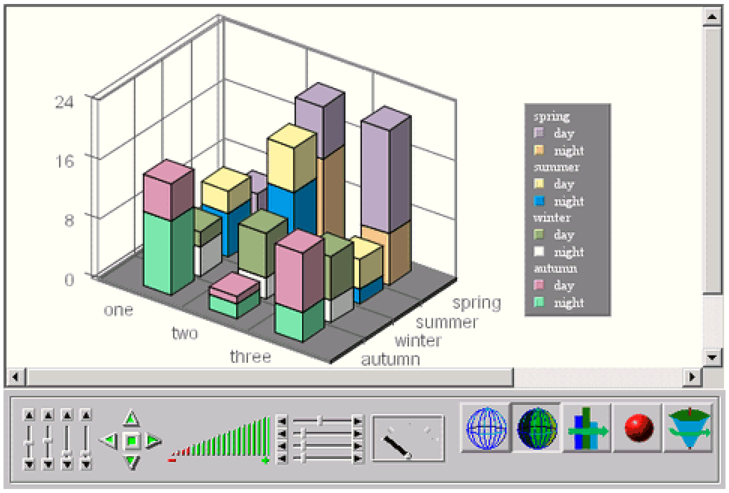
It is possible to have Chart Viewer read in data directly from the HTML/jsp file itself rather than from a data file or from a database.

```
<applet-desc code = "quadbase.chartviewer.Viewer.class" width=640
 height=480>

      <param name="sourceData" value="int, string, int | value, name, vol |
 10, 'John', 20 | 3, 'Mary', 30 | 8, 'Kevin', 3 | 9, 'James', 22">
      <param name="dataMap" value="-1 1 -1 2">
      <param name="chartType" value="3D Bar">

</applet-desc>
```

The format for the parameter sourceData is the same as the data file format, except that each line is ended by a vertical bar “|”.

3.8.3. Using Chart Viewer

You can manipulate the chart appearance simply by using the mouse in Chart Viewer. For three-dimensional charts, Chart Viewer screen comprises of two sections: the *drawing panel* and the *navigation panel*. Only the *drawing panel* is visible for two-dimensional charts. Within the *drawing panel*, the chart itself is located on the *plot area*.



A Sample Chart Viewer Display

Below is a list, not by order or importance, of chart manipulations that are possible when using Chart Viewer:

- Using the navigation panel, you can:
  - Change light position
  - Rotate the chart
  - Translate the chart
  - Zoom in/out on the chart
  - Scale the chart in x, y, z dimension
  - Adjust the thickness ratio of bar/line/point/pie
  - Start the chart animation and control the speed of the animation
  - Toggle between wireframe and solid mode
  - Draw a black outline on all edges of the chart
  - Perform Gouraud shading
- Drag the left mouse button on the chart, main title, x, y, and z-labels, or legend to move the item
- Drag the right mouse button on plot area to resize the chart
- Press the **Alt** key and drag the right mouse button on plot area to resize the canvas
- To toggle the navigation panel (show and hide), **double click** the left mouse button on the drawing panel
- Using the mouse you can also query data points individually
- **Left single click** on a data point to view the data associated with that point
- A **right single click** on data provides the name of the hyperlink associated with the data point
- **Left double click** to jump to the hyperlink associated with the data point. (Note that this creates a new browser window. Users can not use the *Back* button on the browser to return to the previous chart)
- **Right double click** to jump back to the previous chart (if appropriate)
- Use **Alt**+**Z** to specify the zoom-in parameters

- Use **Ctrl**+**R** to refresh data manually
- Use **Ctrl**+**Left Click** to select the bounds to zoom in and **Ctrl**+**Right Click** to zoom out

3.8.4. Axis Rulers

There is now an option to show Axis Rulers in Chart Viewer (by setting a parameter `{axisRuler}` in the applet-desc tag) and Chart API (Please refer to the API Documentation quadbase.util.IAxisRuler [https://data.quadbase.com/Docs71/eres/help/apidocs/quadbase/util/IAxisRuler.html]). This provides a reference point when scrolling is enabled and the chart is moved such that the chart's own axes are not visible. This feature is for 2D charts only.

3.8.5. Parameter Server

The Parameter Server allows Chart Viewer to listen to requests from a specified port using TCP/IP and update the data dynamically. The syntax in JNLP is:

```
<applet-desc
name="Chart Viewer"
main-class="quadbase.chartviewer.Viewer"
width="640"
height="480">

<param name="ParameterServer" value="machine:portno">

</applet-desc>
```

For security reasons, the machine is usually the same as the web server machine. Therefore, you can leave machine empty (i.e. **`value=":portno"`**).

For more information on the parameter server, please refer to Appendix 2.A - Parameter Server.

3.8.6. Pop-up Menu

If the pop-up menu option was selected during the chart creation/editing, the chart can be modified. The menu is opened by right clicking on the chart and the options there can be selected by highlighting and left clicking on it. Options available for viewing, when enabled, are dynamic data drill-down, changing chart types, axis zooming, and time-series data zooming.

3.8.6.1. Changing Chart Dimension and Type

The chart dimension and chart type can be changed by using the pop-up menu and selecting either the *2D/3D* option or the *Type* option.

Please note that *Overlay*, *Box* and *Dial* chart options do not appear if the chart is three-dimensional.

3.8.6.2. Axis Zooming

In the case of a large chart whose dimensions exceed that of the viewport (i.e. if a chart is large enough that it cannot be completely seen within window), an option exists to allow for axis zooming. This allows you to move all the axis and zoom in and out of the axis.

This option is only available for two-dimensional charts. Please note that the bubble, dial, pie, polar, radar, scatter, stack area, and vertical box charts do not support this feature.

You can enable this option by opening the pop-up menu and selecting *Enable Zoom*. You can choose from the zooming in on the x-axis, the y-axis or both.

To zoom in, left click on and drag out the desired area. If only the y-axis zooming is selected, you need not worry about the length or position in the x-axis.

Scrolling is enabled only for axes that have been zoomed. There will be scrollbars visible on the particular axis when axis zooming is enabled. You can left click and drag this scrollbar to shift the viewport of the applet.

Holding down the control key while left clicking will take the applet back to the previous zoom. Please note that only one previous zoom is stored in memory and thus you can only go back one step. To go all the way back to the default x and y scale, press the **Home** key on your keyboard.

3.8.6.3. Zooming

If the chart being shown is a zoom enabled chart, the zoom option will be available from the pop-up menu. Using the zooming option, you can group the category elements into user-defined intervals and aggregate the points in each group. For details on date/time based zooming, see Section 3.5.8.2 - Date/Time Based Zooming. Enabling zoom will allow you to input various zoom options such as lower bound, upper bound (you can also disable the bounds in which case it will go from the first data point to the last), the time scale, and whether you want the x-axis to be linear or not.

3.8.6.4. Dynamic Data Drill-Down

You can configure the next drill-down by selecting *drill-down* from the pop-up menu. This option is available only if dynamic data drill-down is enabled for the chart. After selecting the *Drill-Down* option the in the pop-up menu, you can view the current setting for the next drill-down chart. If the *Category* is *None*, the next level has not yet been configured. If there are unused columns in the data used to generate the chart, you can select a column for the *Category*. After the *Category* has been selected, the *Type* of the chart in the next level can be set (along with *Series* and *SumBy*). The data points can now be left clicked to move down a level and right clicked to move up a level.

3.8.6.5. Query Parameter

When viewing a parameterized chart, you can enter in a different value for the parameter (or parameters depending on the number) by opening the pop-up menu and selecting the *Query Parameter* option. The chart is then redrawn with the new data based on the parameters selected.

3.8.7. Swing Version Available

A JFC/Swing version of the Chart Viewer can also be used to referring to `SwingEspressViewer.jar` instead of `EspressViewer.jar` in the `EspressReport/lib` directory. Call the following class when using the Swing viewer: `quadbase.chartviewer.swing.Viewer.class`.

3.9. EspressReport Chart API

3.9.1. Introduction and Setup

In addition to designing and creating chart templates in the EspressReport Designer, EspressReport also provides an easy-to-use application programming interface (API) that enables users to create and customize 2D and 3D charts within their own applications and applets. It is written in 100% Pure Java and thus can be run on any platform with little or no modifications necessary. The chart template is completely customizable using the API. You can use as little as 4 lines of code to add a chart to another chart.

The data for the chart can come from two sources:

- The data can come from the report section to which the chart template is set. For instance, a Report Footer, which has a chart template in it, will show the entire report's data. If, however, the same template is assigned to the Group Footer, then only the partial data within the group is shown in the chart. Therefore, care must be taken to ensure that the correct data is shown with the chart and that the chart templates are set in the proper locations in the report.

- The data can come from an independent data source. This data source can be the same as the data source used to create the report or different from the data source. Please note that the chart data will not change depending on its placement within the report. However, multiple charts may appear depending on which section of the report the chart has been added to.

In addition to adding charts to reports, you can also generate and deploy stand-alone charts, independent of a report. You do not necessarily need to add a chart to a report (even a blank one) to show the chart.

The main classes, `QbChart` and `ChartObject` (which extends `quadbase.ChartAPI.QbChart`) are used for creating and adding a chart object to the report. Associated with this component, is a set of auxiliary classes contained within two packages: `quadbase.ChartAPI` and `quadbase.util`. The remainder of this document explains the constituents of the API and their usage.

Tip

Please note that the complete API documentation is located at help/apidocs/index.html.

To use the API, add `EspressAPI.jar` and `ExportLib.jar` (located in the `EspressReport/lib` directory) to your CLASSPATH. Please note that if you are also using XML (to output the data, or to read in data), you will also need to add `xercesImpl.jar` and `xml-apis.jar` (also located in the same directory) to the CLASSPATH as well. If you wish to use to export the chart to SVG or to Flash, you will need to add `SVGExport.jar` or `FlashExport.jar` (also located in the same directory) to the CLASSPATH. If you want to use parameterized database queries as your data sources, add `jsqlparser.jar` to your CLASSPATH. Please note that you will also need to include `qblicense.jar` in your CLASSPATH. If your application is on a Windows or Solaris machine, you will have to add the following environment variable (depending on the platform):

```
(For Windows) set PATH=%PATH%;<path to EspressReport root directory>\lib
(For Solaris) export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<path to EspressReport
 root directory>/lib
```

3.9.2. Recommended Approach for Using Chart API

EspressReport provides API through which charts can be generated programmatically from scratch. However, this generally involves a significant amount of code. We recommend using Designer and creating chart templates (`.TPL` file) first. The chart template can be passed in the `QbChart` constructor and thus have its look and feel applied to the newly created chart object. Therefore, most of the look and feel will have been set at the time of the chart generation. You can then write code to modify, add, or remove the properties. This approach will save you coding time and improve performance as well.

When using EspressReport Chart API, you have the option of connecting to EspressManager or of not connecting to it. While a connection is required when using Designer, you do not need to connect to EspressManager when running any application that utilizes EspressReport Chart API. We generally recommend that you do not connect to EspressManager and thereby avoid another layer in your architecture. For more details, please refer to the next section.

All examples and code given in the manual follow the above two recommendations: a template based approach and no connection to EspressManager. Unless otherwise noted, all code examples will use a template (can be downloaded in corresponding chapters) and will not connect to EspressManager.

Also note that if you have applets that use EspressReport Chart API, the browser must have at least a 1.5 JVM plugin.

3.9.3. Interaction with EspressManager

EspressReport is generally used in conjunction with EspressManager. The chart component connects to EspressManager in order to read and write files, to access databases, and to perform data pre-processing required for certain advanced features (such as aggregation). However, the chart component can also be used in a stand-alone mode, in which it performs file I/O and database access directly, without the use of EspressManager.

Both approaches have their own advantages. If the chart is contained within an applet, security restrictions prevent it from directly performing file input/output. Database access is also difficult without the presence of a JDBC driver on the client. In such cases, EspressManager provides the above services to the chart. On the other hand, if the chart is used in an application, there are no such restrictions, and performance can be improved by direct access. The application can be run without the necessity of having EspressManager running at a well-known location.

For instance, the applet(More in: Section 1.2.7 - Run Applets in WebStart with JNLP file) or application may be running on machine **`Client`** and may require data from a database on machine **`DbMachine`**. However, the machine **`DbMachine`** may be behind a firewall or a direct connection and may not be allowed to machine **`DbMachine`** from machine **`Client`** due to security restrictions. EspressManager can be run on machine **`Server`** and the applet(More in: Section 1.2.7 - Run Applets in WebStart with JNLP file)/application can connect to Espress-

Manager on the machine **Server**. JDBC can then be used to connect to machine **DbMachine** from machine **Server** and get the data. The data is then delivered to machine **Client** and the chart is generated. This is useful when you want to keep the data secure and non-accessible from your client machines and make all connections come through a server machine (a machine running EspressManager). You can also utilize this option to keep a log of all the clients accessing the data through EspressReport (you can have a log file created when starting EspressManager. The log file is called `espressmanager.log`). Note that this functionality comes at a cost. You will face a slight performance overhead because your code is connecting to the data through EspressManager (i.e., another layer has been added).

By default, a chart component requires the presence of EspressManager. To change the mode, use the `QbChart` class static method at the beginning of your applet(More in: Section 1.2.7 - Run Applets in WebStart with JNLP file)/application (before any `QbChart` objects are created):

```
static public void setEspressManagerUsed(boolean b)
```

Both applications and applet(More in: Section 1.2.7 - Run Applets in WebStart with JNLP file) can be run with or without accessing EspressManager. Communication is done by using http protocol. The location of the server is determined by an IP address and port number passed in the API code. Below are instructions on how to connect to the EspressManager:

3.9.4. Connecting to EspressManager

3.9.4.1. EspressManager Running as Application

EspressManager is primarily run as an application. If you wish to use ChartAPI to connect to EspressManager running as an application, you can use API methods to specify the IP address or machine name where EspressManager is located, as well as the port number that EspressManager is listening on.

You use the following two API methods to set the connection information:

```
static void setServerAddress(java.lang.String address);
static void setServerPortNumber(int port);
```

For example, the following lines of code:

```
QbChart.setServerAddress("someMachine");
QbChart.setServerPortNumber(somePortNumber);
```

will connect to EspressManager running on **someMachine** and listening on **somePortNumber**.

Please note that if EspressManager connection information is not specified, the code will attempt to connect to EspressManager on the local machine and listening to the default port number (22071).

Please note that these methods exist in `QbReport, QbChart, QbReportDesigner`, and `QbScheduler` classes.

3.9.4.2. EspressManager Running as Servlet

EspressManager can also be run as a servlet. If you wish to use Chart API to connect to EspressManager running as a servlet, you will have to use the following methods:

```
public static void useServlet(boolean b);
public static void setServletRunner(String comm_url);
public static void setServletContext(String context);
```

For example, the following lines of code:

```
QbChart.useServlet(true);
QbChart.setServletRunner("http://someMachine:somePortNumber");
QbChart.setServletContext("EspressReport/servlet");
```

will connect to EspressManager running at `http://someMachine:somePortNumber/EspressReport/servlet`.

Please note that these methods exist in `QbReport, QbChart, QbReportDesigner`, and `QbScheduler` classes.

3.9.5. Using the API

The following section details how to utilize the API. Again, the API examples and code is designed using the above recommendation (template based and no EspressManager).

Note that unless otherwise noted, all examples use the Woodview HSQL database, which is located in the `<EspressReportInstall>/help/examples/DataSources/database` directory. In order to run the examples, you will need to add database HSQL JDBC driver (`hsqldb.jar`) to your classpath. The driver is located in the `<EspressReportInstall>/lib` directory.

Also, all the API examples will show the core code in the manual. To compile the examples, make sure the CLASSPATH includes `ReportAPIWithChart.jar` and `qblicense.jar`.

For more information on the API methods, please refer the API documentation [https://data.quadbase.com/Docs71/er/help/apidocs/index.html].

3.9.5.1. Loading a Chart

Charts can be saved to a file using the PAC, CHT or TPL formats (both proprietary formats). A TPL file stores all of the chart information, except for the actual data; PAC or CHT file stores the actual data as well. These formats can be used to reconstruct a chart object. The data is automatically reloaded from the original data source each time the TPL file is opened. When PAC or CHT file is opened, the data used to create it is shown (although the option to fetch data from the data source also exists).

It is important to note that the TPL file, by default, does NOT contain the data. It contains, along with the chart template information (i.e., the look and feel of the chart), the specified data source. Therefore, when loading a TPL file, the data for the chart is obtained by querying the data source. The CHT file, on the other hand, does NOT query the data source when it is opened. It shows the data used to create the chart to begin with. Both formats can be obtained by using `Designer` or the `export()` method provided in the `QbChart` class.

The following example, which can run as a Java Web Start Application, reads a CHT file and reconstructs a chart:

```
Component doOpeningTemplate(Applet parent) {

      // Do not use EspressManager
      QbChart.setEspressManagerUsed(false);

      // Open the template
      QbChart chart = new QbChart (parent, // container
                        "../templates/OpeningChartTemplate.cht"); // template

      // Show chart in Viewer

      return chart;

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/OpeningChartTemplate.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/OpeningChartTemplate.png]

The constructor in this example is:

```
public QbChart(Applet applet, String file);
```

where applet is the applet container and file is a CHT/TPL file name.

Tip

File names may be specified either as URL strings or by using relative/absolute paths. Path names are interpreted as being relative to the current directory of EspressManager or to the current application if EspressManager is not used.

3.9.5.1.1. Parameterized Charts

Charts can also contain parameters to either limit the data in some form. Typically, query (or IN) parameters are used by the data source to limit the data.

Query parameters can be both single-value and multi-value parameter types.

When a parameterized template is opened using following constructor:

```
QbChart(Applet parent, String templateName);
```

a dialog box appears, asking for the value(s) of the parameter(s). This dialog box is the same as the one that appears in Designer when the template is opened.

3.9.5.1.1.1. Object Array

A parameterized chart can also be opened without the dialog box prompting for any value(s). This can be done by passing in an object arrays. Each element in the array represents the value for that particular parameter.

The order of the array must match the order in which the parameters were created in `Designer`. For correct results, the data type of the value must also match the data type of the parameter.

Query parameters can also be multi-value parameter types. For multi-value parameters, a vector is passed to the object array. The vector contains all the values for the multi-value parameter.

The following example, which can run as a Java Web Start Application, passes in the parameter values when opening a chart template:

```
Component doObjectArray(Applet parent) {

      // Do not use EspressManager
      QbChart.setEspressManagerUsed(false);

      // Object array for Query Parameters
      Vector vec = new Vector();
      vec.addElement("CA");
      vec.addElement("NY");

      // Object array for Query Parameters
      Object queryParams[] = new Object[3];
      queryParams[0] = vec;
      queryParams[1] = new Date(101, 0, 4);
      queryParams[2] = new Date(103, 01, 12);

      // Open the template
      QbChart chart = new QbChart (parent, // container
                      "../templates/ObjectArray.cht", // template
                      queryParams); // Query Parameters

      // Show chart in Viewer
      return chart;

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ObjectArray.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/ObjectArray.png]

3.9.5.2. Applying a Chart Template

When a QbChart object is created from scratch (see Appendix 3.C - Creating the Chart), the chart is created using default attributes. However, you can use a chart template (.tpl) to specify user defined attributes during chart construction. Almost all the attributes (except for data source and chart type) are extracted from the template and applied to the `QbChart` object. The template name usually appears as the last argument in the `QbChart` constructors.

You can also specify the template name using the `applyTemplateFile(String fileName)` method in the `QbChart` class.

The following example, which can be run as a applet(More in: Section 1.2.7 - Run Applets in WebStart with JNLP file) or application, applies a template onto the `QbChart` object:

```
Component doApplyingTemplate(Applet parent) {

      // Do not use EspressManager
      QbChart.setEspressManagerUsed(false);

      String templateLocation = "..";

      // Apply the template
      QbChart chart = new QbChart (parent, // container
                      QbChart.VIEW2D, // chart dimension
                      QbChart.BAR, // chart type
                      data, // data
                      columnMapping, // column mapping
                      templateLocation); // template

      // Show chart in Viewer
      return chart;

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ApplyingChartTemplate.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/ApplyingChartTemplate.png]

Please note that the above code is not complete and is only there as a guide. However, the link contains a complete application code that can be run.

You can also take the column mapping from the template and have it applied on the `QbChart` object being created. This is done by passing in **`null`** instead of a `ColInfo` object.

3.9.5.3. Modifying Data Source

You can create chart templates in Designer and open those templates using the API. The `QbChart` object created uses the same data source as the template and attempts to fetch the data. However, it may be that while the template has all the look and feel needed, the data source may be an incorrect one. The following sections show how to switch the data source, without recreating the entire chart.

Please note that for best results, the number of columns and the data type of each column must match between the two data sources (i.e., the one used to create the template in Designer and the new data source).

After switching the data source, the `QbChart` object must be forced to fetch the new data. This can be done by calling the refresh method in the `QbChart` class.

3.9.5.3.1. Data from a Database

Switching the data source to point to a database is simple. All you would need to do is provide the database connection information as well as the query to be used and pass that to the `QbChart` object. You can provide the database connection information (as well as the query) in a `DBInfo` object (for more information on creating a `DBInfo` object, please refer to Appendix 3.B.1 - Data from a Database).

The following example, which can be run as a applet(More in: Section 1.2.7 - Run Applets in WebStart with JNLP file) or application, switches the data source of the `QbChart` object to a database:

```
Component doChartSwitchToDatabase(Applet parent) {

      // Do not use EspressManager
      QbChart.setEspressManagerUsed(false);

      // Open the template
      QbChart chart = new QbChart (parent, // container
                      "SwitchToDatabase.cht"); // template

      // New database connection information
      DBInfo newDatabaseInfo = new DBInfo(.....);

      try {
            // Switch data source
            chart.gethInputData().setDatabaseInfo(newDatabaseInfo);

            // Refresh the chart
            chart.refresh();

      } catch (Exception ex)
      {
            ex.printStackTrace();

      }

      // Show chart in Viewer
      return chart;

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ChartSwitchToDatabase.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/ChartSwitchToDatabase.png]

Please note that the above code is not complete and is only there as a guide. However, the link contains a complete application code that can be run.

3.9.5.3.1.1. JNDI

You can also change the data source to a JNDI data source. This is done by specifying the JNDI connection information in a `DBInfo` object and then passing it to the `QbChart` object.

The following example, which can be run as a applet(More in: Section 1.2.7 - Run Applets in WebStart with JNLP file) or application, switches the data source of the `QbChart` object to a JNDI database:

```
Component doChartSwitchToDatabaseJNDI(Applet parent) {

      // Do not use EspressManager
      QbChart.setEspressManagerUsed(false);

      // Open the template
      QbChart chart = new QbChart (parent, // container
                      "ChartSwitchToDatabaseJNDI.cht"); // template

      // New database connection information
      DBInfo newDatabaseInfo = new DBInfo(.....);
```

```
        try {
              // Switch data source
              chart.gethInputData().setDatabaseInfo(newDatabaseInfo);

              // Refresh the chart
              chart.refresh();

        } catch (Exception ex)
        {
              ex.printStackTrace();

        }

        // Show chart in Viewer
        return chart;

  }
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ChartSwitchToDatabaseJNDI.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/ChartSwitchToDatabaseJNDI.png]

Please note that the above code is not complete and is there only as a guide. However, the link contains a complete application code that can be run. Note that for the application code to run, the Woodview database needs to be set up as a JNDI data source in the Tomcat environment and the application code changed to match the connection information.

3.9.5.3.2. Data from a Data File (TXT/DAT/XML)

You can switch the data source to a text file as long as the text file follows the Quadbase guidelines (for more details, please refer to Appendix 3.B.2 - Data from a Data file (TXT/DAT/XML)).

The following example, which can be run as a applet(More in: Section 1.2.7 - Run Applets in WebStart with JNLP file) or application, switches the data source of the `QbChart` object to a text file:

```
Component doChartSwitchToDataFile(Applet parent) {

      // Do not use EspressManager
      QbChart.setEspressManagerUsed(false);

      // Open the template
      QbChart chart = new QbChart (parent, // container
                      "../templates/ChartSwitchToDataFile.cht"); // template

      try {
            // Switch data source
            chart.gethInputData().setDataFile(....);

            // Refresh the chart
            chart.refresh();

      } catch (Exception ex)
      {
            ex.printStackTrace();

      }

      // Show chart in Viewer
      return chart;
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ChartSwitchToDataFile.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/ChartSwitchToDataFile.png]

Please note that the above code is not complete and is there only as a guide. However, the link contains a complete application code that can be run.

3.9.5.3.3. Data from an XML Data Source

You can switch the data source to your custom XML data as long as there is a .dtd or .xml schema accompanying your data. The XML data information is specified (for more details, please refer to Appendix 3.B.3 - Data from a XML Data Source) and then passed to the `QbChart` object.

The following example, which can be run as a applet(More in: Section 1.2.7 - Run Applets in WebStart with JNLP file) or application, switches the data source of the `QbChart` object to XML data:

```
Component doSwitchToXMLData(Applet parent) {

      // Do not use EspressManager
      QbChart.setEspressManagerUsed(false);

      // Open the template

      QbChart chart = new QbChart (parent, // container
                      "../templates/ChartSwitchToXMLData.cht"); // template

      // XML data source information
      XMLFileQueryInfo newData = new XMLFileQueryInfo(...);

      try {
            // Switch data source
            chart.gethInputData().setXMLFileQueryInfo(newData);

            // Refresh the chart
            chart.refresh();

      } catch (Exception ex)
      {
            ex.printStackTrace();

      }

      // Show Chart in Viewer
      return chart;

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ChartSwitchToXMLData.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/ChartSwitchToXMLData.png]

Please note that the above code is not complete and is there only as a guide. However, the link contains a complete application code that can be run.

3.9.5.3.4. Data from Custom Implementation

In addition to the regular data sources, you can also pass in your own custom data. The custom data is passed to the `QbChart` object using the `IDataSource` interface (for more details, please refer to Appendix 3.B.5 - Data Passed in your Custom Implementation).

The following example, which can be run as a applet(More in: Section 1.2.7 - Run Applets in WebStart with JNLP file) or application, switches the data source to a custom implementation:

```
Component doSwitchToCustomData(Applet parent) {

      // Do not use EspressManager
      QbChart.setEspressManagerUsed(false);

      // Open the template
      QbChart chart = new QbChart (parent, // container
                      "ChartSwitchToCustomData.cht"); // template

      try {
            // Switch data source
            chart.gethInputData().setClassFile(..);

            // Refresh the chart
            chart.refresh();

      } catch (Exception ex)
      {
            ex.printStackTrace();

      }

      // Show chart in Viewer
      return chart;

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ChartSwitchToCustomData.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/ChartSwitchToCustomData.png]

Please note that the above code is not complete and is there only as a guide. However, the link contains a complete application code that can be run.

3.9.5.3.5. Data passed in an Array in Memory

You can also pass in data using arrays. The array data is usually stored is memory and passed to the `QbChart` object (for more details, please refer to Appendix 3.B.4 - Data Passed in an Array in Memory).

The following example, which can be run as a applet(More in: Section 1.2.7 - Run Applets in WebStart with JNLP file) or application, switches the data source to an array in memory:

```
Component doSwitchToArrayData(Applet parent) {

      // Do not use EspressManager
      QbChart.setEspressManagerUsed(false);

      // Open the template
      QbChart chart = new QbChart (parent, // container
                      "ChartSwitchToArrayData.cht"); // template

      // Create array data
      DbData newData = new DbData(...);

      try {
            // Switch data source
            chart.gethInputData().setData(newData);

            // Refresh the chart
            chart.refresh();
```

```
        } catch (Exception ex)
        {
              ex.printStackTrace();

        }

        // Show chart in Viewer
        return chart;

  }
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ChartSwitchToArrayData.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/ChartSwitchToArrayData.png]

Please note that the above code is not complete and is there as a guide. However, the link contains a complete application code that can be run.

3.9.5.4. Modifying Chart Attributes

EspressReport has hundreds of properties, which provide a fine control over various elements of a chart. In order to facilitate ease-of-use, most properties have been categorized into groups and exposed in the form of interfaces. An application first obtains a handle to a group interface using a `gethXXX` method and then manipulates the chart's properties directly by calling methods on that interface. Most interfaces are contained in the quadbase.util [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/package-summary.html] and quadbase.ChartAPI [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/ChartAPI/package-summary.html] packages.

3.9.5.4.1. Modifying Color, Font, ...

A chart is comprised of different elements (for instance, the canvas, the axis, the legend, the chart data, etc). Each element can be modified independently of the other by getting the appropriate interface and changing the properties by calling the methods therein.

For instance, the following code sets the background color of the chart to white:

```
chart.gethCanvas().setBackgroundColor(Color.white);    // chart being an
 object of type QbChart
```

while the code given below changes the color of the X axis to black:

```
chart.gethXAxis().setColor(Color.black);              // chart being an
 object of type QbChart
```

You can also set the color of the data elements of the chart based on the series or based on the category (if no series is defined). For example, if the chart is a Column chart with 5 elements in the series, the following code sets the columns colors in the series, to be yellow, orange, red, green and blue.

```
Color dataColors[] = {Color.yellow, Color.orange, Color.red, Color.green,
 Color.blue};
chart.gethDataPoints().setColors(dataColors);        // chart being an
 object of type QbChart
```

Note that the number of colors defined **must** match the number of unique elements in the series (or the category if the series is not defined).

Similarly, the font styles can be set by calling the appropriate interface and using the method. The following code sets the text used in the legend to be of Arial, bold type and size 15.

```
Font legendFont = new Font("Arial", Font.BOLD, 15);
```

```
chart.gethLegend().setFont(legendFont);           // chart being an object
 of type QbChart
```

while the following code sets the font and color of the labels used in the Y Axis:

```
Font YAxisFont = new Font("Helvetica", Font.ITALIC, 15);
chart.gethYAxis().gethLabel().setFont(YAxisFont);    // chart being an
 object of type QbChart
chart.gethYAxis().gethLabel().setColor(Color.blue);  // chart being an
 object of type QbChart
```

Similarly, other properties can be set using the methods in the interfaces.

3.9.5.4.2. Setting Predefined Patterns

The class `QbPattern` is a subclass of `java.awt.Color`, so there is no new method introduced to set patterns explicitly. You create a `QbPattern` object first, and then use it as if it is a `Color` object. The pattern feature is only applicable to data points. For other chart elements (such as axis, canvas), EspressReport will ignore the `QbPattern` properties, and use it just like a `Color`.

The following example, which can be run as a applet(More in: Section 1.2.7 - Run Applets in WebStart with JNLP file) or application, shows how to set a pattern to a data point:

```
Color[] dataColors = chart.gethDataPoints().getColors();
QbPattern dataPattern = new QbPattern(colors[2],
 QbChart.PATTERN_THICK_FORWARD_DIAGONAL);
dataColors[2] = dataPattern;
chart.gethDataPoints().setColors(dataColors);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ChartSetPredefinedPattern.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/ChartSetPredefinedPattern.png]

There are totally 30 predefined patterns available for you to use. The pattern ID range from 0 to 29. If the user passes an integer beyond this range, it will be considered as ID = 0, which is a solid color block. The Pattern ID is defined in quadbase.ChartAPI.IMiscConstants [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/ChartAPI/IMiscConstants.html] class as well as in `quadbase.common.swing.color.PatternImage` class. Since `QbChart` implements the `IMiscConstants` interface, you can obtain these IDs directly from `QbChart`. You can also see what each pattern looks like in the pattern palette `Designer`.

3.9.5.4.3. Setting Customized Patterns

You can also set up your own pattern (texture) images. This feature is only available via API and the modification will not be saved to the template. This feature is mainly for those who want to modify the chart during the run time.

The following example, which can be run as a applet(More in: Section 1.2.7 - Run Applets in WebStart with JNLP file) or application, shows how to set a custom pattern to a data point:

```
Color[] dataColors = chart.gethDataPoints().getColors();
QbPattern dataPattern = new QbPattern(dataColors[2]);
File customImageFile = new File("Quadbase_Logo.png");
BufferedImage customImage = ImageIO.read(customImageFile);
dataPattern.setPatternImage(customImage);
dataColors[2] = dataPattern;
chart.gethDataPoints().setColors(dataColors);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ChartSetCustomPattern.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/ChartSetCustomPattern.png]

In the above code, a `QbPattern` object is created without specifying the pattern ID. The newly created object is equivalent to a `java.awt.Color object`. The developer is responsible for providing an image so as to avoid the `NullPointerException` prompting in the procedure. An existing image (from a file) is then passed as the pattern for data point 2.

3.9.5.4.4. Modifying Size

There are two sizes to be considered when creating a chart:

Relative Size This size is defined relative to the chart canvas. For instance, the chart plot height might be set as .7 and the chart plot width set as .8 . In the case where the canvas is 300 by 300 pixels, the height of the chart plot becomes 210 (.7 x 300) pixels and the width becomes 240 pixels (or .8 x 300). If the canvas size is increased to 500 by 600 pixels, the height of the chart plot becomes 420 pixels (.7 * 600) and the width becomes 400 pixels (.8 * 500). Relative sizes depend on the canvas height and width.

Absolute Size This size denotes an absolute size of the canvas in pixels.

Every element of the chart (where the size parameter is used) is defined relative to the chart canvas. A float is used to represent the relative size of the chart element. For instance, the following code sets the chart plot height to .85 and the chart plot width to .65:

```
chart.gethChartPlot().setRelativeHeight(.85f);    // chart being an object
 of type QbChart
chart.gethChartPlot().setRelativeWidth(.65f);     // chart being an object
 of type QbChart
```

The code given below sets the canvas size of the chart:

```
Dimension canvasSize = new Dimension(500, 450);
chart.gethCanvas().setSize(canvasSize);           // chart being an object
 of type QbChart
```

Similarly, the sizes of the other elements can be set using the methods in the interfaces.

3.9.5.4.5. Modifying Date/Time Zoom Charts

You can modify the properties (such as scale and starting/ending time perion) of a Date/Time Zoom template created in the Designer. This is done by getting a handle to the Zoom properties (using the `IZoomInfo` interface) and using the various methods there to change the presentation.

The following example, which can be run as a applet(More in: Section 1.2.7 - Run Applets in WebStart with JNLP file) or application, shows how to modify a Date/Time Zoom chart:

```
// Modify starting and ending period and scale
IZoomInfo zoomInfo = chart.gethZoomInfo();

// Begin Date
Calendar beginDate = new GregorianCalendar(2001, 0, 1);

// End Date
Calendar endDate = new GregorianCalendar(2003, 11, 31);

zoomInfo.setLowerBound(beginDate.getTime());
zoomInfo.setUpperBound(endDate.getTime());

zoomInfo.setScale(6, IZoomInfo.MONTH);

chart.refresh();
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ZoomChart.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/ZoomChart.png]

In the above code, chart is an object of type `QbChart`. Note that after modifying the chart's zoom properties, the chart **must** be refreshed for the modified properties to take effect.

3.9.5.4.6. Modify Heatmap Chart Settings

You can change the heatmap chart colormap with the following code:

> **Note**
>
> The Gradient class contains predefined color sets.

```
// set colormap
chart.setHeatmapColors(Gradient.YlGn); // default is Gradient.Blues
chart.setCmapLen(56); // default is 256
```

You can also define a custom color set in addition to using the predefined color sets. For example:

```
Color[] Red_Gray = { Color.RED, Color.GRAY };
```

To add and configure the data border in heatmap chart:

```
// set data border
chart.showDataBorder(true); // default is false
chart.setDataBorderThickness(2); // default is 1
chart.setDataBorderColor(Color.lightGray); // default is black
```

To show the data value in heatmap chart:

```
// set data value display
chart.getChart().showTopValue= true; // default is false
chart.getChart().setTopValueFont(new Font(Font.SERIF, Font.ITALIC,
 7),10); // default is Dialog
```

3.9.5.4.7. Modifying Chart in Report

While stand-alone charts can be modified directly using the API, any charts in a report (whether the chart is independent or not) cannot be modified as easily. You can change the look and feel of a chart (in a report) by implementing `IChartModifier` interface to create a class that invokes the desired changes. Note that the actual chart template is not changed, merely the chart's appearance within the report.

The following example, which can be run as a applet(More in: Section 1.2.7 - Run Applets in WebStart with JNLP file) or application, shows how to modify a chart in a report:

```
ReportChartObject[] reportCharts = report.getReportChartObjects();
for (int i = 0; i < reportCharts.length; ++i)
{

      reportCharts[i].setChartModifier(new ModifyChart());
}

public class ModifyChart implements IChartModifier {

      public IChart modifyChart(Object chartInfo) {
            ChartObject chart = new ChartObject(chartInfo); // Get actual
 ChartObject
            chart.gethCanvas().setBackgroundColor(Color.white);
            return chart; }

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ExampleIChartModifier.zip]

Exported Results Before Modification [https://data.quadbase.com/Docs71/help/manual/code/export/IChartModifier_Before.pdf]

Exported Results After Modification [https://data.quadbase.com/Docs71/help/manual/code/export/IChartModifier_After.pdf]

3.9.5.5. Exporting the Chart

Any charts included in the report can be exported into JPEG, GIF and PNG formats as well; although by default, the charts in the report are exported as JPEG's. The format for the chart can be changed, when creating the `ReportChartObject` object, by specifying the image type use the method `setImageType(int)`.

The EspressReport API has the capability to export stand-alone charts in a variety of formats. These include GIF, JPEG, PNG, BMP and PDF formats. In addition, a chart may be exported to the proprietary .cht or .tpl formats. A .cht stores all the chart information and can be considered the EspressReport equivalent of a static image file. A .cht file differs from a static image file in that its data can be refreshed from the data source and it allows additional functionality such as zooming, drill-down etc A .tpl file is identical to a .cht file except it doesn't store actual data. For a .tpl file, the data is automatically reloaded from the original data source at the time of chart reload. Both .cht and .tpl files can be viewed using the EspressReport Viewer or EspressReport API. When using stand-alone charts, you can specify the format by creating a `QbChart` object and using the various export methods within the class.

There are different methods available for exporting a stand-alone chart in a variety of formats with different options. The simplest method would be:

```
public export(int format, String filename)
```

In the above method, format is one of the format constants and filename is the output filename (with or without an extension).

The following is a list of the constants used for exporting the chart components:

Bitmap	QbChart.BMP
GIF	QbChart.GIF
JPEG	QbChart.JPEG
PNG	QbChart.PNG
PDF	QbChart.PDF
Flash	QbChart.FLASH
Scalable Vector Graphics	QbChart.SVG
Text Data	QbChart.TXTFORMAT
XML Data	QbChart.XMLFORMAT

Windows Meta File	QbChart.WMF

Depending on the format selected, additional options are also available. For example, exporting a chart object as a jpeg gives you the choice of the image quality (represented as number from 0 - 99) while export a chart object as a PNG gives you the choice of specifying the compression used. The options can be specified using the following method:

```
public export(int format, String filename, int option)
```

The following code, which can be run as an applet or application, shows how to construct and export a chart:

```
// Open the template
QbChart chart = new QbChart(parent, "ExportChart.cht");

try {

      chart.export(QbChart.PNG, "ExportChart");

} catch (Exception ex) {

      ex.printStackTrace();

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ExportChart.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/ExportChart.png]

Please note that when you export the chart to a text file, the default delimiter is a tab space. However, you can set another delimiter (available delimiters are “,”, “;” and “ ”) by using the following method:

```
chart.exportDataFile("TextData", QbChart.COMMA, QbChart.TXTFORMAT);    //
 where chart is an object of type QbChart
```

If you plan on exporting the chart object without viewing it, setting the following static method to true would improve performance slightly:

```
QbChart.setForExportOnly(boolean);
```

Calling this method before constructing the `QbChart` object would result in better performance.

You can also export the `QbChart` object to a byte array using the following method:

```
public byte[] exportChartToByteArray();
```

This will give the .cht equivalent in the form of a byte array. This is useful if you need to serialize the `QbChart` object.

3.9.5.5.1. Record File Exporting

EspressReport also has record file exporting to handle large amounts of data. In record file exporting, you specify the number of records to be held in memory and a temp directory. When the number of records in the data exceeds the number specified to be held in memory, the data is stored on disk in the temp directory specified.

There are certain conditions that have to be met before you can use this feature. You must:

- make sure that EspressManager is not used;
- make sure that the data is NOT from a XML source;

- make sure that the data is sorted;

To use record file export, the following code must be added before calling the `QbChart` constructor:

```
QbChart.setEspressManagerUsed(false);
QbChart.setTempDirectory(<specify the temp directory to store data. Default
 is ./temp>);
QbChart.setMaxCharForRecordFile(<specify the maximum character length.
 Default is 40>);
QbChart.setMaxRecordInMemory(<specify the maximum number of rows to be kept
 in memory. Default is -1 i.e., store all records in memory>);
QbChart.setFileRecordBufferSize(<specify the number of rows to be retrieved
 at one time from disk. Default is 10,000>);
```

3.9.5.5.2. Streaming Charts

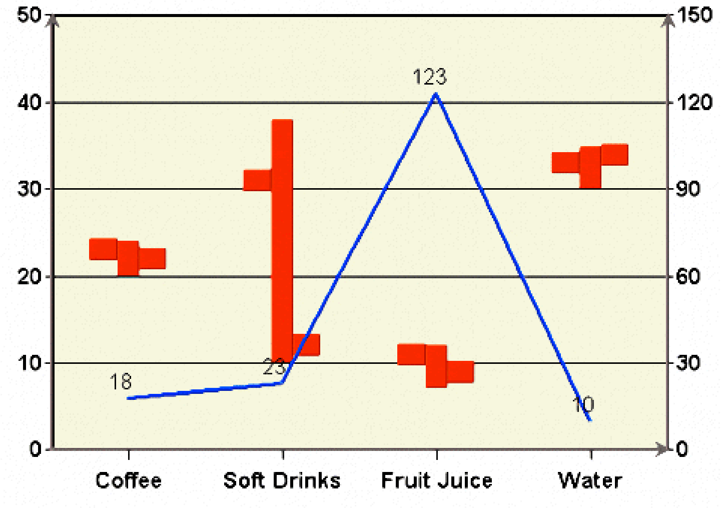
In addition to exporting to local drives, charts can also be exported as a byte stream and streamed directly to a web browser. Note that server side code typically exports the image only in a binary format. You are responsible for creating a wrapper (i.e., DHTML/HTML content) around the exported image.

Given below is an example that exports a chart to PNG and streams it to the browser. In order to run the example, you will need to configure and compile the source code, then deploy the class file in the servlet directory of your servlet runner. Replace the `chartTemplate` variable with either an absolute path or a path relative to the working directory of your application server.

```
public void doGet(HttpServletRequest req, HttpServletResponse resp) throws
 ServletException, IOException {

      // Do not use EspressManager
      QbChart.setEspressManagerUsed(false);

      String chartTemplate = "StreamingChart.cht";

      // Open template
      QbChart chart = new QbChart((Applet)null, chartTemplate);

      // Export the chart to PNG and stream the bytes
      ByteArrayOutputStream chartBytes = new ByteArrayOutputStream();

      try {
            chart.export(QbChart.PNG, chartBytes);

      } catch (Exception ex)
      {
            ex.printStackTrace();

      }

      resp.setContentType("image/x-png");
      resp.setContentLength(chartBytes.size());

      OutputStream toClient = resp.getOutputStream();
      chartBytes.writeTo(toClient);
      toClient.flush();
      toClient.close();

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/StreamingChart.zip]

To run the example, make sure to copy the chart template to the location accessed by the code. If using a relative path (as shown in the example), remember that the current directory is the working directory of your application server. For example, since the working directory in Tomcat is `<Tomcat>/bin`, you will need to create the directory `<Tomcat>/templates/` and copy the chart template there to run the code. The resulting chart is shown below.



Streaming Chart

3.9.6. API Only Features

While EspressReport API can reproduce all the functionality of Chart Designer, the API also has additional functionalities. These additional features will be described below. Please note that some of the features are for standalone charts only.

Note

Note that in the snippets of code provided, chart is an object of type `QbChart`.

3.9.6.1. Visual

The features, shown below, deal with the presentation of the chart and its components. Each feature below changes the chart, in some manner, visually. These changes are also carried forth when exported to a static image.

3.9.6.1.1. Canvas/Plot

The following features describe the various visual changes that can be made to the chart plot and/or canvas using the API. Note that the features below deal with general changes to the chart plot and/or canvas. Any component specific change is described in its own section.

3.9.6.1.1.1. Customizable Message for No-Data-In-Plot

EspressReport allows the message, which gets displayed when there is no data or not enough data to be plotted, to be changed. This can be done by getting a handle to the `INoDataToPlotMessage` and specifying the new message.

```
INoDataToPlotMessage noData = chart.gethNoDataToPlotMessage();
noData.setMessage("Not enough data to plot chart");
```

Please refer to the online API documentation for more information (quadbase.util.INoDataToPlotMessage [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/INoDataToPlotMessage.html]).

3.9.6.1.1.2. Set Chart to Fit Canvas

EspressReport allows charts to be resized and fit into the canvas correctly, taking into account all the text and labels associated with the chart. To correctly fit a chart onto its canvas, use the method `setFitOnCanvas(boolean b)` in the `ICanvas` interface.

```
ICanvas canvas = chart.gethCanvas();
canvas.setFitOnCanvas(true);
```

Please refer to the online API documentation for more information (quadbase.util.ICanvas [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/ICanvas.html]).

3.9.6.1.1.3. Set Chart Invisible

A chart can be made invisible so that only the plot data in table form is shown. This is accomplished by getting a handle to the `ICanvas` interface and using the `setChartVisible` method.

```
ICanvas canvas = chart.gethCanvas();
canvas.setChartVisible(false);
```

Please refer to the online API documentation for more information (quadbase.util.ICanvas [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/ICanvas.html]).

3.9.6.1.1.4. Applying Different Graphics Rendering

EspressReport allows you to apply different rendering techniques (such as anti-aliasing) to the chart. This allows you to draw the chart to your specifications.

To utilize this feature in the API, call the `setRenderingHint` method in `QbChart` and pass in the hint key and hint value parameters.

```
chart.setRenderingHint(java.awt.RenderingHints.KEY_ANTIALIASING,
 java.awt.RenderingHints.VALUE_ANTIALIAS_ON);
```

You can also specify the horizontal text to be anti-aliased only.

```
chart.forceApplyAntiAliasToHorizontalText(true);
```

Please refer to the online API documentation for more information (quadbase.ChartAPI.QbChart [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/ChartAPI/QbChart.html]).

This functionality is only available for stand-alone charts.

3.9.6.1.1.5. Chart Plot Position

EspressReport allows the chart position to be placed down to -0.5 (x and y position relative to the canvas). This allows the chart to be placed right along the edge of the canvas.

To utilize this feature using the API, get a handle to `IPlot` and use the `setPosition` method.

```
IPlot chartPlot = chart.gethChartPlot();
chartPlot.setPosition(new Position(-0.5, -0.5));
```

Please refer to the online API documentation for more information (quadbase.util.IPlot [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IPlot.html]).

3.9.6.1.1.6. Drawing Multiple Charts in same Plot

EspressReport allows multiple charts to be overlapped and shown one above the other (using the primary chart's axes). The canvas of the primary chart is used and therefore 2D charts cannot be added to 3D charts and vice versa. All charts overlapped onto the primary chart will have any text and canvas information removed. Please note that since resulting chart will use the primary chart's category and scale, add on charts must also use the similar categories and scales to be displayed correctly. For Scatter and Surface charts, both x and y axis scales should be taken into consideration.

To utilize this feature using the API, you would use the `setAddOnChart` method in `QbChart`.

The following code, which can be run as an applet or application, shows how to overlap 3 charts in one plot:

```
// Open the template
QbChart primaryChart = new QbChart(parent,
      // container
                                    "../templates/AddOnChart1.cht"); //
 template

// Open other two templates
QbChart chart2 = new QbChart(parent, "../templates/AddOnChart2.cht");
QbChart chart3 = new QbChart(parent, "../templates/AddOnChart3.cht");

primaryChart.setAddOnChart(new QbChart[] {chart2, chart3});
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/AddOnChart.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/AddOnChart.png]

Please refer to the online API documentation for more information (quadbase.ChartAPI.QbChart [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/ChartAPI/QbChart.html]).

3.9.6.1.2. Hint Box

The following features describe the various visual changes that can be made to the chart hint box using the API. These include modifying the content as well as look of the hint box.

3.9.6.1.2.1. Modify Hint Box

EspressReport allows the Hint Box to be modified, i.e., you can dictate what the Hint Box says when it is viewed. This is done by creating a class that implements `IHintBoxInfo` and assigning that class using the method `setHintBoxInfo` in `IHint`.

The following code, which can be run as an applet or application, shows how to modify the hint box info:

```
// Open the template
QbChart chart = new QbChart(parent,
  // container
                             "../templates/ModifyHintBox.cht"); // template

IHintBoxInfo hintBoxInfo = new Hint();
chart.gethDataPoints().gethHint().setHintBoxInfo(hintBoxInfo);

...

class Hint implements IHintBoxInfo {

      public Vector getHint(PickData pickData) {
            Vector vec = new Vector();

            if (pickData.series != null)
                  vec.addElement("(" + pickData.seriesName+ ") " +
 pickData.s_series);

            if (pickData.category != null)
                  vec.addElement("(" + pickData.categoryName + ") " +
 pickData.s_category);

            if (pickData.s_value != null)
                  vec.addElement("(" + pickData.valueName + ") " +
 pickData.s_value + (pickData.value > 7 ? ":Good" : ":Restock"));

            return vec;

      }
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ModifyHintBox.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/ModifyHintBox.png]

Please refer to the online API documentation for more information (quadbase.util.IHintBoxInfo [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IHintBoxInfo.html]).

This functionality is only available for stand-alone charts.

3.9.6.1.2.2. Data and Hyperlink Hint Box Offset

EspressReport allows an offset to be set for the Data and the HyperLink hint box so that it doesn't overlap the chart or any other component in the chart.

To utilize this feature in the API, get a handle to `IHint` (from either the chart object or the `IHyperLinkSet` object) and then use the `setOffset` method.

```
IHint dataHints = chart.gethDataPoints().gethHint();
dataHints.setoffset(new Dimension (30, 20));
```

Please refer to the online API documentation for more information (quadbase.util.IHint [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IHint.html]).

3.9.6.1.2.3. Hint Box Border Color

EspressReport allows the color of the Hint box border to be set using the API. This can be done by getting a handle to `IHint` and using the `setBorderColor` method.

```
IHint chartHintBox = chart.gethHint();
chartHintBox.setBorderColor(Color.red);
```

Please refer to the online API documentation for more information (quadbase.util.IHint [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IHint.html]).

3.9.6.1.2.4. Customize Image Map Hint Box

EspressReport allows the customization of the hint box text when exporting the chart to a map file. When the map file is included in a DHTML/HTML file, the customized hint box is visible when the mouse is moved over the data points of the chart. You can create a customized image map hint box by creating a class that implements `ICustomizedImageDataMapHintBox` and assign that class to the `setImageMapDataHintBoxHandle` method in `QbChart`.

The following code, which can be run as an applet or application, shows how to customize the image map hint box:

```
// Open the template
QbChart chart = new QbChart(parent,
  // container
                            "../templates/CustomizeImageMapHintBox.cht"); //
 template

chart.setImageMapDataHintBoxHandle(new customizeImageMap());

try {

      chart.export(QbChart.PNG, "../export/CustomizeImageMapHintBox", "../
export/CustomizeImageMapHintBox", 0, 0, true);

} catch (Exception ex) {

      ex.printStackTrace();
}

...

class customizeImageMap implements ICustomizeImageMapDataHintBox {

      public String customizeHintBox(String str) {
            return str.substring(6, 11) + " " +
 str.substring(str.length()-3);

      }

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/CustomizeImageMapHintBox.zip]

Exported Map [https://data.quadbase.com/Docs71/help/manual/code/export/CustomizeImageMapHintBox.map]

Exported Image [https://data.quadbase.com/Docs71/help/manual/code/export/CustomizeImageMapHintBox.html]

Please refer to the online API documentation for more information (quadbase.util.ICustomizeImageMapDataHintBox [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/ICustomizeImageMapDataHintBox.html]).

3.9.6.1.3. Legend/Annotation

The following features describe the various visual changes that can be made to the chart legend and/or any annotation using the API. These include modifying the content as well as look of the chart legend/annotations.

3.9.6.1.3.1. Annotation with Symbol

EspressReport allows symbols to be included with an Annotation thus allowing symbols and strings to be placed in the chart. One of the applications of this feature is to create your own legend in place of the default legend created by EspressReport.

A new constructor has been created for `IAnnotation`, which can be used for adding symbols and text.

The following code, which can be run as an applet or application, shows how to combine symbols with an Annotation:

```
// Open the template
QbChart chart = new QbChart(parent,
        // container
                                    "../templates/AnnotationWithSymbol.cht"); //
 template

// Create custom legend
IAnnotationSet set = chart.gethAnnotations();
String[] text = {"New Legend", "Hello World", "ABC", "I got It"};
int[] shape = {QbChart.PLUS, QbChart.NOSYMBOL, QbChart.SQUARE,
 QbChart.DASH};
Color[] color = {Color.red, Color.black, Color.blue, Color.white};
IAnnotation anno = set.newAnnotation(text, shape, color);
Point_2D newPosition = new Point_2D(.7f, .7f);
anno.setRelativePosition(newPosition);
set.addAnnotation(anno);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/AnnotationWithSymbol.zip]

Exported Image [https://data.quadbase.com/Docs71/help/manual/code/export/AnnotationWithSymbol.png]

Please refer to the online API documentation for more information (quadbase.util.IAnnotationSet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IAnnotationSet.html] and quadbase.util.IAnnotation [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IAnnotation.html]).

3.9.6.1.3.2. Set Relative Shift of Annotation Border

EspressReport allows shift of Annotation border from Relative Position of Annotation and Annotation text. The following methods allows you to set Shift in x,y axis in pixels.

```
void setxShift(int x);
void setyShift(int y);
```

Please refer to the online API documentation for more information (quadbase.util.IAnnotation [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IAnnotation.html]).

3.9.6.1.3.3. Set Legend Title

EspressReport allows you to add the legend title if the chart doesn't have a secondary axis. The following methods allow you to set the Title of legend.

```
ILegend hLegend = chart.gethLegend();
hLegend.setTitle("Day");
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/LegendTitle.zip]

Exported Image [https://data.quadbase.com/Docs71/help/manual/code/export/LegendTitle_exported.png]

Please refer to the online API documentation for more information(quadbase.util.ILegend [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/ILegend.html]).

3.9.6.1.3.4. Set LegendLineSpacing

EspressReport allows you to increase the line spacing of the primary Legend with the following API lines, where "15" is the spacing width in pixels. The following methods allow you to set the Line Spacing of the primary legend.

```
ILegend hLegend = chart.gethLegend();
hLegend.setLineSpacing("15");
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/LegendSpacing.zip]

Exported Image [https://data.quadbase.com/Docs71/help/manual/code/export/LegendSpacing_exported.png]

Please refer to the online API documentation for more information(quadbase.util.ILegend [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/ILegend.html]).

3.9.6.1.4. Misc.

The following features describe the various visual changes that can be made to the different components of the chart. These include modifying the content as well as look of the chart and its elements.

3.9.6.1.4.1. Ticker Label Replacement

EspressReport allows ticker labels to be replaced by user-defined strings. This enables the users to put in their own ticker values.

To replace ticker labels, use the method `setTickerLabels` in `IAxis`.

```
IAxis hXAxis=chart.gethXAxis();
hXAxis.setTickerLabels(new String[] {new String("a"), new String("b"), new
 String("c")});
```

Please refer to the online API documentation for more information (quadbase.util.IAxis [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IAxis.html]).

3.9.6.1.4.2. Customizable Data Top Label

EspressReport allows data top labels for both primary and secondary charts to be customized. You can do this by creating a class that implements `IDataLabelInfo` and passing that class using the `setDataLabelInfo` method in `IDataPointSet`.

The following code, which can be run as an applet or application, shows how to combine symbols with an Annotation:

```
// Open the template
QbChart chart = new QbChart(parent,
         // container
                            "../templates/CustomizeDataTopLabel.cht"); //
 template

chart.gethDataPoints().setDataLabelInfo(new customizeDataTopLabel());

...

class customizeDataTopLabel implements IDataLabelInfo {

      public String getDataLabel(PickData pickData, String
 originalDataLabel) {
            return (pickData.toString() + " (" + originalDataLabel + ")")

      }

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/CustomizeDataTopLabel.zip]

Exported Image [https://data.quadbase.com/Docs71/help/manual/code/export/CustomizeDataTopLabel.png]

Note that any customization to the data top labels will not be evident unless they are visible.

Please refer to the online API documentation for more information (quadbase.util.IDataLabelInfo [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IDataLabelInfo.html]).

3.9.6.1.4.3. Show Axis as Date/Time/Timestamp

EspressReport allows the value axis to be shown in units of time/date (instead of numeric units) for any chart, which has a value axis. Note that the data for the value axis must still be numeric.

To utilize this feature using the API, get a handle to `IAxis` and use the `setDisplayLabelAsDate` method.

```
IAxis chartYAxis = chart.gethXAxis();
chartXAxis.setDisplayLabelAsDate(true);
```

Please refer to the online API documentation for more information (quadbase.util.IAxis [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IAxis.html]).

3.9.6.1.4.4. Selective String Rendering

Strings, which are drawn horizontally or vertically, look better when not anti-aliased. However, strings at other angles look better when anti-aliased. EspressReport allows certain text to be anti-aliased and other strings (0 and 90 degree angles) to be drawn without anti-aliasing.

To utilize this feature using the API, get a handle to `IDataPointSet` and use the `setDisableJava2DForStraightText` method.

```
IDataPointSet dataPoints = chart.gethDataPoints();
dataPoints.setDisableJava2DForStraightText(true);
```

Please refer to the online API documentation for more information (quadbase.util.IDataPointSet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IDataPointSet.html]).

3.9.6.1.4.5. Drawing Data Points above Horizonval/Vertical/Trend Lines

EspressReport allows the data points to be drawn on top of any horizontal/vertical/trend lines i.e., it appears like the data points are resting on top of the line instead of being overshadowed by the line.

To utilize this feature using the API, get a handle to `ILinePropertySet` and use the method `setDataDrawnOnTop`.

```
ILinePropertySet lineProperties = chart.gethLineProperties();
lineProperties.setDataDrawnOnTop(true);
```

Please refer to the online API documentation for more information (quadbase.util.ILinePropertySet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/ILinePropertySet.html]).

3.9.6.2. Data

The features, shown below, deal with the data in the chart and its components. Each feature below changes data shown (typically the presentation of the data shown) or obtains the data in the chart. These changes are also carried forth when exported to a static image.

3.9.6.2.1. Getting the Coordinates

EspressReport allows you to get the information of a datapoint based on a specified pixel position. This returns the category, value, series and sumby (if they are there) of the data point at the specified pixel location and otherwise returns null.

To get pickData, use the method `getPickData(width, height)` in `IHint`.

```
IDataPointSet dataPoints=chart.gethDataPoints();
IHint hint=dataPoints.gethHint();
PickData pickdata=hint.getPickData(200, 300)        // pixel at width=200
 and height=300
```

Please refer to the online API documentation for more information (quadbase.util.IHint [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IHint.html]).

3.9.6.2.2. Set Data Limit at Axis Scale

When a manual axis is applied, EspressReport gives you the option to truncate data points that are beyond the maximum value on the axis. You can set the data limit by using the method `setLimitAtAxisScale` in `IDataPointSet`.

```
IDataPointSet dataPoints = chart.gethDataPoints();
dataPoints.setLimitAtAxisScale(true);
```

Please refer to the online API documentation for more information (quadbase.util.IDataPointSet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IDataPointSet.html]).

3.9.6.2.3. Set Null Data as Zero

EspressReport can now represent any null data as zero data (i.e., datapoints with an associated 0 value). To accomplish this, use the `setNullDataAsZero` method in `IDataPointSet`.

```
IDataPointSet dataPoints = chart.gethDataPoints();
dataPoints.setNullDataAsZero(true);
```

Please refer to the online API documentation for more information (quadbase.util.IDataPointSet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IDataPointSet.html]).

3.9.6.2.4. Additional Trend Line Options

EspressReport allows you to get the minimum, the maximum, the probability as well as the inverse normal in addition to the mean value for the normal curve trend line as well as draw standard deviation trend lines, by calling `ITrendLine` and using the appropriate method.

To utilize this feature in the API, you need to call `ITrendLine` and use the appropriate methods.

Note

You can only draw standard deviation trend lines for a normal curve if the chart is a histogram chart with `setLinearScale` and `setRounded` true.

The following code, which can be run as an applet or application, shows how to get information from a normal curve trendline in the chart:

```
ITrendLine normalCurve =
 (ITrendLine)chart.gethDataLines().elements().nextElement();
System.out.println("Mean = " + normalCurve.getMean());
System.out.println("St. Dev = " + normalCurve.getStandardDev());
System.out.println("Min = " + normalCurve.getMin());
System.out.println("Max = " + normalCurve.getMax());
System.out.println("Dev of 1.0, Prob = " + normalCurve.getProbability(1.0));
System.out.println("Back Calculated Dev = " +
 normalCurve.getInverseNorm(normalCurve.getProbability(1.0)));
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/AdditionalTrendLine.zip]

Exported Image [https://data.quadbase.com/Docs71/help/manual/code/export/AdditionalTrendLine.png]

Please refer to the online API documentation for more information (quadbase.util.ITrendLine [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/ITrendLine.html]).

3.9.6.2.5. Adding Multiple Control Lines to Stack Type Chart

EspressReport allows you to draw the minimum, the maximum, the average in addition to the stand deviation lines of any level exsisting in the stack data together, by calling `newControlLine(int linetype, String label, int level)` method with proper parameters setting.

To utilize this feature in the API, you need to set the right parameter value, for the parameter of `linetype`: `MINIMUM = 12`, `MAXIMUM = 13`, `CONTROL_AVERAGE = 10`, `STANDARD_DEVIATION = 11`.

The parameter of `label` is the text dispaly in the legend.

The last parameter of `level` is the level of the data display int the stack chart.

Note

This feature only works for Stack Type Chart, i.e., stack column, stack bar and stack area. For stack column chart with combo type is stack area, this feature only works for the main axis data, that is, stack column chart.

The following code, which can be run as an applet or application, shows how to draw multiple control lines to a stack area chart:

```
QbChart chart = new QbChart(this, QbChart.VIEW2D,
 QbChart.STACKAREA, "sample.dat",colInfo);

IDataLineSet hDataLines = chart.gethDataLines();

IControlLine clLine1 = hDataLines.newControlLine(13, "Max4", 4);
IControlLine clLine2 = hDataLines.newControlLine(13, "Max3", 3);
IControlLine clLine3 = hDataLines.newControlLine(13, "Max2", 2);
IControlLine clLine4 = hDataLines.newControlLine(13, "Max1", 1);
IControlLine clLine5 = hDataLines.newControlLine(10, "Avg4", 4);
clLine1.setColor(Color.RED);
clLine2.setColor(Color.YELLOW);
```

```
clLine3.setColor(Color.GREEN);
clLine4.setColor(Color.BLUE);
clLine5.setColor(Color.GRAY);
hDataLines.add(clLine1);
hDataLines.add(clLine2);
hDataLines.add(clLine3);
hDataLines.add(clLine4);
hDataLines.add(clLine5);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/MultipleControlLines.zip]

Exported Image [https://data.quadbase.com/Docs71/help/manual/code/export/MultipleControlLines.png]

Please refer to the online API documentation for more information (quadbase.util.IControlLine [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IControlLine.html]).

3.9.6.2.6. Additional Data Line Options

EspressReport allows you to draw a constant line at a fixed value on the X or Y-axis. Additionally, in API, you can set the start point and end point of the constant line, by calling `IHorzVertLine` and using the appropriate method.

To utilize this feature in the API, you need to set the right parameter value, for the parameter of `linetype`: `MINIMUM = 12`, `MAXIMUM = 13`, `CONTROL_AVERAGE = 10`, `STANDARD_DEVIATION = 11`.

To utilize this feature in the API, you need to call `IHorzVertLine` and use the appropriate methods

The last parameter of `level` is the level of the data display int the stack chart.

```
QbChart chart = new QbChart(this, QbChart.VIEW2D,
 QbChart.STACKAREA, "sample.dat",colInfo);

IDataLineSet hDataLines = chart.gethDataLines();
     IHorzVertLine hLineA =
 hDataLines.newHorzVertLine(IHorzVertLine.VERTICAL_LINE, "hLineA");
     hLineA.setLineValue(18); // the value on X-Axis
     hLineA.setLineFromValue(36878); // the value on Y-Axis, by default,
 this is minimum value of Y-Axis
     hLineA.setLineToValue(320397);  // the value on Y-Axis, by default,
 this is maximum value of Y-Axis
     hLineA.setThickness(1);
     hLineA.setLineStyle(IDataLine.DOTTED_STYLE);
     hLineA.setColor(new Color(120, 120, 120));
     hLineA.setTitleVisibleInLegend(false);
     hDataLines.add(hLineA);

     IHorzVertLine hLineB =
 hDataLines.newHorzVertLine(IHorzVertLine.HORIZONTAL_LINE, "hLineB");
     hLineB.setLineValue(320397); // the value on Y-Axis
     hLineB.setLineFromValue(6); // the value on X-Axis, by default, this is
 minimum value of X-Axis
     hLineB.setLineToValue(18);  // the value on X-Axis, by default, this is
 minimum value of X-Axis
     hLineB.setThickness(1);
     hLineB.setLineStyle(IDataLine.DASH_STYLE);
     hLineB.setColor(new Color(120, 120, 120));
     hLineB.setTitleVisibleInLegend(false);
     hDataLines.add(hLineB);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/DataLineExtraOptions.zip]

Exported Image [https://data.quadbase.com/Docs71/help/manual/code/export/SalesByCustomerID_exported.png]

Please refer to the online API documentation for more information (quadbase.util.IDataLineSet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IDataLineSet.html]) and (quadbase.util.IHorzVertLine [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IAnnotation.html]).

3.9.6.3. Chart Specific

The features, shown below, deal with the specific chart types. Each feature below shows additional functionality available to the chart, depending on the chart type. These changes are also carried forth when exported to a static image.

3.9.6.3.1. Column/Bar Charts

The following features describe the various changes that can be made to column/bar charts using the API. These include modifying the content as well as look of the chart.

3.9.6.3.1.1. Color Separator

EspressReport allows different colors to be shown for the columns based on defined category values. To use color separator, use the method `setColorSeparators` in `IDataPointSet`.

The following code, which can be run as an application or applet, shows how to set up the color separator:

```
IDataPointSet hDataPoints=chart.gethDataPoints();
hDataPoints.setColorSeparators(new Color[]{Color.green, Color.red,
 Color.blue},
                              new Integer[]{new
 Integer((int)Math.rint(9)), new Integer((int)Math.rint(14))},
                              QbChart.ASCENDING);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ColorSeparator.zip]

Exported Image [https://data.quadbase.com/Docs71/help/manual/code/export/ColorSeparator.png]

Please refer to the online API documentation for more information (quadbase.util.IDataPointSet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IDataPointSet.html]).

3.9.6.3.1.2. Disabling Shadow

EspressReport allows the shadow that appears for the columns/bars to be rendered visible or invisible. You can use this feature by getting a handle to `IDataPointSet` and use the `setShowShadowOnPoint` method.

```
IDataPointSet dataPoints = chart.gethDataPoints();
dataPoints.setShowShadowOnPoint(false);
```

Please refer to the online API documentation for more information (quadbase.util.IDataPointSet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IDataPointSet.html]).

3.9.6.3.2. Pie Charts

The following features describe the various changes that can be made to pie charts using the API. These include modifying the content as well as look of the chart.

3.9.6.3.2.1. Drawing Pie Slices Clockwise/Counter Clockwise

EspressReport now allows the slices in a pie chart to be drawn in clockwise as well as counter clockwise order. To set the clockwise/counter clockwise option, use the `reverseOrder` method in `IDataPointSet`.

```
IDataPointSet dataPoints = chart.gethDataPoints();
dataPoints.reverseOrder(QbChart.CATEGORY);
```

Please refer to the online API documentation for more information (quadbase.util.IDataPointSet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IDataPointSet.html]).

3.9.6.3.2.2. Pie Border for 0% and 100% Slices

EspressReport offers the option of removing the pie border for slices that are 0% or 100% of the whole pie. You can do this by getting a handle to `IPiePropertySet` and use the `setRadialBorderDrawnForZero` method.

```
IPiePropertySet pieProperties = chart.gethPieProperties();
pieProperties.setRadialBorderForZero(true);
```

Please refer to the online API documentation for more information (quadbase.util.IPiePropertySet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IPiePropertySet.html]).

3.9.6.3.2.3. Customize Separator between Category and Percent Value Strings in Pie Legend

EspressReport allows user-defined separators to be placed between the Category and Percent Value Strings in the legends for Pie Charts. This can be done by getting a handle to `IPiePropertySet` and use the `setSepSymbol` method.

```
IPiePropertySet pieProperties = chart.gethPieProperties();
pieProperties.setSepSymbol(",");
```

Please refer to the online API documentation for more information (quadbase.util.IPiePropertySet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IPiePropertySet.html]).

3.9.6.3.2.4. Pie Border Color Customizable

EspressReport allows the user to specify the color for the pie border. This is accomplished by using the `setBorderColor` method in `IPiePropertySet`.

```
IPiePropertySet pieProperties = chart.gethPieProperties();
pieProperties.setBorderColor(Color.red);
```

Please refer to the online API documentation for more information (quadbase.util.IPiePropertySet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IPiePropertySet.html]).

3.9.6.3.3. Line Charts

The following features describe the various changes that can be made to line charts using the API. These include modifying the content as well as look of the chart.

3.9.6.3.3.1. Line Area

EspressReport allows you to create line areas between a horizontal line and the data line to denote the change. For example, you could have a horizontal line at 25 and a data line. All areas enclosed by the data line and horizontal line and above the horizontal line can be one color and all areas enclosed by the horizontal line and data line and below the horizontal line can be another color. Please note that this feature is only available for 2D Line charts with no series. Also note that you need to set the color above and below the horizontal line in order to use this feature.

To use this feature in the API, you must get a handle to `ILinePropertySet` and use the `setAreaVisible` and `setAreaColors` methods.

The following code, which can be run as an applet or application, shows how to use line area:

```
ILinePropertySet lineProperties = chart.gethLineProperties();
lineProperties.setAreaVisible(true);
lineProperties.setAreaColors(Color.green, Color.yellow);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/LineArea.zip]

Exported Image [https://data.quadbase.com/Docs71/help/manual/code/export/LineArea.png]

Please refer to the online API documentation for more information (quadbase.util.ILinePropertySet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/ILinePropertySet.html]).

3.9.6.3.4. Scatter Charts

The following features describe the various changes that can be made to scatter charts using the API. These include modifying the content as well as look of the chart.

3.9.6.3.4.1. Show Series in Top Label

EspressReport allows series to be shown in the top labels for scatter charts. This can be done by using the method `showSeriesInTopLabel` in `IDataPointSet`.

```
IDataPointSet dataPoints=chart.gethDataPoints();
dataPoints.showSeriesInTopLabel(true);
```

Please refer to the online API documentation for more information (quadbase.util.IDataPointSet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IDataPointSet.html]).

3.9.6.3.4.2. Drawing Order

EspressReport allows the connecting lines for a Scatter chart to be drawn in the order of the dataset. For example, if the data contains the following points (0, 2), (3, 4), (1, 2), (2, 5), the default presentation for the connecting lines would generate a line from (0, 2) to (1, 2) to (2, 5) and finally (3, 4). However, you can generate the connecting lines in the order of the data.

To utilize this feature using the API, get a handle to `IDataPointSet` and use the `setConnectLinesInOriginalOrder` method.

```
IDataPointSet dataPoints = chart.gethDataPoints();
dataPoints.setConnectLinesInOriginalOrder(true);
```

Please refer to the online API documentation for more information (quadbase.util.IDataPointSet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IDataPointSet.html]).

3.9.6.3.4.3. Scatter Chart Cube Width

EspressReport allows the cube width of 3D Scatter charts to be changed to any size. To modify the cube width, first get a handle to `IDataPointSet` and use the `setScatterCubeWidth` method.

```
IDataPointSet dataPoints = chart.gethDataPoints();
dataPoints.setScatterCubeWidth(15);
```

Please refer to the online API documentation for more information (quadbase.util.IDataPointSet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IDataPointSet.html]).

3.9.6.3.5. Overlay Charts

The following features describe the various changes that can be made to overlay charts using the API. These include modifying the content as well as look of the chart.

3.9.6.3.5.1. Multiple Axes Titles

EspressReport allows the setting of a different title for each axis in an overlay chart. This can be done by getting a handle to each individual axis and using the `gethTitle().setValue` method. You get the handle to the individual axis by specifying the layer.

```
IAxis axis0 = chart.gethYAxis();
axis0.gethTitle().setValue("XYZ");

IAxis axis1 = chart.gethAxis(1); // Get axis of Layer 1
axis1.gethTitle().setValue("ABC");

IAxis axis2 = chart.gethAxis(2); // Get axis of Layer 2
axis2.gethTitle().setValue("DEF");
```

Please note that before you set the titles, you need to draw the chart in the background.

Please refer to the online API documentation for more information (quadbase.util.IAxis [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IAxis.html]).

3.9.6.3.6. Dial Charts

The following features describe the various changes that can be made to dial charts using the API. These include modifying the content as well as look of the chart.

3.9.6.3.6.1. Control Area Scale Labels

EspressReport allows the starting and ending scale of a control area to be shown as labels. This can be done by obtaining a handle to a control area and use the `setShowLabel method`.

```
ControlRange cr1 = chart.gethControlRanges().elementAt(0);
cr1.setShowLabel(true);
```

Please refer to the online API documentation for more information (quadbase.util.ControlRange [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/ControlRange.html]).

3.9.6.3.7. HLCO Charts

The following features describe the various changes that can be made to HLCO charts using the API. These include modifying the content as well as look of the chart.

3.9.6.3.7.1. Changing Candle Stick Color

EspressReport allows you to change the candlestick color for HLCO charts using the API only. To do so, you will need to get a handle to `IDataPointSet` and use `setCandleStickColors` method.

```
IDataPointSet dataPoints = chart.gethDataPoints();
// Up color is green, Down color is red
dataPoints.setCandleStickColors(Color.green, Color.red);
```

Please refer to the online API documentation for more information (quadbase.util.IDataPointSet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IDataPointSet.html]).

3.9.6.3.7.2. Changing Candle Stick Wicker Width

EspressReport allows you to change the candlestick wicker width (the upper and lower extensions of candlesticks) for HLCO charts using the API only. To do so, you will need to get a handle to `IDataPointSet` and use the method `setCandleStickWidth`. The number passed is the ratio of the candle wicker width to the candle width.

```
IDataPointSet dataPoints = chart.gethDataPoints();
// Set width to 0.5
dataPoints.setCandleStickWidth((float)0.5);
```

Please refer to the online API documentation for more information (quadbase.util.IDataPointSet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IDataPointSet.html]).

3.9.6.3.8. Surface Charts

The following features describe the various changes that can be made to surface charts using the API. These include modifying the content as well as look of the chart.

3.9.6.3.8.1. Heat Map

EspressReport allows users to draw a surface chart like a contour map. Basically, the surface chart can be drawn in sections, with different colors according to the threshold values specified.

The following code, which can be run as an applet or application, shows how to create a surface chart with a heat map:

```
double [] heatMapValues = {3, 6};
Color [] heatMapColors = { Color.green, Color.yellow, Color.red};
ColorSpectrum heatMapColorSpectrum = new ColorSpectrum(heatMapColors,
 heatMapValues);
I3DPropertySet set = chart.geth3DProperties();
set.setColorSpectrum(heatMapColorSpectrum);
```

The code above creates a 3D surface chart with 3 color sections, green, yellow and red. The threshold values determining the colors are 3 and 6.

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/HeatMap.zip]

Exported Image [https://data.quadbase.com/Docs71/help/manual/code/export/HeatMap.png]

Please refer to the online API documentation for more information (quadbase.util.ColorSpectrum [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/ColorSpectrum.html] and quadbase.util.I3DPropertySet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/I3DPropertySet.html]).

3.9.6.4. Performance

The features, shown below, deal with improving the performance of the chart generation and export. Each feature below shows additional functionality available to the chart, to decrease memory resources and time needed to generate the chart.

3.9.6.4.1. BufferedImage or Frame

EspressReport allows the choice of using either `java.awt.image.BufferedImage` or `java.awt.Frame` during export to improve performance. By default, EspressReport uses `java.awt.Frame` to create the chart object. Using `java.awt.Frame` gives better performance as the number of data points increases while using `java.awt.image.BufferedImage` yields better performance on larger chart dimensions. This is done by using the `setBufferedImageUsed` method in `QbChart`.

```
chart.setBufferedImageUsed(true);
```

Please refer to the online API documentation for more information (quadbase.ChartAPI.QbChart [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/ChartAPI/QbChart.html]).

This functionality is only available for stand-alone charts.

3.9.6.4.2. Chart Generation Order

EspressReport allows you to choose the order in which the chart is generated and even add elements in the generation of the chart using the API only. For example, you can now draw a background, and then a circle and have the chart generated in the center of the circle to create a chart. You can do this by creating a class that implements the `IChartGraphics` interface and then assigning the class to the `setChartGraphics` method in `QbChart`. In essence, the `IChartGraphics` interface allows you to add or modify any graphics information before or after drawing the chart.

The following code, which can be run as an applet or application, shows how to generate charts and graphics in a specific order:

```
// Open the template
QbChart chart = new QbChart(parent,
  // container
                            "../templates/ChartGeneration.cht"); // template

chart.setChartGraphics(new chartGenerationGraphics());;

...

class chartGenerationGraphics implements IChartGraphics {

      public void initializeGraphics(Graphics g, int w, int h) {
            g.setColor(Color.red);
            g.fillOval(50, 50, 400, 400);

      }

      public void finalizeGraphics(Graphics g, int w, int h) {
            g.setColor(Color.white);
            g.fillOval(125, 225, 50, 50);
            g.setColor(Color.orange);
            g.drawString("HELLO WORLD", 150, 250);
```

```
        }

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ChartGeneration.zip]

Exported Image [https://data.quadbase.com/Docs71/help/manual/code/export/ChartGeneration.png]

Please refer to the online API documentation for more information (quadbase.util.IChartGraphics [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IChartGraphics.html]).

This functionality is only available for stand-alone charts.

3.9.6.5. Viewer

The features, shown below, deal with the Viewer when viewing the chart in either a java application or java applet(More in: Section 1.2.7 - Run Applets in WebStart with JNLP file). Each feature below shows additional functionality available to Chart Viewer.

3.9.6.5.1. Call Back Mechanism

EspressReport has a call back mechanism for higher levels to handle the event. When a data object in the chart is selected by the viewer, an action event is generated. The event argument contains an instance of `PickData`, which provides information of the series, category, value, etc of the data point selected.

The following code, which can be run as an applet or application, shows how to capture an event:

```
static TextField textField;

// Open the template
QbChart chart = new QbChart(parent,                                 //
 container
                            "../templates/CallBack.cht"); // template

chart.setActionListener(new callBackActionListener());;

...

class callBackActionListener implements ActionListener {

      public void actionPerformed(ActionEvent e) {
            Object arg = ((QbChart) e.getSource()).getArgument();

            String click;

            switch (e.getModifiers()) {

            case QbChart.LEFT_SINGLECLICK:
                  click = "Left single click";
                  break;

            case QbChart.LEFT_DOUBLECLICK:
                  click = "Left double click";
                  break;

            case QbChart.RIGHT_SINGLECLICK:
                  click = "Right single click";
                  break;

            case QbChart.RIGHT_DOUBLECLICK:
                  click = "Right double click";
                  break;
            default: // shall not happen
                  click = "Error !";

            }

            if (arg instanceof PickData)
                  textField.setText(((PickData) arg).toString() + " " +
 click);

            else
                  textField.setText((String) arg + " " + click);

      }

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/CallBack.zip]

Exported Image [https://data.quadbase.com/Docs71/help/manual/code/export/CallBack.png]

This functionality is only available for stand-alone charts.

3.9.6.5.2. Disable/Enable Tool Tips Text

EspressReport allows the tool tips text for the navigation panel to be enabled and disabled. This can be done by using the method `setToolTipEnabled` in `I3DControlPanel`. Note that this option is for 3D charts only.

```
I3DControlPanel controlPanel = chart.geth3DControlPanel();
controlPanel.setToolTipEnabled(true);
```

Please refer to the online API documentation for more information (quadbase.util.I3DControlPanel [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/I3DControlPanel.html]).

3.9.6.5.3. Canvas Area

EspressReport allows the viewpanel containing the canvas to be selected so that more event properties (i.e., user defined event properties) can be added. You can do this by getting a handle to `ICanvas` and using the `getCanvasArea()` method to return the component.

```
ICanvas chartCanvas = chart.gethCanvas();
Component chartCanvasComponent = chartCanvas.getCanvasArea();
```

Please refer to the online API documentation for more information (quadbase.util.ICanvas [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/ICanvas.html]).

3.9.7. Changing Chart Viewer Options

At times, you may want to configure what users can or cannot do when they are viewing the chart using the Chart Viewer.

When the user is using Chart Viewer to view the chart, right clicking on the chart causes a menu to pop up. Using this menu, the user can select chart options such as changing chart type, changing chart dimension etc The user can also choose to export the chart and the type of the static image. While the default pop-up menu lists all available choices, API methods exist that control what options are available in the pop-up menu of Chart Viewer.

These API calls are available in `IPopupMenu`:

```
IPopupMenu popupMenu = chart.gethPopupMenu();
popupMenu.setDimMenuEnabled(boolean b);
popupMenu.setPopupMenuEnabled(boolean b);
popupMenu.setTypeMenuEnabled(boolean b);
```

Please refer to the online API documentation for more information (quadbase.util.IPopupMenu [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IPopupMenu.html]).

3.9.8. Javadoc

Javadoc for the entire API is provided along with EspressReport. The API covers both the Chart and the Charting API. It is located at Quadbase website [https://data.quadbase.com/Docs71/er/help/apidocs/index.html].

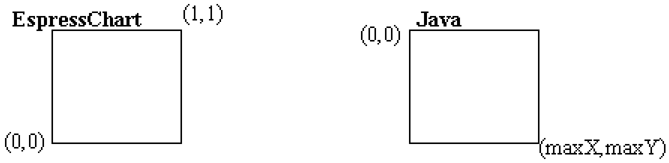
3.9.9. Swing Version

1.1 JFC/Swing versions of the EspressReport charting API is also available. For more details, please refer to quadbase.ChartAPI.swing package [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/ChartAPI/swing/package-summary.html].

3.9.10. Summary

The EspressReport API provides an easy-to-use, yet powerful charting library for business applications. Combined with Chart Designer, programming is as simple as adding one line of code to your applet(More in: Section 1.2.7 - Run Applets in WebStart with JNLP file) or application. All of the attributes of a chart may be set in a template file, which can be created with the EspressReport Designer.

3.A. Customizing Chart Layout



Component Layout Differences between EspressReport and Java

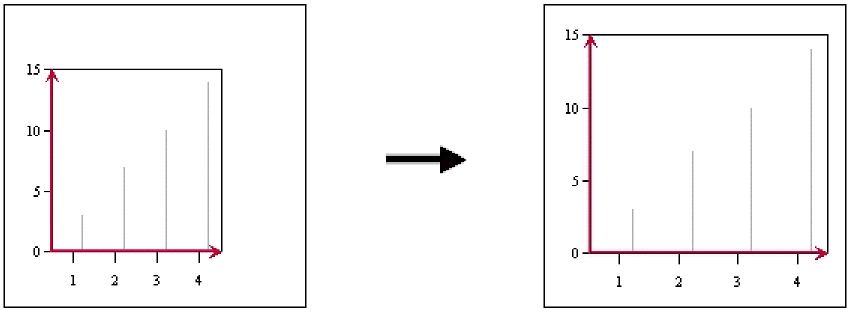
EspressReport describes its component layout in relative coordinates. On the other hand, Java uses layout format in terms of pixels. Due to backward-compatibility issues, the EspressReport layout format will not be changed. The issues pertaining to relative & absolute position is less relevant in Chart Designer since there is a graphical user's interface. However, it is helpful to know how the layout format is in Chart API: the origin (0, 0) is at the lower left-hand corner of the canvas and the maximum (1, 1) is at the upper right-hand corner.

In the Java layout format, the origin is at the upper left-hand corner and the maximum pixel (maximumX, maximumY) is at the lower right-hand corner.

In the `IAxis` Class, the methods `setMaxScale`, `setMinScale`, and `setScaleStep` refer to the range and interval of the axis. Therefore, these methods would not affect the apparent length of the axis, rather, they specify the range of the axis.

The following sections discuss some examples of customization techniques relating to layout of chart components, re-sizing and data point labeling.

3.A.1. Changing the Chart Plot Area



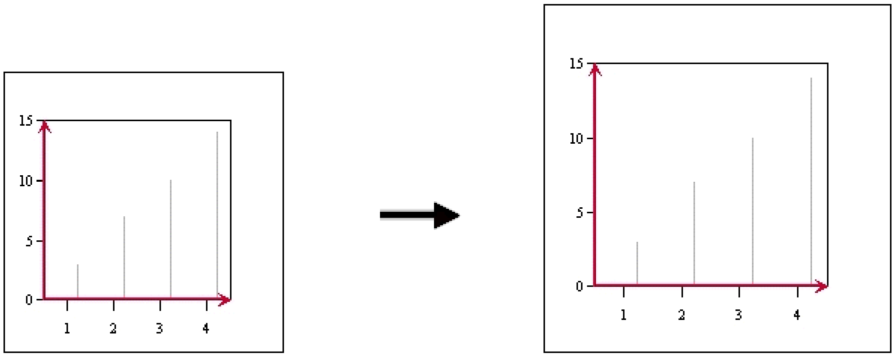
Enlarging the Chart Plot Area

The default relative size of the chart plot area is 0.6 in the x dimension and 0.6 in the y dimension. If you want to adjust these ratios using the API, you can use the `setRelativeHeight()` and `setRelativeWidth()` methods.

```
// chart plot’s default relative size is 0.6, 0.6
IPlot chartPlot = chart.gethChartPlot();
chartPlot.setRelativeHeight( (float) 0.75);
chartPlot.setRelativeWidth( (float) 0.8);
```

To adjust the chart plot in the Chart Designer, simply click and hold down the right mouse button when the mouse cursor is on the chart. Drag the mouse to the right to increase the chart plot area and to the left to decrease the chart plot area. Depending on the direction that you move the mouse (either vertically, horizontally, or diagonally), the chart plot changes in one or in both dimensions.

3.A.2. Changing the Canvas Area



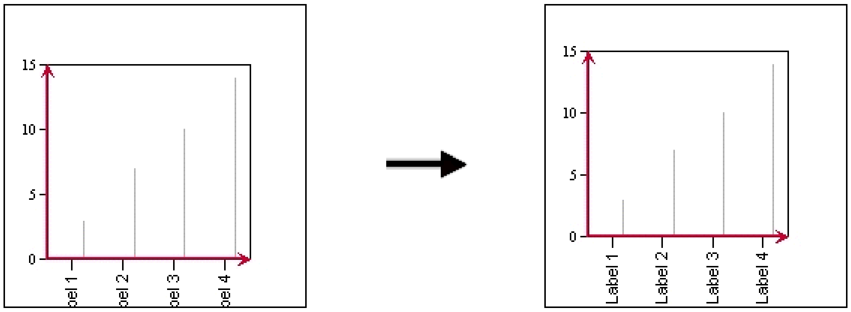
Enlarging the Canvas Area

This example shows you how to adjust the size of the canvas while keeping the Chart Plot size constant in relation to the canvas.

```
// default canvas size is 600 pixels by 500 pixels
int width = 500; // in pixels
int height = 550; // in pixels
chart.gethCanvas().setSize( new Dimension(width, height));
```

To change the canvas area in the Chart Designer, go to Format → Canvas and change the dimensions there. Note that the canvas is always equal to or greater than the Chart Designer window size. If you want a smaller Canvas area, you must reduce the dimensions of the Chart Designer window first.

3.A.3. Fit Charts Elements

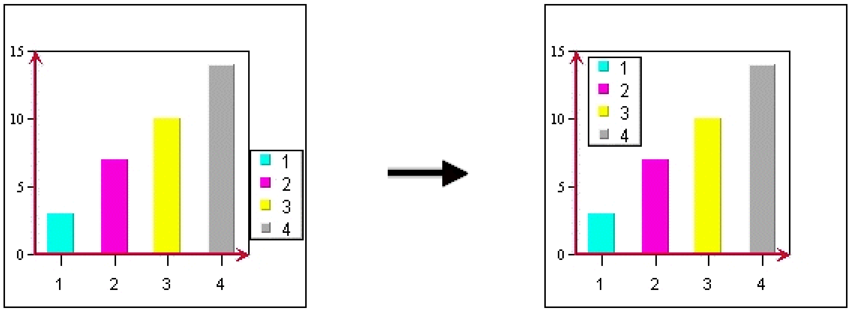


Fitting Chart Elements onto Canvas

If you notice that there are chart elements (e.g., ticker labels for the Category Axis) that are “chopped off” by the canvas, you can use the `setFitOnCanvas()` method to do the necessary adjustments.

```
// try to reposition chart location or reduce chart size until
// EspressReport is able to fit a whole chart
// on canvas. Default value is FALSE
chart.gethCanvas().setFitOnCanvas(true);
```

3.A.4. Changing Position of Legend Box



Changing the Position of the Legend Box

The position of the legend box can be adjusted if the default location is not what you want. Please note that the reference point is the center of the legend box.

```
// Legend Box can be moved with a specified new relative position
ILegend hLegend = chart.gethLegend();
float x = 0.2;
float y = 0.7;
hLegend.setPosition( new Position( x, y) );
```

Legend box size is dependent on the size of the text font & text strings. Therefore, you cannot set the legend box size directly, but you can get the size of the legend box. Here are some sample codes:

```
ILegend hLegend = chart.gethLegend()
float relWidth = hLegend.getRelativeWidth();
float relHeight = hLegend.getRelativeHeight();
```

To change the position of the legend in the Chart Designer, left click on the legend and drag it to the desired position on the canvas.

3.A.5. Attaching Labels to Datapoints

Depending on your needs, it is sometimes necessary to have labels attached to data points. EspressReport provides a few features to fulfill these requirements. For example, you can attach annotation text to trend lines. When the trend line moves with the data, the annotation text will move in tandem with the trend line. Top labels can be set visible and appear at the top of every data point. However, in certain situations, you may want to have labels attached to just a few selected data points. How would you go about doing this?

You can use existing features to achieve this as follows: First, draw a constant horizontal line with value of the desired data point. Next, attach an annotation text to the line. Hide the legend for the line. Finally, make the horizontal line invisible by simply choosing a dashed line style and making both the fill and empty pixel lengths to be 255.

The following code fragment demonstrates this technique and the following tables show the data referred to in the code. The objective is to attach labels to two data points at the far right of the chart (the maximum of both series) and to attach one label to a data point at the left side of the chart (the minimum as shown below).

Please note that the positions of the labels are updated automatically when one of the last data points changed from $700 to $500.

Day	Value1	Value2
1	100	100
2	150	300
3	200	500
4	250	700

Original Data

Day	Value1	Value2
1	100	100
2	150	300
3	200	450
4	250	500

New Data

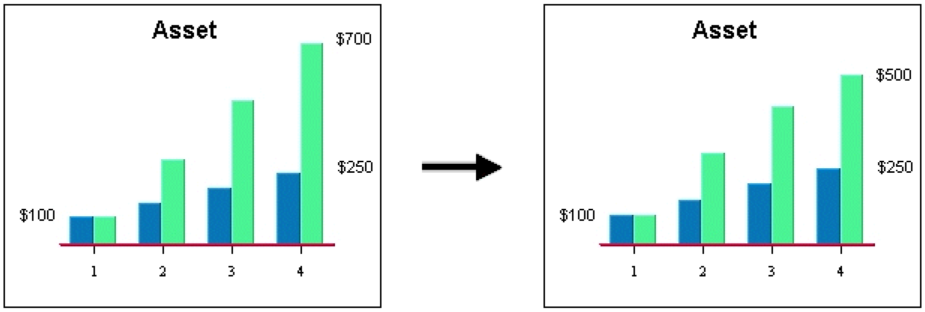


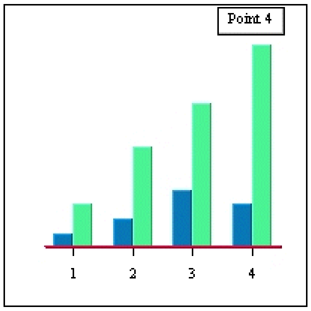
Chart with Original Data Changing to Chart with New Data

```
// example finds one of the value points in the last row of a database
 query.
// it’s not necessary to label the last value in particular, you can specify
// whichever values to label in your program, i.e. maximum, minimum,
// or average....

// get handle on the first set of data points
IRow rowInit = chart.gethInputData().getRow(0);
// get initial value.  if the first series is from column 3, index is 3
int valueInit = Integer.parseInt(rowInit.getObject(3).toString());

// get handle on the last set of data points
int lastRownumber = chart.gethInputData().getRowCount() - 1;
IRow lastRow = chart.gethInputData().getRow(lastRownumber);
IRow lastButOneRow = chart.gethInputData().getRow(lastRownumber-1);

// the second series last value is in the last row
int value1 = Integer.parseInt(lastRow.getObject(3).toString());
// the first series last value is in the last but one row
int value2 = Integer.parseInt(lastButOneRow.getObject(3).toString());
```

```
String labelInit = "$" + valueInit;
String label1 = "$" + value1;
String label2 = "$" + value2;

// add the line for the initial value
IDataLineSet hDataLines = chart.gethDataLines();
IHorzVertLine hvInit = hDataLines.newHorzVertLine(
IHorzVertLine.HORIZONTAL_LINE, labelInit);

hvInit.setLineValue(valueInit);   // horizontal line at value0
hvInit.setLineStyle( ((255*256) + 255) *256); // set line invisible
hDataLines.add(hvInit);

// add the line for the ending value of the first series
IHorzVertLine hv1 = hDataLines.newHorzVertLine(
IHorzVertLine.HORIZONTAL_LINE, label1);

hv1.setLineValue(value1);   // horizontal line at value1
hv1.setLineStyle( ((255*256) + 255) *256); // set line invisible
hDataLines.add(hv1);

// add the line for the ending value of the second series
IHorzVertLine hv2 = hDataLines.newHorzVertLine(
IHorzVertLine.HORIZONTAL_LINE, label2);

hv2.setLineValue(value2);   // horizontal line at value1
hv2.setLineStyle( ((255*256) + 255) *256); // set line invisible
hDataLines.add(hv2);

// Add Annotations to the lines
IAnnotationSet hAnnotation = chart.gethAnnotations();
IAnnotation annoInit = hAnnotation.newAnnotation(labelInit, hvInit);
IAnnotation anno1 = hAnnotation.newAnnotation(label1, hv1);
IAnnotation anno2 = hAnnotation.newAnnotation(label2, hv2);

// Add annoInit to the left of the chart plot area. ($100)
hvInit.addAnnotation(annoInit);
annoInit.setRelativePosition(new Point_2D( -
(float)chart.gethChartPlot().getRelativeWidth(), 0f));

// Add annotation1 to the line, so it will appear on the chart
hv1.addAnnotation(anno1);

// Add annotation2 to the chart, so it will appear on the chart
hv2.addAnnotation(anno2);

// set legend box invisible
chart.gethLegend().setVisible(false);
```

If you want to show individual category top labels selectively, you can add a vertical trend line with the corresponding category value. Within EspressReport's graphical mechanism, category data point positions are always referenced as numbers even though they are usually not numbers. The first data point is always referenced as 0.5 and each of the following points would have 1.0 added to it. Thus, a set of points would always be referenced as 0.5, 1.5, 2.5, 3.5...etc. The only exceptions are in Scatter and Bubble Charts.

After inserting the line in the right position, we can make the line invisible by setting the line style. Annotation is then added.



Adding "Point 4" to the chart

```
// adding a label above the fourth point, “Point 4”
ITrendLine tr1 = hDataLines.newTrendLine(ITrendLine.VERTICAL_LINE, 1, "Point
 4");
tr1.setTitleVisibleInLegend(false);   // don't show title on legend
tr1.setLineValue(3.5);   // vertical line on the 4th data point
tr1.setLineStyle( ((255*256) + 255) *256); // set line invisible
hDataLines.add(tr1);

// Add Annotations to the lines
IAnnotationSet hAnnotation = chart.gethAnnotations();
IAnnotation annoTr1 = hAnnotation.newAnnotation(“Point 4”, tr1);

// Add annoTr1 to the line, so it will appear on the chart
tr1.addAnnotation(annoTr1);
```

As the scale of the chart or data point value changes, the label will always be displayed at the top of the data point.

3.B. Getting the Chart Data

The two appendices in this Chapter (Appendix 3.B - Getting the Chart Data and Appendix 3.C - Creating the Chart) contain information to help you build a chart from scratch (without using the designer). Keep in mind that this method is typically not recommended as it will make the chart more difficult to deploy and maintain. However, under certain circumstances, it may be necessary to construct charts in this manner.

The following is an example of a `QbChart` constructor. Like reports, there are numerous chart constructors available. The typical `QbChart` constructor requires at least four parameters. To create a new chart, you must specify the chart dimensions, chart type, the input data source information, and a mapping of the data columns to the respective columns of the chart.

```
QbChart(java.applet.Applet applet, int dimension, int chartType,
 IDatabaseInfo dbinfo, IColumnMap cmap, java.lang.String template)
```

The template is not required (can be null) and the applet is only required if the chart is to be displayed in an applet, but the other four parameters must always contain meaningful information. This appendix takes an in depth look at the data source parameter and the methods used to connect to different data sources. Appendix 3.C - Creating the Chart discusses the dimensions, the chart type, the column mapping, and the actual creation of the chart. Appendix 3.C - Creating the Chart also contains working examples for you to try.

The data for a chart can be obtained from a report (if the chart is inserted into a report) or from an independent data source. The following sections deal with connecting to an independent data source in order to obtain data to generate a chart.

The data used to create a chart may be fetched from one of several different types of sources. These sources include:

- Fetch the data from a local or remote database using a JDBC driver;

- Read the data from a plain text file, which contains database records in plain text (ASCII) format. Files of this format can be generated by most database programs;
- Read the data from a spreadsheet model;
- Pass the dataset as an array in memory;
- Pass your own data through API;
- Fetch the data from an EJB data source;
- Merge from multiple data sources;

In the remainder of this section, we discuss the above methods of data extraction in greater detail.

3.B.1. Data from a Database

One of the powerful features of Chart API is its ability to fetch data from a database directly through JDBC. With this approach, all you need to do is to specify the information necessary to connect to the database and the precise form of the SQL statement. Thus your program can connect to virtually any database provided that a JDBC driver is available.

The database and query information must be stored in a DBInfo object prior to constructing the chart. Use the following code to instantiate the DBInfo object.

```
DBInfo dbinfo = new DBInfo(

                  "jdbc:odbc:ODBCDatabase",             // URL
                  "sun.jdbc.odbc.JdbcOdbcDriver",       // JDBC driver
                  "myName",                             // username
                  "myPassword",                         // password
                  "select * from sales");               // SQL
```

Since DBInfo implements the interface IDatabaseInfo, you can use the DBInfo object in the following QbChart constructor:

```
public QbChart(Applet parent, int dimension, int chartType, IDatabaseInfo
 dbinfo, ColInfo colMap, String templateFile);
```

In some cases, you may not wish to pass the entire database information (such as userid, password, location, driver, and the query) to EspressReport. You may want to make the connection yourself and just provide the result set of the query directly to the API. This can be done by creating a QueryResultSet object from the ResultSet object you have generated and passing that QueryResultSet object as the data source, instead of a DBInfo object.

```
// Create a QueryResult object from a java.sql.ResultSet object resultSet
QueryResultSet queryResultSet = new QueryResultSet(resultSet);

// use queryResultSet in place of IResultSet data in the following
 constructor
QbChart(java.applet.Applet applet, int dimension, int chartType, IResultSet
 data, IColumnMap cmap, java.lang.String template)
```

In the above example, an instance of class QueryResultSet is created from the ResultSet object passed to the API and this instance is passed to the QbChart constructor to create the chart object. Please note that in this scenario, the chart is not refreshable. To update the chart, you will need to make the connection to the database again and pass the result set object to the chart and refresh it yourself.

3.B.1.1. JNDI

EspressReport also allows data to be obtained from a JNDI source. JNDI data sources are treated like database data sources and support the same functionalities. Using a JNDI data source makes it easier to migrate charts between different environments. If the data sources in both environments are setup with the same lookup name, charts can be migrated without any changes.

To connect to a JNDI data source, you must have a data source deployed in your application server. You must also provide the **`INITIAL_CONTEXT_FACTORY`** and **`PROVIDER_URL`** to successfully make the connection. Please note that when connecting to a JNDI data source deployed in Tomcat, the **`INITIAL_CONTEXT_FACTORY`** and **`PROVIDER_URL`** need not be provided (although EspressManager must be running as a servlet under the Tomcat environment).

```
// create a DBInfo object with JNDI lookup name and query
String JNDIName = "java:comp/env/jdbc/TestDB";
String query = "select * from testdata";

// The environment hashtable is empty for tomcat because EspressManager is
// running inside Tomcat context. If other application server is used,
// need to set INITIAL_CONTEXT_FACTORY and PROVIDER_URL.
Hashtable env = new Hashtable();
DBInfo dbInfo = new DBInfo(JNDIName, query, env);
```

In the above example, an instance of class `DBInfo` provides the information that the program needs to connect to a JNDI data source and retrieve data. To construct the chart, use the following constructor containing `IDatabaseInfo`:

```
public QbChart(Applet parent, int dimension, int chartType, IDatabaseInfo
 dbinfo, ColInfo colMap, String templateFile);
```

3.B.1.2. JNDI example using Tomcat and EspressManager running from servlet

The following JNDI example has been run successfully using Tomcat and existing hsqldb woodview datasource, that is located under (EspressReport>\help\examples\DataSources\database\woodview with the following steps.

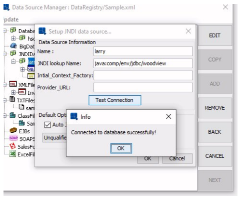
First, install EspressReport (e.g. ER70) at the webroot of Tomcat (e.g. C:\apache-tomcat-8.5.83) and configure to run EspressManager from servlet following paragraph Section 1.2.4.2 - Starting EspressManager as a Servlet Second, modify the context.xml file in Tomcat conf directory and include the following codes at the end of context.xml before the last context tag.

```
<Resource auth="Container"
driverClassName="org.hsqldb.jdbcDriver"
maxActive="100" maxIdle="30" maxWait="10000"
name="jdbc/woodview" type="javax.sql.DataSource"
url="jdbc:hsqldb:C:\apache-tomcat-8.5.83a\webapps\ROOT\ER70\help\examples
\DataSources\database\woodview"
username="sa" password="" validationQuery="SELECT 1 from
 INFORMATION_SCHEMA.SYSTEM_USERS"/>
```

Third, modify the web.xml that is located inside the EspressReport\WEB-INF directory and append the following code before the last web-app tag.

```
<resource-ref>
<res-ref-name>jdbc/woodview</res-ref-name>
<res-type>javax.sql.DataSource</res-type>
<res-auth>Container</res-auth>
<res-sharing-scope>Shareable</res-sharing-scope>
</resource-ref>
```

Then, restart Tomcat and run EspressReport\reportdesigner.bat to start the reportdesigner while espressmanager start from servlet In reportdesigner, File/New and open DataRegistry JNDI node. Click on Add, enter any Name in the Name field and java:comp/env/jdbc/woodview should connect to hsqldb...woodview database as shown in the image below.


Data Source Manager : DataRegistry/Sample.xml

Setup JNDI data source...

Data Source Information

Name : larry

JNDI lookup Name: java:comp/env/jdbc/woodview

Intial_Context_Factory:

Provider_URL:

Info

Connected to database successfully!

Woodview database connection

3.B.2. Data from a Data file (TXT/DAT/XML)

Data for generating a chart can also be imported from a data file, which can be a text file as well as an XML formatted file. A chart data file (with a .dat extension) is a plain text file where each line represents one record, except the first two lines, which contain the data types and field names, respectively. Here is an example of a data file, which contains four records and where each record has three fields. The first line specifies the data type of each field, and the second line specifies the field name.

```
string, date, decimal
Name, Day, Volume
"John", "1997-10-3", 32.3
"John", "1997-4-3",  20.2
"Mary", "1997-9-3", 10.2
"Mary", "1997-10-04", 18.6
```

The use of the comma character is optional, but most database programs can output a file in this format (except for the first two lines) when exporting records from a table in text format. As a result, even if you do not have a JDBC driver for your database, you can still quickly produce charts. You can simply export the data from your database in text format and then your program, using the API, can read it.

For XML format, the specification is as follows:

```
<EspressData>

      <DataType>string</DataType>
      <DataType>date</DataType>
      <DataType>decimal</DataType>
      <FieldName>Name</FieldName>
      <FieldName>Day</FieldName>
      <FieldName>Volume</FieldName>
      <Row>
            <Data>John</Data>
            <Data>1997-10-3</Data>
            <Data>32.3</Data> </Row>
      <Row>

        <Data>John</Data>
        <Data>1997-4-3</Data>
        <Data>20.2</Data>
    </Row>
    <Row>
        <Data>Mary</Data>
        <Data>1997-9-3</Data>
        <Data>10.2</Data>
    </Row>
    <Row>
        <Data>Mary</Data>
        <Data>1997-10-4</Data>
        <Data>18.6</Data>
    </Row>
</EspressData>
```

For more details about XML Data, please refer to the Section 1.3.3 - Data from XML and XBRL Files.

Specifying the text file to use is very straight forward. Use the following constructor and replace the variable filename with the file path (relative or full path) of the data file. If you are connecting to the server, relative paths should be in respect to the EspressReport root directory. Otherwise, the path will be relative to the current working directory. Also replace the **`fileType`** parameter with one of the following options: **`QbChart.DATAFILE`**, **`QbChart.QUERYFILE`**, or **`QbChart.XMLFILE`**

```
public QbChart(Applet parent, int dimension, int chartType, int fileType,
 String filename, boolean doTransposeData, ColInfo colMap, String
 templateFile);
```

The same constructor can also be used for passing in XML data generated by a servlet. You can specify the fileType as **`QbChart.XMLFILE`** and the filename as the URL to your servlet (e.g. **`http://localhost:8080/servlet/XMLDataGenerator`**).

3.B.3. Data from a XML Data Source

In addition to the above, EspressReport allows you to retrieve data and query XML files. XML data can be in virtually any format, but you need to specify a DTD file or an XML schema along with the XML data. The following code demonstrates how to set up an XML query:

```
// Set up the XML Data Source
String xmlfilename = "Inventory.xml";
String xmlcondition = "/Inventory/Category/Product/ProductID < 45";

XMLFieldInfo[] fields = new XMLFieldInfo[5];
fields[0] = new XMLFieldInfo(new String[]
 {"Inventory", "Category", "Product"}, "ProductID");
fields[0].setAttributeDataType(DTDDataType.INT);

fields[1] = new XMLFieldInfo(new String[]
 {"Inventory", "Category", "Product", "ProductName"});

fields[2] = new XMLFieldInfo(new String[]
 {"Inventory", "Category", "Product", "UnitPrice"});
fields[2].setElementDataType(DTDDataType.DOUBLE);

fields[3] = new XMLFieldInfo(new String[]
 {"Inventory", "Category", "Product", "UnitsInStock"});
fields[3].setElementDataType(DTDDataType.INT);

fields[4] = new XMLFieldInfo(new String[]
 {"Inventory", "Category", "Product", "ShipDate"});
fields[4].setElementDataType(DTDDataType.DATE);
fields[4].setDateFormat(XMLDataTypeUtil.YYYY_MM_DD);

XMLFileQueryInfo xmlInfo = new XMLFileQueryInfo(xmlfilename, fields,
 xmlcondition, fields);
```

The `XMLFieldInfo` instance is created using one of two constructors. The first constructor, used to select xml fields, contains one parameter. For this constructor, you need to pass in a String array that specifies each xml tag in the hierarchy leading to the target field. In the above example, fields[1-4] are created using the first constructor. The second constructor, used to select xml attributes, contains two parameters. In addition to the String array parameter, the second constructor also requires another string for the attribute name. In the above example, field[0] is created using this constructor because “ProductID” is an attribute of “Product”.

You may have also noticed that for any non-String field, you must explicitly set the data type. Once you have created the `XMLFileQueryInfo` instance, you can use the following constructor to create the `QbChart`.

```
public QbChart(Applet applet, int dimension, int chartType, XMLFileQueryInfo
 xmlInfo, boolean doTranspose, int[] transposeColumns, IColumnMap colMap,
 String templateFile);
```

You can also pass in an XML stream instead of an XML file, when using XML data as a data source. To pass in an XML stream, you would pass in the byte array containing the XML data instead of the XML data file name.

In the above example, you can pass in a XML stream via a byte array (for example, a byte array called xmlByteArray) in the XMLFileQueryInfo [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/common/util/internal/XMLFileQueryInfo.html] constructor:

```
XMLFileQueryInfo xmlInfo = new XMLFileQueryInfo(xmlByteArray, fields,
 xmlCondition, fields);
```

3.B.4. Data Passed in an Array in Memory

The API allows input data to be passed directly in memory, as an array. This is made possible by the interface IResultSet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IResultSet.html] (defined in quadbase.util [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/package-summary.html] package). This interface is used to read data in the tabular form, and is quite similar to the `java.sql.ResultSet` interface used for JDBC result sets (but is much simpler). Users can provide their own implementations of `IResultSet` (which is discussed in the next section), or use one provided by EspressReport. The simplest implementation is provided by the class `DbData` (Other classes that provide an `IResultSet` implementation are QueryResultSet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/QueryResultSet.html] and StreamResultSet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/StreamResultSet.html]). If you can fit all the data you need for the chart in memory, you can simply pass the array as an argument in `DbData` with one line of code. There are three constructors for `DbData`:

- `DbData(java.lang.String s)` - Construct `DbData` by parsing the data value argument from an HTML page
- `DbData(java.lang.String[] fieldName, java.lang.Object[][] records)` - Construct a new `DbData` class
- `DbData(java.lang.String[] dataType, java.lang.String[] fieldName java.lang.String[][] records)` - Construct a new `DbData` class

We will use the following constructor in the example here.

```
public DbData(String dataType[], String fieldName[], String records[][]);
```

This is a similar construction to reading in data from a data file. Here, the first argument presents the data types (the first line in the data file) and the second argument presents the field names (the second line). The third argument, records[][], provides an array of records, records[i] being the ith record. The following shows how it works.

```
String dataType[] = {"varchar", "decimal"};
String fieldName[] = {"People", "Sales"};
String records[][] = {{"Peter", "93"}, {"Peter", "124"},
                      {"John", "110"}, {"John", "130"},
                      {"Mary", "103"}, {"Mary", "129"}};

DbData data = new DbData(dataType, fieldName, records);
```

To create the chart, use the following `QbChart` constructor:

```
public QbChart(Applet parent, int dimension, int chartType, IResultSet data,
 ColInfo colMap, String templateFile);
```

3.B.5. Data Passed in your Custom Implementation

For maximum flexibility, you can retrieve and prepare the dataset in any way you want and pass it to the charting engine. To pass in your class file as the data source, your class file must implement the IDataSource [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IDataSource.html] interface. Given below is an example of code that implements `IDataSource` and passes in a class file as the data source:

```
public class ChartCustomClassData extends Applet implements IDataSource {

      // Setting DbData for passing data as arguments
      String dataType[] = {"string", "String", "double"};
      String fieldName[] = {"Destination", "Time", "Price"};
      String records[][] = {{"Mayfair", "13:43", "3.50"},
            {"Bond Street", "13:37", "3.75"},
            {"RickmansWorth", "13:12", "5.25"},
            {"Picadilly", "13:24", "3.00"}};
      DbData data = new DbData(dataType, fieldName, records);

      public IResultSet getResultSet()
      {
            return data; }

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/CustomClassData.zip]

The example above creates data (`DbData` instance) and stores it in memory. When the `getResultSet()` method is called, it returns the `DbData` object which implements the IResultSet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IResultSet.html] interface. Keep in mind that it is not necessary to create your data in this manner. As long as you are able to return an object that implements `IResultSet`, you can obtain the data from any data source. Use the following constructor to create your chart:

```
public QbChart(Applet parent, int dimension, int chartType, int fileType,
 String filename, ColInfo colMap, String templateFile);
```

For custom class files, set the fileType to **`QbChart.CLASSFILE`** and the filename to the name of the classfile.

Please note that if you are passing in your own class file as the data source and you are using the EspressManager, the class file must be accessible from the CLASSPATH of the EspressManager.

You can also pass in a parameterized class file as the data source for the chart. The parameter is obtained at run-time from the user and the data is then fetched and used to generate the chart. Note that this will only work for a stand-alone chart configuration.

```
public class ParameterizedClassFile implements IParameterizedDataSource {
 public IQueryInParam[] getParameters()
 {
  SimpleQueryInParam[] params = new SimpleQueryInParam[1];
  params[0] = new SimpleQueryInParam("param2", "Enter the price:", false,
 null, null, Types.INTEGER, new Integer(2), null);
  return params;
 }

 public IResultSet getResultSet(IQueryInParam[] params) {
  double price = 3.5;
  if ((params != null) && (params.length >= 1))
  {
   Object obj = params[0].getValue();
   if ((obj != null) && (obj instanceof Integer)) price =
 ((Integer)obj).intValue();
  }
  String dataType[] = {"string", "String", "double"};
  String fieldName[] = {"Destination",  "Time", "Price"};
  String records[][] = {{"Mayfair",  "13:43", price+""},
    {"Bond Street", "13:37", price+""},
    {"Rickmansworth ", "13:12", price+""},
    {"Picadilly",  "13:24", price+""}};
  return new DbData(dataType, fieldName, records);
 }
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ParameterizedClassFile.zip]

When using a parameterized class file as the data source, the parameter dialog box is usually a text box with no choices available. However, you can specify what the selection choices can be (available via a drop-down box) by implementing the IQueryParamValuesProvider [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/IQueryParamValuesProvider.html] interface.

```
public class CustomParamClassFile implements IParameterizedDataSource {

 public IQueryInParam[] getParameters() {
  mySimpleQueryMultiValueInParam[] params = new
 mySimpleQueryMultiValueInParam[1];
  params[0] = new mySimpleQueryMultiValueInParam("region", "Select
 Region(s):", true, "Customers", "Region", Types.VARCHAR, "East", null);
  return params;
 }

 public IResultSet getResultSet(IQueryInParam[] params) {
  QueryResultSet data = null;
  ResultSet rs = null;

  String paramValue = "'East'";
  if ((params != null) && (params.length >= 1)) {
   Vector selectedValues = null;
   if (params[0] instanceof IQueryMultiValueInParam)
    selectedValues = ((IQueryMultiValueInParam)params[0]).getValues();

   for (int i = 0; i < selectedValues.size(); i++) {
    if ((selectedValues.get(i) != null) && (selectedValues.get(i) instanceof
 String)) {
     if (i == 0) paramValue = "'" + (String)selectedValues.get(i) + "'";
     else paramValue += ",'" + (String)selectedValues.get(i) + "'";
    }
   }
  }
  String myQuery = "select cu.region, c.categoryname, count(o.orderid),
 sum(od.quantity), sum(p.unitprice * od.quantity) from customers cu,
 categories c, products p, orders o, order_details od where cu.customerid =
 o.customerid and c.categoryid = p.categoryid and p.productid = od.productid
 and o.orderid = od.orderid and cu.region in (" + paramValue + ") group by
 cu.region, c.categoryname";

  try {
   Class.forName("org.hsqldb.jdbcDriver");
   String url = "jdbc:hsqldb:help/examples/DataSources/database/woodview";
   Connection conn = DriverManager.getConnection(url, "sa", "");
   Statement stmt = conn.createStatement();

   rs = stmt.executeQuery(myQuery);
   data = new QueryResultSet(rs);
   // conn.close();
  } catch (Exception ex) {
   ex.printStackTrace();
  }
  return data;
 }

 public class mySimpleQueryMultiValueInParam extends
 SimpleQueryMultiValueInParam implements IQueryParamValuesProvider {

  public String paramName, promptName, tableName, colName;
  boolean mapToColumn;
  int sqlType;
  Object defaultValue;
  Vector values;
  public mySimpleQueryMultiValueInParam(String paramName, String
 promptName, boolean mapToColumn, String tableName, String colName, int
 sqlType, Object defaultValue, Vector values) {
   super(paramName, promptName, mapToColumn, tableName, colName, sqlType,
 defaultValue, values);
  }

  public Vector getSelectionChoices() {
   System.out.println("getSelectionChoices called");
   try {
    Class.forName("org.hsqldb.jdbcDriver");

    String url = "jdbc:hsqldb:help/examples/DataSources/database/woodview";
    Connection conn = DriverManager.getConnection(url, "sa", "");
    Statement stmt = conn.createStatement();
    String query = "SELECT DISTINCT " + getColumnName() + " FROM " +
 getTableName();
    ResultSet rs = stmt.executeQuery(query);
    Vector v = new Vector();

    while (rs.next()) {
     switch (getSqlType()) {
     case Types.INTEGER:

      v.add(new Integer(rs.getInt(1)));
      break;

     case Types.VARCHAR:

      v.add(rs.getString(1));
      break;
     }
    }
    stmt.close();
    conn.close();
```

```
    return v;
   } catch (Exception ex) {
    ex.printStackTrace();
   }
   return null;
  }

 }

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/CustomParamClassFile.zip]

3.B.6. Data from a Spreadsheet Model

A chart can function as a view to a spreadsheet model in a Model-View-Controller (MVC) architecture. It automatically reads the spreadsheet data and plots itself. The chart registers itself as a listener to the spreadsheet model and updates itself when notified of any changes to the spreadsheet data. A spreadsheet (Java) object is provided by the user, which is an instance of a class that implements the ISpreadSheetModel [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/ISpreadSheetModel.html] interface. The event class `SpreadSheetModelEvent` is used by the model to notify its listeners of any changes to data. The following example shows how a spreadsheet model can be used for a chart. It uses the utility class SimpleSpreadSheet [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/SimpleSpreadSheet.html] (defined in the quadbase.util [https://data.quadbase.com/Docs71/er/help/apidocs/quadbase/util/package-summary.html] package), which implements the `ISpreadSheetModel` interface.

Note

This method is applicable only to live Java program spreadsheet objects that contain data to be plotted, and hence complements the methods for reading data from a database or data file in spreadsheet format.

```
String[] columnVals = {"quantity", "high"};
String[] rowVals = {"coffee", "Soft Drinks", "Fruit
 Juice", "Water", "beer"};
Double[][] vals = { {new Double(1), new Double(30)},

                        {new Double(3), new Double(33)},
                        {new Double(7), new Double(34)},
                        {new Double(8), new Double(40)},
                        {new Double(8), new Double(40)} };

// Please see quadbase.util.SimpleSpreadSheet
sss = new SimpleSpreadSheet(rowVals, columnVals, vals);
```

Here the data is given in a spreadsheet format and looks like the table below:

	quantity	high
coffee	1	30
Soft Drinks	3	33
Fruit Juice	7	34
Water	8	40
beer	8	40

Data in Spreadsheet Format

EspressReport then transposes the data (this is done internally) so that the data is changed to the following:

coffee	quantity	1
coffee	high	30
Soft Drinks	quantity	3
Soft Drinks	high	33
Fruit Juice	quantity	7
Fruit Juice	high	24
Water	quantity	8
Water	high	40
beer	quantity	8
beer	high	40

Data After Transpose

The data mapping for the chart is then done with respect to the transposed data.

Use the following constructor to create the `QbChart` object:

```
public QbChart(Applet applet, int dimension, int chartType,
 ISpreadSheetModel spreadsheet, IColumnMap colMap, String templateFile);
```

3.B.7. Data from Enterprise Java Beans (EJBs)

Data can be passed from an EJB data source to the chart by allowing users to query data directly from an entity bean. To add an EJB as a data source, the EJB must first be deployed in the application server and the client JAR file containing the appropriate stub classes must be added to your classpath (or the `-classpath` argument of the EspressManager batch file when using the API in conjunction with EspressManager).

Unlike reports, it is not necessary to create an `EJBInfo` object to store the connection information. Provide the names directly into the following constructor to create a `QbChart` object.

```
public QbChart(Applet applet,
               int dimension,
               int chartType,
               String jndiName,
               String homeName,
               String remoteName,
               String selectedMethodName,
               Object[] selectedMethodParamVal,
               IColumnMap cmap,
               String template);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/DataFromEJB.zip]

The above code can be run both as an application and as a JNLP (More about Applets in JNLP: Section 1.2.7 - Run Applets in WebStart with JNLP file).

The constructor for this class is:

```
public QbChart(Applet applet, int dimension, int chartType, String jndiName,
 String homeName, String remoteName, String selectedMethodName, Object[]
 selectedMethodParamVal, IColumnMap cmap, String template);
```

3.B.8. Data from multiple Data Sources

EspressReport also provides a functionality to merge multiple data sources together. You can create a `DataSheet` object from any combination of data sources, for example, data file, database or `IResultSet` (see “Data Passed as an Argument” section). You can use a `QbChart` constructor to create a chart object from an array of `DataSheet`.

The following example program fragment demonstrates how to combine the data from the examples in the above sections.

```
// Declaration of DataSheet object to be used to merge data
DataSheet dataSheet[] = new DataSheet[3]; // Declaration For Database (Data
 From a Database section)
DBInfo dbinfo = new DBInfo("jdbc:quadbase:/machine/schema",
"quadbase.jdbc.QbDriver", "myName","myPassword","select * from sales"); //
Declaration For Data Passed as an Argument (Data Passed as an Argument
 section)
String dataType[] = {"varchar", "decimal"};
String fieldName[] = {"People", "Sales"};
String records[][] = {{"Peter", "93"}, {"Peter", "124"},
                      {"John" , "110"}, {"John" , "130"},
                      {"Mary" , "103"}, {"Mary" , "129"}};
DbData data = new DbData(dataType, fieldName, records); // Create DataSheet
 from data file (Data From a Text File section)
// DataSheet(Applet applet, String dataFile)
dataSheet[0] = new DataSheet(this, "help/examples/data/Columnar1.dat"); //
 Create DataSheet from database (Data From a Database section)
// DataSheet(Applet applet, IDatabaseInfo dbInfo)
dataSheet[1] = new DataSheet(this, dbinfo); // Create DataSheet from
 IResultSet  (Data Passed as an Argument section)
// DataSheet(Applet applet, IResultSet data)
dataSheet[2] = new DataSheet(this, data); // create QbChart
QbChart chart = new QbChart

                        (this,                  // applet
                        QbChart.VIEW2D,         // Two-Dimensional
                        QbChart.PIE             // Pie Chart
                        dataSheet,              // DataSheet
                        colInfo,                // column information
                        null);                  // No template
```

Note

After you have created a `DataSheet` object, you can modify it (add/delete/update row values) by using DataSheet API. Data is not refreshable if merging data from `IResultSet`.

3.B.9. Data in Spreadsheet Format

Data obtained using the above methods may also be interpreted as a spreadsheet. A spreadsheet has a grid structure, much like a table. The leftmost column and first row in a spreadsheet contain labels (or headings). Each cell represents a distinct data point comprising its row label, column label, and the cell value (in contrast to the normal tabular notation, where each row represents a distinct data point).

Consider the following example:

Date	Nasdaq	Dow	SP500
"12/04/2000"	2304	10503	1240
"12/05/2000"	2344	10486	1239
"12/06/2000"	2344	10458	1224
------	------	------	------

Suppose the data in your database is arranged as four columns as shown. You can plot the three indices (Nasdaq, Dow, and SP500) as three lines with Date as the category axis if you treat the data as in spreadsheet format.

Several QbChart constructors are provided to deal with this issue. The three main ones are listed below.

```
QbChart(java.applet.Applet applet, int dimension, int chartType, IResultSet
 data, boolean

      doTransposeData, IColumnMap cmap, java.lang.String template);

QbChart(java.applet.Applet applet, int dimension, int chartType, int
 fileType,

      java.lang.String filename, boolean doTransposeData, IColumnMap cmap,
 java.lang.String
      template);

QbChart(java.applet.Applet applet, int dimension, int chartType,
 IDatabaseInfo dbinfo, boolean

      doTransposeData, IColumnMap cmap, java.lang.String template);
```

To specify that data is in spreadsheet format, you need to set the doTransposeData flag to true.

The above constructors transpose all the columns from the second column to the last column into a 3-column table. Thus, the data type from the second column onwards must be numeric otherwise the transpose will not be successful.

EspressReport also allows transposing of selective columns. The columns to be transposed must share the same data type. After transposing, the original columns are removed and the new columns inserted at the end of the table data. Selective transposing can be done only once; you cannot transpose certain numeric columns and then try to transpose other numeric or string columns again.

As with the complete transposing, QbChart has several constructors to allow selective transposing. The three main ones are listed below:

```
QbChart(java.applet.Applet applet, int dimension, int chartType, IResultSet
 data, boolean

      doTransposeData, int[] transposeCol, IColumnMap cmap, java.lang.String
 template);

QbChart(java.applet.Applet applet, int dimension, int chartType, int
 fileType,

      java.lang.String filename, boolean doTransposeData, int[]
 transposeCol, IColumnMap cmap,
      java.lang.String template);

QbChart(java.applet.Applet applet, int dimension, int chartType,
 IDatabaseInfo dbinfo, boolean

      doTransposeData, int[] transposeCol, IColumnMap cmap, java.lang.String
 template);
```

To specify the columns to be transposed, you need to pass in an array containing the column indices for transposeCol and set the doTransposeData flag to true.

3.B.10. Transposing Data

EspressReport allows data to be obtained from various data sources. You can also “transpose” the data (i.e., transform the data so that the column names become part of the data) before passing the data to the desired chart type.

When the data is transposed, the original data columns (used in the transpose) are removed from the dataset and two new columns are added at the end. These two new columns would contain the transposed column names as well as the values of the original columns. For example, if the original data set has five columns and three are selected for transpose, the new data set would have five minus three (the number of transposed columns) plus two (the new columns added) or four columns.

When transposing data columns they are two things to note:

- The data columns must all have the same datatype;
- The column index passed to the chart's column mapping will refer to the new dataset (and not the original data set).

The sections below describe the various ways the data can be transposed:

3.B.10.1. Non-Selective Transposing

In this scenario, all the columns, except for the very first column (Column 0) is transposed. For example, given the data below:

Product	January	February	March
Chairs	$3872.35	$3962.21	$4218.57
Tables	$6534.98	$6018.43	$5928.71

the transposed data will appear as follows:

Product	ColumnLabel	Value
Chairs	January	$3872.35
Chairs	February	$3962.21
Chairs	March	$4218.57
Tables	January	$4218.57
Tables	February	$6018.43
Tables	March	$5928.71

To non-selectively transpose the data using the API, you use any `QbChart` constructor that has a `doTransposeData` boolean parameter, such as the following constructor:

```
QbChart(java.applet.Applet applet, int dimension, int chartType, int
 fileType, java.lang.String filename, boolean doTransposeData, IColumnMap
 cmap, java.lang.String template)
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/NonTranspose.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/NonTranspose.png]

Please note that the column index passed in `IColumnMap` would refer to the new dataset.

3.B.10.2. Selective Transposing

In this scenario, you choose which columns are to be transposed. You can only transpose those columns that share the same data type. With selective transposing, you can choose the very first column to be transposed as well. For example, given the data below:

Category	Product	January	January	March
Chairs	Side Chairs	$3872.35	$3962.21	$4218.57
Chairs	Arm Chairs	$2654.84	$1924.83	$2543.24
Tables	Round Tables	$6534.98	$6018.43	$5928.71
Tables	Rectangular Tables	$10227.32	$9721.83	$11748.93

After transposing the numeric columns, the transposed data will appear as follows:

Category	Product	ColumnLabel	Value
Chairs	Side Chairs	January	$3872.35
Chairs	Side Chairs	February	$3962.21
Chairs	Side Chairs	March	$4218.57
Chairs	Arm Chairs	January	$2654.84
Chairs	Arm Chairs	February	$1924.83
Chairs	Arm Chairs	March	$2543.24
Tables	Round Tables	January	$6534.98
Tables	Round Tables	February	$6018.43
Tables	Round Tables	March	$5928.71
Tables	Rectangular Tables	January	$10227.32
Tables	Rectangular Tables	February	$9721.83
Tables	Rectangular Tables	March	$11748.93

To selectively transpose the data using the API, you use any `QbChart` constructor that has a `doTransposeData` boolean parameter and integer array that takes the indices of the columns to be transposed, such as the following constructor:

```
QbChart(java.applet.Applet applet, int dimension, int chartType, int
 fileType, java.lang.String filename, boolean doTransposeData, int[]
 transposeCol, IColumnMap cmap, java.lang.String template)
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/Transpose.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/Transpose.png]

Please note that the column index passed in `IColumnMap` would refer to the new dataset.

3.C. Creating the Chart

To create a new chart, you must specify the chart type, the dimension of the chart, the input data source information, and the column mapping for the chart template. In this appendix, we look at the various chart types and the methods used to map the columns. There are also a number of fully functional examples in this appendix. Note that unless otherwise noted, all examples use the Woodview HSQL database, which is located in the `<EspressReportInstall>/help/examples/DataSources/database` directory. In order to run the examples, you'll need to add database HSQL JDBC driver (`hsqldb.jar`) to your classpath. The driver is located in the `<EspressReportInstall>/lib` directory.

Charts can be either two or three-dimensional. Given below are the dimensions and the constant to select that dimension:

Two-Dimensional	QbChart.VIEW2D
Three-Dimensional	QbChart.VIEW3D

Charts have one of the following chart types. Given below are the different dimensions and the constants to select them:

Column Chart	`QbChart.COL`
Bar Chart	`QbChart.BAR`
Line Chart	`QbChart.LINE`

Area Chart	`QbChart.AREA`
Pie Chart	`QbChart.PIE`
XY(Z) Scatter Chart	`QbChart.SCATTER`
Stack Column Chart	`QbChart.STACKCOL`
Stack Bar Chart	`QbChart.STACKBAR`
Stack Area Chart	`QbChart.STACKAREA`
High Low Chart	`QbChart.HILOW`
HLCO Chart	`QbChart.HLCO`
Percentage Column Chart	`QbChart.PERCENTCOL`
Surface Chart	`QbChart.SURFACE`(Three-Dimensional Only)
Bubble Chart	`QbChart.BUBBLE`(Two-Dimensional Only)
Overlay Chart	`QbChart.OVERLAY`(Two-Dimensional Only)
Box Chart	`QbChart.BOX`(Two-Dimensional Only)
Radar Chart	`QbChart.RADAR`(Two-Dimensional Only)
Dial Chart	`QbChart.DIAL`(Two-Dimensional Only)
Gantt Chart	`QbChart.GANTT`(Two-Dimensional Only)
Polar Chart	`QbChart.POLAR`(Two-Dimensional Only)

For more information on the different chart types, please refer to the Section 3.4 - Chart Types and Data Mapping.

Each chart type requires two (or more) data columns to be mapped in order to create a chart object. For more detail on the mapping (and for a definition of the terms used here), please refer to the Section 3.4 - Chart Types and Data Mapping.

To define the Column Mapping for the chart template, the `ColInfo` class (located in the `quadbase.reportdesigner.util` package) is used.

The columns (from the data source) are numbered from left to right, starting with Column “0”. The column positions, passed to the `ColInfo` object, are based on the data table. Note that the parameters for `ColInfo` objects are “-1” by default. A negative column position indicates that the particular parameter is not being used.

Caution

As you may know, creating objects in java is a resource intensive operation. It is always recommended that you do not create too many objects. One way to conserve resource is to reuse `QbChart` objects whenever possible. For example, if a lot of your users request for a simple Columnar Chart in your web site, instead of creating a new `QbChart` object when each such request is received, you can have one (or a limited number of) such `QbChart` object(s) created and reuse the object(s) by simply modifying the data and attributes of the ahrt for each particular request.

3.C.1. Column, Bar, Line, Area, Pie and Overlay Charts

Column/Bar/Line/Area/Pie/Overlay charts need a `category` and a `value` axis. Those are the minimum requirements for constructing such types of charts. You can also add in a `series` and a `subvalue` if needed. The `subvalue` refers to the secondary axis i.e., it contains the column that gets mapped to the secondary axis.

3.C.1.1. Column Mapping

For instance, given the data shown below:

Order #	Product	Units Ordered	Units Shipped
12	Chair	15	12
12	Table	34	26
14	Chair	8	8
14	Table	23	14

Original Data

The following code sets the category to be Column 1 (Product), the series to be Column 0 (Order #), the value to be Column 3 (Units Shipped) and the subvalue to be Column 2 (Units Ordered):

```
ColInfo colInfo = new ColInfo();
colInfo.category = 1;
colInfo.series = 0;
colInfo.value = 3;
colInfo.subvalue = 2;
```

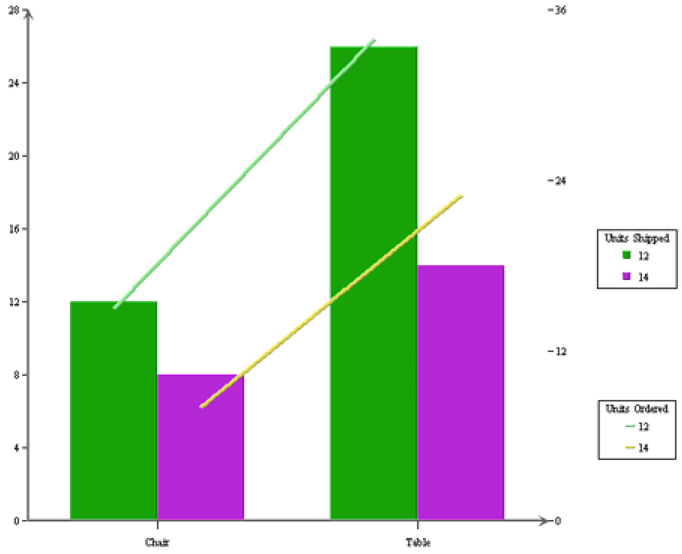
3.C.1.2. Creating the Chart

Constructing a Column chart is relatively straight forward. We have already discussed how to set the `ColInfo` array and how to obtain the data in previous sections. The following code demonstrates how to create the aforementioned Column chart.

```
Component doDataFromArguments(Applet applet) {


    // Do not connect to EspressManager
    QbChart.setEspressManagerUsed(false);

    // Column Mapping
    ColInfo colInfo = new ColInfo();
    colInfo.category = 1;
    colInfo.series = 0;
    colInfo.value = 3;
    colInfo.subvalue = 2;

    String dataType[] = {"integer", "varchar", "decimal", "decimal"};
    String fieldName[] = {"Order #", "Product", "Units Ordered", "Units
 Shipped"};
    String records[][] = {{"12", "Chair","15", "12"},
 {"12", "Table","34", "26"},
                        {"14" , "Chair","8", "8"}, {"14"
 , "Table","23", "14"}};
    DbData data = new DbData(dataType, fieldName, records);
    QbChart chart = new QbChart
                             (applet,              // Applet
                             QbChart.VIEW2D,       // Two-Dimensional
                             QbChart.COL,          // Column Chart
                             data,                 // Data
                             colInfo,              // Column information
                             null);                // No specified
 template
    return chart;

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ColumnChart.zip]

When the above code is run as a Java Web Start Application, it will produce a two-dimensional Column chart shown below:



Generated Column Chart

Note that the chart created is a default chart without any formatting done to it.

3.C.2. Radar Charts

The `ColInfo` parameters for a Radar chart are the same as the parameters for a Column/Bar/Line/Area chart **except** the `subvalue`. A secondary axis cannot be defined.

3.C.2.1. Column Mapping

For instance, given the data shown below:

Order #	Product	Units Ordered
12	Chair	15
12	Table	34
12	Cabinet	21
12	Dresser	24
14	Chair	23
14	Table	23
14	Cabinet	16
14	Dresser	19

Original Data

The following code sets the category to be Column 1 (Product), the series to be Column 0 (“Order #”) and the value to be Column 2 (“Units Ordered”):

```
ColInfo colInfo = new ColInfo();
colInfo.category = 1;
colInfo.series = 0;
colInfo.value = 2;
```

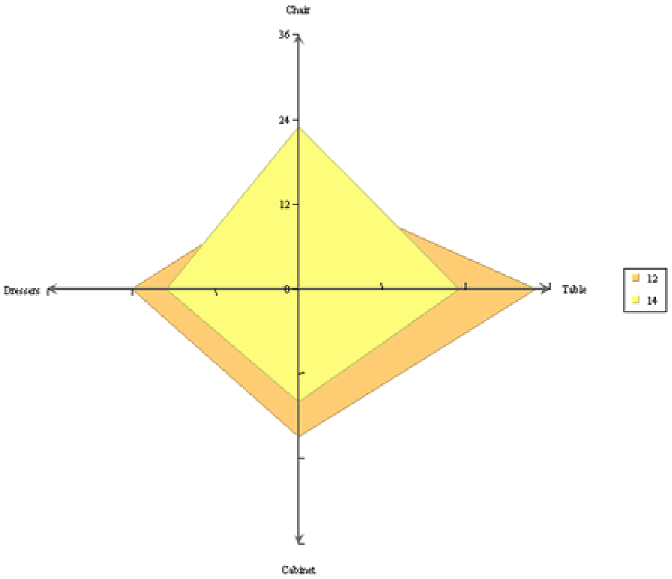
3.C.2.2. Creating the Chart

The following code demonstrates how to create the aforementioned Radar chart.

```
Component doDataFromArguments(Applet applet) {


    // Do not connect to EspressManager
    QbChart.setEspressManagerUsed(false);

    // Column Mapping
    ColInfo colInfo = new ColInfo();
    colInfo.category = 1;
    colInfo.series = 0;
    colInfo.value = 2;

    String dataType[] = {"integer", "varchar", "decimal"};
    String fieldName[] = {"Order #", "Product", "Units Ordered"};
    String records[][] = {{"12", "Chair", "15"}, {"12", "Table", "34"},
                         {"12" , "Cabinet", "21"}, {"12" , "Dresser", "24"},
                         {"14" , "Chair", "23"}, {"14" , "Table", "23"},
                         {"14" , "Chair", "23"}, {"14" , "Table", "23"}};
    DbData data = new DbData(dataType, fieldName, records);
    QbChart chart = new QbChart
                                (applet,              // Applet
                                QbChart.VIEW2D,       // Two-Dimensional
                                QbChart.RADAR,        // Radar Chart
                                data,                 // Data
                                colInfo,              // Column information
                                null);                // No specified
 template
    return chart;

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/RadarChart.zip]

When the above code is run as a Java Web Start Application, it will produce a two-dimensional Radar chart shown below:



Generated Radar Chart

Note that the chart created is a default chart without any formatting done to it.

3.C.3. XY(Z) Scatter Charts

Scatter charts need a `xvalue` and a `yvalue` axis. Those are the minimum requirements for constructing such types of charts. You can also add in a series and a `zvalue` (for three-dimensional Scatter charts) axis if needed.

3.C.3.1. Column Mapping

For instance, given the data shown below:

Season	High Temperature	Average Temperature	Low Temperature
Summer	110	101	96
Fall	103	85	78
Winter	85	75	67
Spring	93	88	81

Original Data

The following code sets the xvalue to be Column 2 (Average Temperature), the yvalue to be Column 1 (High Temperature), and the zvalue to be Column 3 (Low Temperature):

```
ColInfo colInfo = new ColInfo();
colInfo.xvalue = 2;
colInfo.yvalue = 1;
colInfo.zvalue = 3;
```

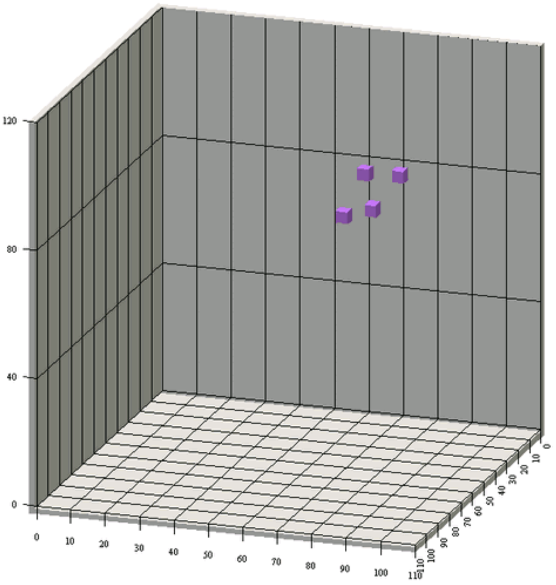
3.C.3.2. Creating the Chart

The following code demonstrates how to create the aforementioned Scatter chart.

```
Component doDataFromArguments(Applet applet) {


    // Do not connect to EspressManager
    QbChart.setEspressManagerUsed(false);

    // Column Mapping
    ColInfo colInfo = new ColInfo();
    colInfo.xvalue = 2;
    colInfo.yvalue = 1;
    colInfo.zvalue = 3;

    String dataType[] = {"varchar", "integer", "integer", "integer"};
    String fieldName[] = {"Season", "High Temperature", "Average
 Temperature", "Low Temperature"};
    String records[][] = {{"Summer", "110", "101", "96"},
 {"Fall", "103", "85", "78"},
                        {"Winter", "85", "75", "67"},
 {"Spring", "93", "88", "81"}};
    DbData data = new DbData(dataType, fieldName, records);
    QbChart chart = new QbChart
                            (applet,            // Applet
                            QbChart.VIEW3D,     // Three-Dimensional
                            QbChart.SCATTER,    // Scatter Chart
                            data,               // Data
                            colInfo,            // Column information
                            null);              // No specified
 template
    return chart;

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/XYZScatterChart.zip]

When the above code is run as a Java Web Start Application, it will produce a three-dimensional Scatter chart shown below:



Generated Scatter Chart

Note that the chart created is a default chart without any formatting done to it.

3.C.4. Stack Column, Percentage Column, Stack Bar and Stack Area Charts

In addition to the parameters for a Column/Line/Area chart, a Stack Column/Percentage Column/Stack Bar/Stack Area also requires the `SumBy` parameter to be set. The minimum requirements for creating a Stack Column/Percentage Column/Stack Bar/Stack Area chart are the `category`, the `value` and the `sumby` parameters. You can also add in a `series` and a `subvalue` if needed.

3.C.4.1. Column Mapping

For instance, given the data shown below:

Day	Drink	Total	Average
Monday	Water	101	5.4
Monday	Coffee	85	2.4
Tuesday	Water	143	6.7
Tuesday	Coffee	92	2.5
Wednesday	Water	186	7.6
Wednesday	Coffee	121	4.2
Thursday	Water	173	6.3
Thursday	Coffee	75	1.1
Friday	Water	88	3.6
Friday	Coffee	193	5.9
Saturday	Water	152	7.3
Saturday	Coffee	57	1.6
Sunday	Water	194	8.8
Sunday	Coffee	25	0.6

Original Data

The following code sets the category to be Column 0 (“Day”), the value to be Column 2 (“Total”), the sumby to be Column 1 (“Drink”) and the subvalue to be Column 3 (“Average”):

```
ColInfo colInfo = new ColInfo();
colInfo.category = 0;
colInfo.value = 2;
colInfo.subvalue = 3;
colInfo.sumBy = 1;
```

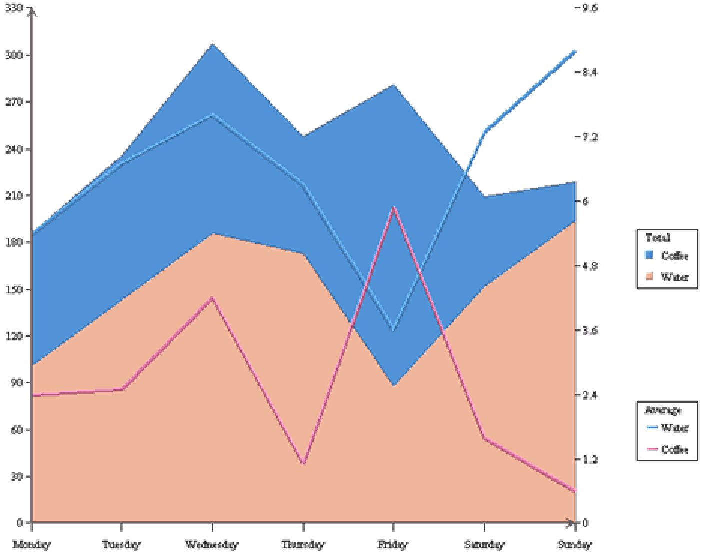
3.C.4.2. Creating the Chart

The following code demonstrates how to create the aforementioned Stack Area chart.

```
Component doDataFromArguments(Applet applet) {


    // Do not connect to EspressManager
    QbChart.setEspressManagerUsed(false);

    // Column Mapping
    ColInfo colInfo = new ColInfo();
    colInfo.category = 0;
    colInfo.value = 2;
    colInfo.subvalue = 3;
    colInfo.sumBy = 1;

    String dataType[] = {"varchar", "varchar", "integer", "double"};
    String fieldName[] = {"Day", "Drink", "Total","Average"};
    String records[][] = {{"Monday", "Water", "101","5.4"},
 {"Monday", "Coffee", "85","2.4"},
                         {"Tuesday", "Water", "143","6.7"},
 {"Tuesday", "Coffee", "92","2.5"},
                         {"Wednesday", "Water", "186","7.6"},
 {"Wednesday", "Coffee", "121","4.2"},
                         {"Thursday", "Water", "173","6.3"},
 {"Thursday", "Coffee", "75","1.1"},
                         {"Friday", "Water", "88","3.6"},
 {"Friday", "Coffee", "193","5.9"},
                         {"Saturday", "Water", "152","7.3"},
 {"Saturday", "Coffee", "57","1.6"},
                         {"Sunday", "Water", "194","8.8"},
 {"Sunday", "Coffee", "25","0.6"}};
    DbData data = new DbData(dataType, fieldName, records);
    QbChart chart = new QbChart
                                (applet,              // Applet
                                QbChart.VIEW2D,       // Two-Dimensional
                                QbChart.STACKAREA,    // Stack Area Chart
                                data,                 // Data
                                colInfo,              // Column information
                                null);                // No specified
 template
    return chart;

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/StackAreaChart.zip]

When the above code is run as a Java Web Start Application, it will produce a two-dimensional Stack Area chart shown below:



Generated Stack Area Chart

Note that the chart created is a default chart without any formatting done to it.

3.C.5. Dial Charts

The `ColInfo` parameters for a Dial chart are the same as the parameters for a Radar chart **except** the `series`. A `series` column cannot be defined.

3.C.5.1. Column Mapping

For instance, given the data shown below:

Product	Units Ordered
Chair	15
Table	34
Cabinet	21
Dresser	24

Original Data

The following code sets the category to be Column 0 (“Product”) and the value to be Column 1 (“Units Ordered”):

```
ColInfo colInfo = new ColInfo();
colInfo.category = 0;
colInfo.value = 1;
```

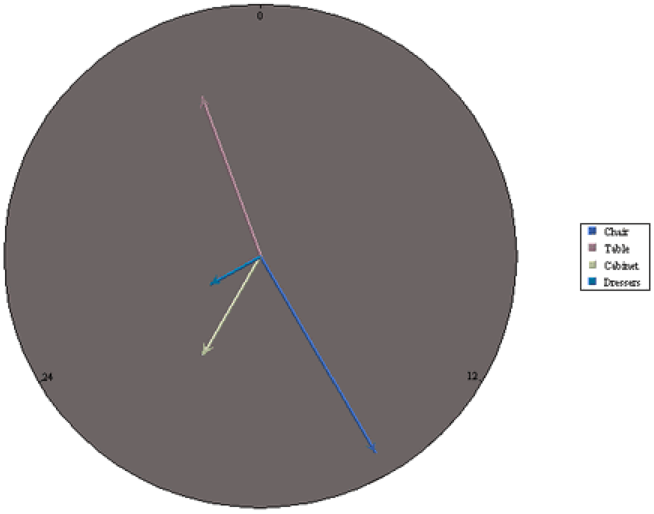
3.C.5.2. Creating the Chart

The following code demonstrates how to create the aforementioned Dial chart.

```
Component doDataFromArguments(Applet applet) {


    // Do not connect to EspressManager
    QbChart.setEspressManagerUsed(false);

    // Column Mapping
    ColInfo colInfo = new ColInfo();
    colInfo.category = 0;
    colInfo.value = 1;

    String dataType[] = {"varchar", "integer"};
    String fieldName[] = {"Product", "Units Ordered"};
    String records[][] = {{"Chair", "15"}, {"Table", "34"},
                         {"Cabinet", "21"}, {"Dresser", "24"}};
    DbData data = new DbData(dataType, fieldName, records);
    QbChart chart = new QbChart
                                (applet,                // Applet
                                QbChart.VIEW2D,         // Two-Dimensional
                                QbChart.DIAL,           // Dial Chart
                                data,                   // Data
                                colInfo,                // Column information
                                null);                  // No specified
 template
    return chart;

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/DialChart.zip]

When the above code is run as a Java Web Start Application, it will produce a two-dimensional Dial chart shown below:



Generated Dial Chart

Note that the chart created is a default chart without any formatting done to it.

3.C.6. Box Charts

The ColInfo parameters for a Box chart are the same as the parameters for a Column/Bar/Line/Area/Pie/Overlay chart **except** the `series`. A `series` column cannot be defined.

3.C.6.1. Column Mapping

For instance, given the data shown below:

Subject	Student	Score
Math	A.S.	65
Math	T.E.	75

Subject	Student	Score
Math	V.Q.	83
Math	X.C.	87
Math	I.Z.	93
Science	A.S.	86
Science	T.E.	90
Science	V.Q.	73
Science	X.C.	95
Science	I.Z.	84

Original Data

The following code sets the category to be Column 0 (“Subject”) and the value to be Column 2 (“Score”):

```
ColInfo colInfo = new ColInfo();
colInfo.category = 0;
colInfo.value = 2;
```

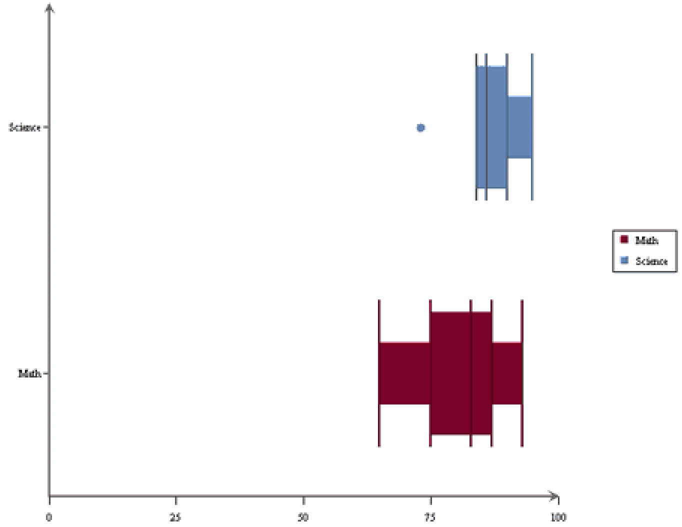
3.C.6.2. Creating the Chart

The following code demonstrates how to create the aforementioned Box chart.

```
Component doDataFromArguments(Applet applet) {


    // Do not connect to EspressManager
    QbChart.setEspressManagerUsed(false);

    // Column Mapping
    ColInfo colInfo = new ColInfo();
    colInfo.category = 0;
    colInfo.value = 2;

    String dataType[] = {"varchar", "varchar", "integer"};
    String fieldName[] = {"Subject", "Student", "Score"};
    String records[][] = {{"Math", "A.S.", "65"}, {"Math", "T.E.", "75"},
                         {"Math", "V.Q.", "83"}, {"Math", "X.C.", "87"},
                         {"Math", "I.Z.", "93"}, {"Science", "A.S.", "86"},
                         {"Science", "T.E.", "90"},
 {"Science", "V.Q.", "73"},
                         {"Science", "X.C.", "95"},
 {"Science", "I.Z.", "84"}};
    DbData data = new DbData(dataType, fieldName, records);
    QbChart chart = new QbChart
                                (applet,              // Applet
                                QbChart.VIEW2D,       // Two-Dimensional
                                QbChart.BOX,          // Box Chart
                                data,                 // Data
                                colInfo,              // Column information
                                null);                // No specified
 template
    return chart;

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/BoxChart.zip]

When the above code is run as a Java Web Start Application, it will produce a two-dimensional Box chart shown below:



Generated Box Chart

Note that the chart created is a default chart without any formatting done to it.

3.C.7. Bubble Charts

The ColInfo parameters for a Bubble chart are the same as the parameters for a three-dimensional Scatter Chart. The `xvalue`, `yvalue` and `zvalue` parameters are necessary to draw a Bubble Chart. Note that for a bubble chart, the `zvalue` refers to the Bubble Size.

3.C.7.1. Column Mapping

For instance, given the data shown below:

Drink	High	Average	Low
Water	3	2	2
Soda	1	1	0
Coffee	5	2	1
Tea	3	1	1

Original Data

The following code sets the series to be Column 0 (“Drink”), xvalue to be Column 1 (“quote>High”), the yvalue to be Column 3 (“Low”), and the zvalue to be Column 2 (“Average”):

```
ColInfo colInfo = new ColInfo();
colInfo.xvalue = 1;
colInfo.yvalue = 3;
colInfo.zvalue = 2;
colInfo.series = 0;
```

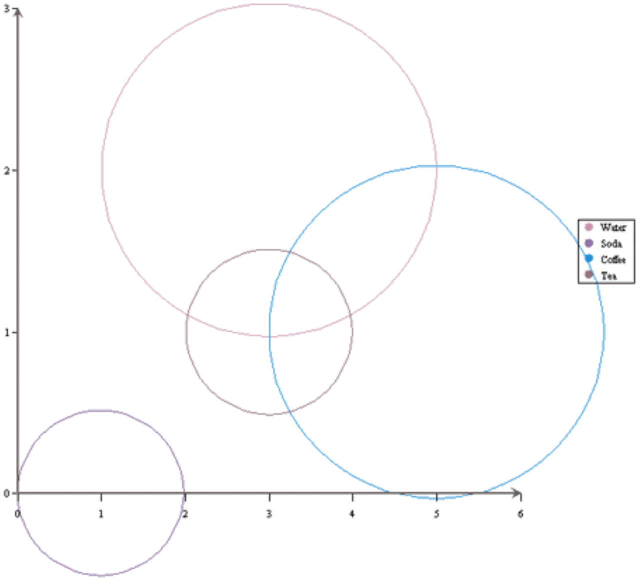
3.C.7.2. Creating the Chart

The following code demonstrates how to create the aforementioned Bubble chart.

```
Component doDataFromArguments(Applet applet) {


    // Do not connect to EspressManager
    QbChart.setEspressManagerUsed(false);

    // Column Mapping
    ColInfo colInfo = new ColInfo();
    colInfo.xvalue = 1;
    colInfo.yvalue = 3;
    colInfo.zvalue = 2;
    colInfo.series = 0;

    String dataType[] = {"varchar", "integer", "integer", "integer"};
    String fieldName[] = {"Drink", "High", "Average", "Low"};
    String records[][] = {{"Water", "3", "2", "2"}, {"Soda", "1", "1", "0"},
                         {"Coffee", "5", "2", "1"},
 {"Tea", "3", "1", "1"}};
    DbData data = new DbData(dataType, fieldName, records);
    QbChart chart = new QbChart
                                (applet,              // Applet
                                QbChart.VIEW2D,       // Two-Dimensional
                                QbChart.BUBBLE,       // Bubble Chart
                                data,                 // Data
                                colInfo,              // Column information
                                null);                // No specified
 template
    return chart;

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/BubbleChart.zip]

When the above code is run as a Java Web Start Application, it will produce a two-dimensional Bubble chart shown below:



Generated Bubble Chart

Note that the chart created is a default chart without any formatting done to it.

3.C.8. High-Low and HLCO Charts

The `ColInfo` parameters for a High-Low/HLCO chart differ from the parameters of the other chart types. Here the value part of the chart is further subdivided into an `open` value, a `close` value, a `high` value and a `low` value.

The minimum requirements for a High-Low chart are the category, the high value and the low value parameters (for a HLCO, the close value and open value parameters are also required). You can also additionally add in the series and subvalue parameters as well.

3.C.8.1. Column Mapping

For instance, given the data shown below:

Day	Company	High	Low	Close	Open	Volume
2001-01-01	ABC	1.33	1.18	1.22	1.23	43723
2001-01-01	DEF	9.24	8.74	9.16	8.89	18478
2001-01-01	GHI	2.20	1.82	2.14	1.97	46743
2001-01-02	ABC	1.87	0.79	1.63	1.12	33605
2001-01-02	DEF	9.48	8.12	8.93	8.66	16758
2001-01-02	GHI	2.47	2.32	2.44	2.34	60671
2001-01-03	ABC	2.47	0.22	0.44	0.63	45211
2001-01-03	DEF	9.94	9.92	9.93	9.93	10697
2001-01-03	GHI	2.48	2.40	2.46	2.44	45238
2001-01-04	ABC	1.8	1.38	1.79	1.44	50224
2001-01-04	DEF	9.49	8.87	8.94	8.93	11868
2001-01-04	GHI	2.06	1.45	1.96	1.46	62053
2001-01-05	ABC	1.23	0.58	0.79	0.79	37285
2001-01-05	DEF	9.94	8.61	8.08	8.94	10476
2001-01-05	GHI	2.8	1.58	2.45	1.96	47117

Original Data

The following code sets the category to be Column 0 (Day), the series to be Column 1 (Company), the High to be Column 2 (High), the Low to be Column 3 (Low), the Close to be Column 4 (Close), the Open to be Column 5 (Open) and the subvalue to be Column 6 (Volume):

```
ColInfo colInfo = new ColInfo();
colInfo.category = 0;
colInfo.series = 1;
colInfo.high = 2;
colInfo.low = 3;
colInfo.close = 4;
colInfo.open = 5;
colInfo.subvalue = 6;
```

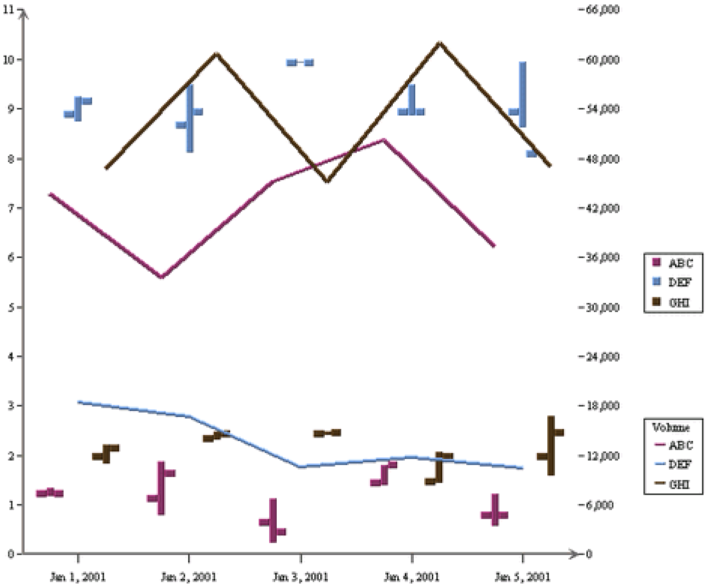
3.C.8.2. Creating the Chart

The following code demonstrates how to create the aforementioned HLCO chart.

```
Component doDataFromArguments(Applet applet) {


    // Do not connect to EspressManager
    QbChart.setEspressManagerUsed(false);

    // Column Mapping
    ColInfo colInfo = new ColInfo();
    ColInfo colInfo = new ColInfo();
    colInfo.category = 0;
    colInfo.series = 1;
    colInfo.high = 2;
    colInfo.low = 3;
    colInfo.close = 4;
    colInfo.open = 5;
    colInfo.subvalue = 6;

    String dataType[] =
 {"date", "varchar", "double", "double", "double", "double", "integer"};
    String fieldName[] =
 {"Day", "Company", "High", "Low", "Close", "Open", "Volume"};
    String records[][] =
 {{"2001-01-01", "ABC", "1.33", "1.18", "1.22", "1.23", "43723"},

 {"2001-01-01", "DEF", "9.24", "8.74", "9.16", "1.23", "18478"},

 {"2001-01-01", "GHI", "2.20", "1.82", "2.14", "1.23", "46743"},

 {"2001-01-02", "ABC", "1.87", "0.79", "1.63", "1.23", "33605"},

 {"2001-01-02", "DEF", "9.48", "8.12", "8.93", "1.23", "16758"},

 {"2001-01-02", "GHI", "2.47", "2.32", "2.44", "1.23", "60671"},

 {"2001-01-03", "ABC", "1.12", "0.22", "0.44", "1.23", "45211"},

 {"2001-01-03", "DEF", "9.94", "9.92", "9.93", "9.93", "10697"},

 {"2001-01-03", "GHI", "2.48", "2.40", "2.46", "2.44", "45238"},

 {"2001-01-04", "ABC", "1.80", "1.38", "1.79", "1.44", "50224"},

 {"2001-01-04", "DEF", "9.49", "8.87", "8.94", "8.93", "11868"},

 {"2001-01-04", "GHI", "2.06", "1.45", "1.96", "1.46", "62053"},

 {"2001-01-05", "ABC", "1.23", "0.58", "0.79", "0.79", "37285"},

 {"2001-01-05", "DEF", "9.94", "8.61", "8.08", "8.94", "10476"},

 {"2001-01-05", "GHI", "2.80", "1.58", "2.45", "1.96", "47117"}};
    DbData data = new DbData(dataType, fieldName, records);
    QbChart chart = new QbChart
                                (applet,              // Applet
                                QbChart.VIEW2D,       // Two-Dimensional
                                QbChart.HLCO,         // HLCO Chart
                                data,                 // Data
                                colInfo,              // Column information
                                null);                // No specified
 template
    return chart;

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/HLCOChart.zip]

When the above code is run as a Java Web Start Application, it will produce a two-dimensional HLCO chart shown below:



Generated HLCO Chart

Note that the chart created is a default chart without any formatting done to it.

3.C.9. Surface Charts

The ColInfo parameters for a Surface chart are similar to those of a three-dimensional Scatter chart. The `xvalue`, `yvalue` and `zvalue` parameters are required to create a surface chart. However, a surface chart does not support the `series` parameter. For a surface chart, the `ColInfo` object defined is always the same no matter what the data is.

3.C.9.1. Column Mapping

For instance, given the data shown below:

	0	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	1	0	0	0	0	0
20	0	0	0	0	1	2	1	0	0	0	0
30	0	0	0	1	2	3	2	1	0	0	0
40	0	0	1	2	3	4	3	2	1	0	0
50	0	1	2	3	4	5	4	3	2	1	0
60	0	0	1	2	3	4	3	2	1	0	0
70	0	0	0	1	2	3	2	1	0	0	0
80	0	0	0	0	1	2	1	0	0	0	0
90	0	0	0	0	0	1	0	0	0	0	0
100	0	0	0	0	0	0	0	0	0	0	0

Original Data

The following code sets the mapping for the surface chart. Note that this mapping is constant for surface charts:

```
int map[] = {0, 2, 1};
ColInfo colInfo = new ColInfo(-1, map);
```

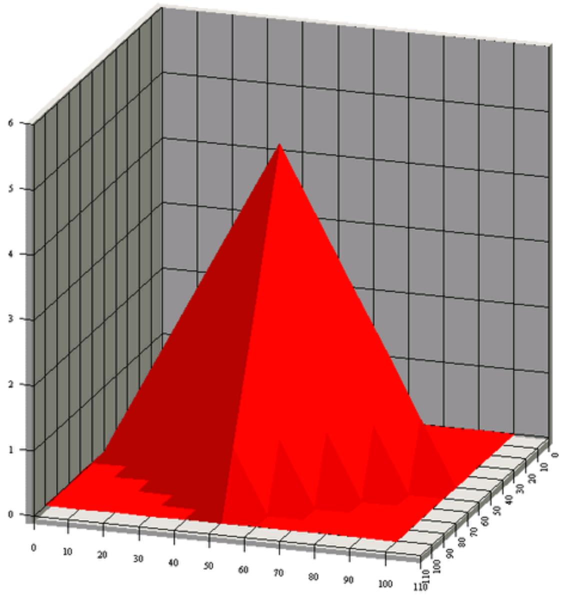
3.C.9.2. Creating the Chart

The following code demonstrates how to create the aforementioned Surface chart.

```
Component doDataFromArguments(Applet applet) {

 QbChart.setEspressManagerUsed(false);

 int map[] = { 0, 2, 1 };
 ColInfo colInfo = new ColInfo(-1, map);

 String inputFileName = "surface.dat";

 QbChart chart = null;

 try {

  chart = new QbChart(applet, // Applet
    QbChart.VIEW3D, // Three-Dimensional
    QbChart.SURFACE, // Surface Chart
    QbChart.DATAFILE, // Type of text file
    inputFileName, // Data
    false, // Do not transpose data
    colInfo, // Column information
    null); // No specified template

 } catch (Exception ex)
 {
  ex.printStackTrace();
 }

 return chart;

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/SurfaceChart.zip]

When the above code is run as a Java Web Start Application, it will produce a three-dimensional Surface chart shown below:



Generated Surface Chart

Note that the chart created is a default chart without any formatting done to it.

3.C.10. Gantt Charts

The `ColInfo` parameters for a Gantt chart are similar to those of a High-Low chart. Here the `category`, `high` (or start date) and `low` (end date) parameters are required to create a Gantt chart. You can also add a `series` parameter if needed.

3.C.10.1. Column Mapping

For instance, given the data shown below:

Project	Task	Starting Date	Ending Date
Project 1	Task A	1998-01-15	1998-03-01
Project 1	Task B	1998-02-25	1998-05-18
Project 1	Task C	1998-06-11	1998-09-29
Project 2	Task A	1998-03-15	1998-09-29
Project 2	Task B	1998-04-25	1998-08-18
Project 2	Task C	1998-08-11	1998-12-29

Original Data

The following code sets the category to be Column 1 (“Task”), the High to be Column 2 (“Starting Date”), the Low to be Column 3 (“Ending Date”) and the series to be Column 0 (“Project”).

```
ColInfo colInfo = new ColInfo();
colInfo.category = 1;
colInfo.high = 2;
colInfo.low = 3;
colInfo.series = 0;
```

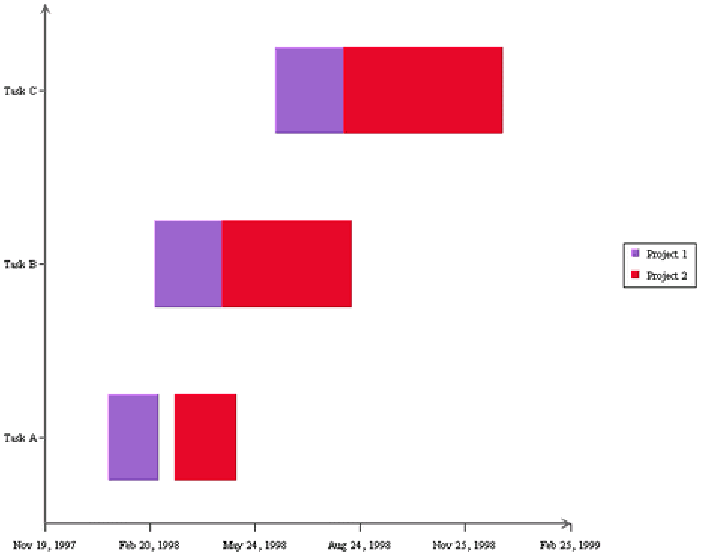
3.C.10.2. Creating the Chart

The following code demonstrates how to create the aforementioned Gantt chart.

```
Component doDataFromArguments(Applet applet) {

    QbChart.setEspressManagerUsed(false);

    // Column Mapping
    ColInfo colInfo = new ColInfo();
    colInfo.category = 1;
    colInfo.high = 2;
    colInfo.low = 3;
    colInfo.series = 0;

    String dataType[] = {"varchar", "varchar", "date", "date"};
    String fieldName[] = {"Project", "Task", "Starting Date","Ending Date"};
    String records[][] = {{"Project1", "Task A", "1998-01-15","1998-03-01"},
                         {"Project1", "Task B", "1998-02-25","1998-05-18"},
                         {"Project1", "Task C", "1998-06-11","1998-09-29"},
                         {"Project2", "Task A", "1998-03-15","1998-05-08"},
                         {"Project2", "Task B", "1998-04-25","1998-08-18"},
                         {"Project2", "Task
 C", "1998-08-11","1998-12-29"}};
    DbData data = new DbData(dataType, fieldName, records);
    QbChart chart = new QbChart
                                (applet,                      // Applet
                                QbChart.VIEW2D,       // Two-Dimensional
                                QbChart.GANTT,        // Gantt Chart
                                data,                 // Data
                                colInfo,              // Column information
                                null);                // No specified
 template
    return chart;

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/GanttChart.zip]

When the above code is run as a Java Web Start Application, it will produce a two-dimensional Gantt chart shown below:



Generated Gantt Chart

Note that the chart created is a default chart without any formatting done to it.

3.C.11. Polar Charts

The `ColInfo` parameters for a Polar chart are similar to those of a Scatter Chart. Here the `radius`, and `angle` parameters are required to create a Polar chart. You can also add a `series` parameter if needed.

3.C.11.1. Column Mapping

For instance, given the data shown below:

Radius	Angle	Series
0	2	A
90	4	A
180	6	A
270	8	A
360	10	A
45	3	B
135	5	B
225	7	B
315	9	B

Original Data

The following code sets the radius to be Column 0 (“Radius”), the Angle to be Column 1 (“Angle”), and the series to be Column 2 (“Series”).

```
ColInfo colInfo = new ColInfo(2, new int[]{0, 1});
```

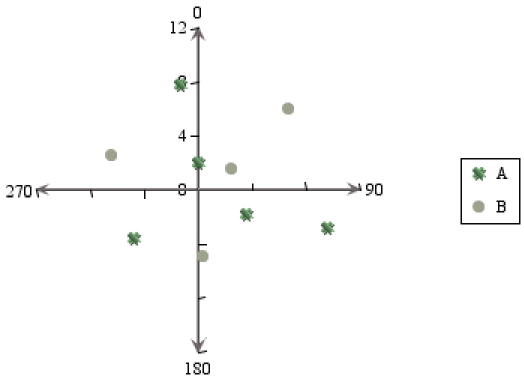
3.C.11.2. Creating the Chart

The following code demonstrates how to create the aforementioned Polar chart.

```
Component doDataFromArguments(Applet applet) {

    QbChart.setEspressManagerUsed(false);

    // Column Mapping
    ColInfo colInfo = new ColInfo(2, new int[]{0, 1});

    String dataType[] = {"integer", "integer", "varchar"};
    String fieldName[] = {"Radius", "Angle", "Series"};
    String records[][] = {{"0", "2", "A"}, {"90", "4", "A"},
 {"180", "6", "A"},
                         {"270", "8", "A"}, {"360", "10", "A"},
 {"45", "3", "B"},
                         {"135", "5", "B"}, {"225", "7", "B"},
 {"315", "9", "B"}};
    DbData data = new DbData(dataType, fieldName, records);
    QbChart chart = new QbChart
                    (applet,                // Applet
                    QbChart.VIEW2D,         // Two-Dimensional
                    QbChart.POLAR,          // Polar Chart
                    data,                   // Data
                    colInfo,                // Column information
                    null);                  // No specified template
    return chart;

}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/PolarChart.zip]

When the above code is run as a Java Web Start Application, it will produce a two-dimensional Polar chart shown below:



Generated Polar Chart

Note that the chart created is a default chart without any formatting done to it.

3.C.12. Date/Time Based Zoom Charts

EspressReport allows date/time based zooming in charts. The date/time based zooming can be applied to the chart of almost any type (except high-low, HLCO, scatter, Surface, Box, Dial, Radar, Bubble and Gantt charts). The only major requirement is that the data along the category axis be of date, time or timestamp type.

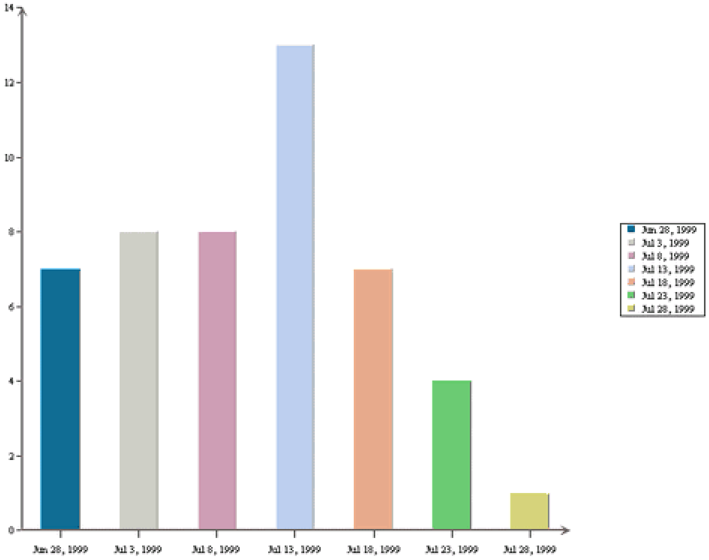
Zooming can be achieved by setting the attributes and then refreshing the chart. The attributes are set using the `quadbase.util.IZoomInfo` interface.

The following example, which can run as example or application, shows a chart that incorporates zooming. Here the default zooming shows 5 days at a time while the maximum zoom-out allowed is 1 month and the maximum zoom-in allowed is 1 day.

```
// Data passed in an array in memory
DbData chartData = new DbData(dataTypes, fieldNames, data);

// Set Column Mapping
ColInfo colInfo = new ColInfo();
colInfo.category = 1;
colInfo.value = 0;

// Create Chart
QbChart chart = new QbChart(

                    (Applet)null,              // Applet
                    QbChart.VIEW2D,            // Dimension
                    QbChart.COL,               // Chart Type
                    chartData,                 // Data
                    colInfo,                   // Column Mapping
                    null);                     // Template


// Get a handle to the Zooming interface
IZoomInfo zoomInfo = chart.gethZoomInfo();
zoomInfo.setAggregateOperator(IZoomInfo.SUM);  // Specify the aggregation
zoomInfo.setMaxScale(1, IZoomInfo.YEAR);       // Specify max zoom out
zoomInfo.setMinScale(1, IZoomInfo.DAY);        // Specify max zoom in
zoomInfo.setScale(5, IZoomInfo.DAY);           // Specify current zooming
zoomInfo.setLinearScale(true);                 // Turn on Linear Scale
zoomInfo.setZoomEnabled(true);                 // Turn on Zooming

try
{

    chart.refresh();                            // Refresh the chart with
 zooming

} catch (Exception ex)
{

    ex.printStackTrace();

}
return chart;
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ZoomChartAppendix.zip]

When the above code is run as a Java Web Start Application, it will produce a top-level chart shown below:



Generated Chart

You can then zoom in or zoom out on this chart, depending on the selections you make in the pop-up menu (which you can see by right-clicking on the chart canvas).

Please note that this is a default chart that is generated without formatting of any kind.

3.C.13. Parameterized Charts

In addition to regular queries, you can pass in queries that has one, or more, parameters and have the chart prompt the user for values for the parameters, before generating the chart. Note that only stand-alone charts can be parameterized.

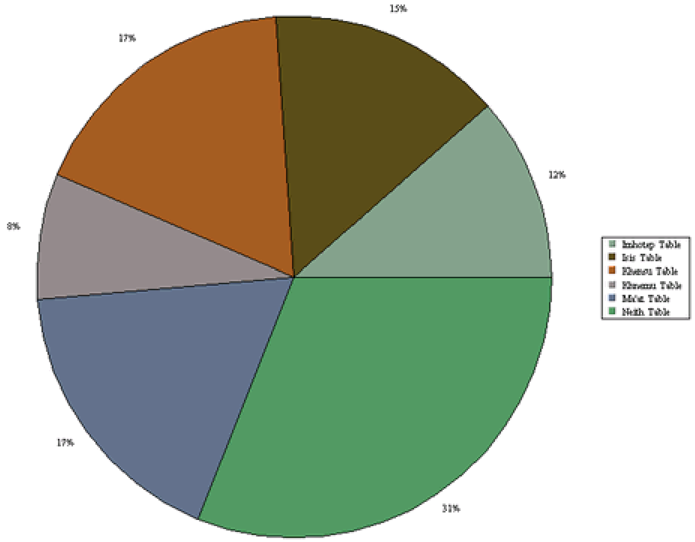
To use a parameterized query as your data source for your chart, you must create a class that implements `IQueryFileInfo` (you can use the implementation already provided, `SimpleQueryFileInfo`) and use that class to pass in the parameter information.

Given below is an example of a chart that uses a parameterized query. The database against which the query is run is WoodView HSQL, so you will need to add database HSQL JDBC driver (`hsqldb.jar`) to your classpath. The driver is located in the `<EspressReportInstall>/lib` directory.

```
Component doParameterizedChart(Applet applet) {
 // Do not use EspressManager
 QbChart.setEspressManagerUsed(false);

 // Begin Code : Adding Parameter Info
 // SimpleQueryInParam(name of Parameter, String to be displayed,
 MapToColumn?, tableName, ColumnName, SQL Type, DefaultValue, value)

 SimpleQueryInParam inParam = new SimpleQueryInParam("param", "Please
 select", true,
   "Categories", "CategoryName", Types.VARCHAR, "Arm Chairs", null);
 SimpleQueryInParam[] paramSet = { inParam };
 SimpleQueryFileInfo chartInfo = new SimpleQueryFileInfo(
   "jdbc:hsqldb:woodview",
   "org.hsqldb.jdbcDriver",
   "sa",
   "",
   "select Products.ProductName, Products.UnitsInStock from Categories,
 Products where Categories.CategoryID=Products.CategoryID and
 Categories.CategoryName=:param order by Categories.CategoryName,
 Products.ProductName;");

 chartInfo.setInParam(paramSet);

 // End Code : Adding Parameter Info // Begin Code : Setting up Column
 Mapping
 ColInfo colInfo = new ColInfo();
 colInfo.category = 0;
 colInfo.value = 1;
 // End Code : Setting up Column Mapping

 QbChart chart = new QbChart(applet, QbChart.VIEW2D, QbChart.PIE, chartInfo,
 colInfo, null);

 return chart;
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ParameterizedChart.zip]

When the above code is run as a Java Web Start Application, it will produce a chart (depending on the parameter selected) similar to the one shown below:



Generated Chart

Please note that this is a default chart that is generated without formatting of any kind.

If you add a parameterized chart to a report, then you must link a column from the main report to provide the values for the parameters defined in the chart. You do this by using the following method in `ReportChartObject`:

```
public void setParameterMap(String[] columnID);
```

The string array contains the IDs of the columns that will provide the values for the parameters in the chart.

You can also assign a parameter to have multiple values, for example, in the case where a user wants to check against a range of values rather than just a single value. The range of values is usually specified within the “IN” clause of a SQL query. Note that EspressReport only considers parameters within the “IN” clause to be multi-value.

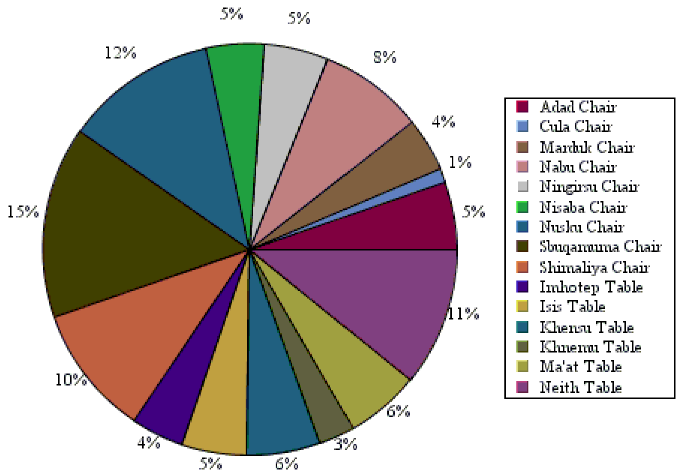
To use a multi-value parameterized query as your data source for your chart, you must create a class that implements `IQueryFileInfo` and use that class to pass in the parameter information.

Given below is an example of a chart that uses a multi-value parameterized query. The database against which the query is run is WoodView HSQL, so you will need to add database HSQL JDBC driver (`hsqldb.jar`) to your classpath. The driver is located in the `<EspressReportInstall>/lib` directory.

```
Component doMultiValueParameter(Applet applet) {
 // Do not use EspressManager
 QbChart.setEspressManagerUsed(false);

 // Begin Code : Adding Parameter Info
 SimpleQueryMultiValueInParam inParam =
   new SimpleQueryMultiValueInParam("param", "Please select", true,
     "Categories", "CategoryName", Types.VARCHAR, "Arm Chairs", null);
 SimpleQueryMultiValueInParam[] paramSet = { inParam };
 SimpleQueryFileInfo chartInfo = new SimpleQueryFileInfo(
   "jdbc:hsqldb:woodview",
   "org.hsqldb.jdbcDriver",
   "sa",
   "",
   "select Products.ProductName, Products.UnitsInStock from Categories,
 Products where Categories.CategoryID=Products.CategoryID and
 Categories.CategoryName in(:param) order by Categories.CategoryName,
 Products.ProductName;");
 chartInfo.setInParam(paramSet);

 // End Code : Adding Parameter Info

 // Begin Code : Setting up Column Mapping

 ColInfo colInfo = new ColInfo();
 colInfo.category = 0;
 colInfo.value = 1;

 // End Code : Setting up Column Mapping

 QbChart chart = new QbChart(applet, QbChart.VIEW2D, QbChart.PIE, chartInfo,
 colInfo, null);

 return chart;
}
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/MultiValueParameterizedChart.zip]

When the above code is run as a Java Web Start Application, it will produce a chart (depending on the parameter selected) similar to the one shown below:



Generated Chart

Please note that this is a default chart that is generated without formatting of any kind.

Note that If there are more than one parameters in the IN clause, each of them is considered single-value. For example, in the query:

```
Select * From Products
Where ProductID IN (:param1, :param2, :param3)
```

:param1, :param2, and :param3 are all single-value parameters.

Initializing a multi-value parameter is the same as initializing a single-value parameter. The only difference is in the value selection dialog. While single-value parameters will translate into a drop down box, multi-value parameters will be given a multi-selection list box.

3.C.14. Drill-Down Charts

EspressReport supports different drill-down capabilities. They are:

- Parameter drill-down
- Data drill-down
- Dynamic drill-down

Like parameterized charts, drill-down charts are available only when generating stand-alone charts.

Constructing the different types of drill-down charts are described in the sections below.

3.C.14.1. Parameter Drill-Down Charts

With the help of parameterized queries, parameter drill-down charts can be created. Instead of having a chart with large amounts of data in it, you can show a top level chart showing the minimum data required and then delve deeper on the selected data. For more information on parameter drill-down charts, please refer to the Section 3.6.3 - Parameter Drill-Down

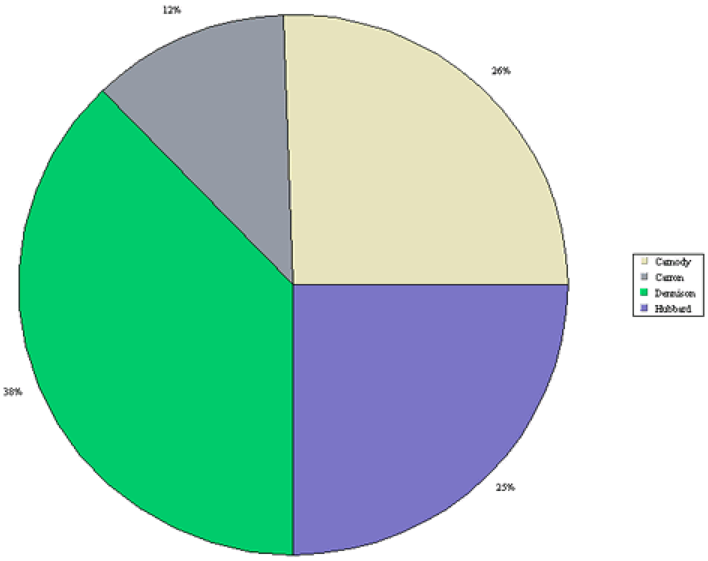
To create a parameter drill-down chart, you need to create the various chart objects (please note that all chart objects, other than the root chart object, will have parameterized queries as their data source) and then specify the order of the drill-down as well as the column/bar/point/line/slice of the chart to attach the next level of the parameter drill-down chart.

Given below is an example of a parameter drill-down chart. The database against which the query is run is Wood-View HSQL, so you will need to add database HSQL JDBC driver (`hsqldb.jar`) to your classpath. The driver is located in the `<EspressReportInstall>/lib` directory.

```
Component doDrillDownChart(Applet applet)  {

   //Do not use EspressManager
   QbChart.setEspressManagerUsed(false);

 // Begin Code : Creating Chart 1  - the Root Chart
 DBInfo rootInfo = new
 DBInfo("jdbc:hsqldb:woodview","org.hsqldb.jdbcDriver",
         "sa",
         "",
         "SELECT Employees.LastName, Count(Orders.OrderID) AS
 Count_Of_OrderID FROM Employees, Order_Details, Orders WHERE
 (Orders.EmployeeID = Employees.EmployeeID) AND (Orders.OrderID =
 Order_Details.OrderID) GROUP BY Employees.LastName");

 ColInfo rootColInfo = new ColInfo(-1, 0, -1, 1);
 QbChart rootChart = new QbChart(applet, QbChart.VIEW2D, QbChart.PIE,
 rootInfo, false, rootColInfo, null);

 SimpleQueryInParam inParam = new SimpleQueryInParam("LastName", "Please
 select", true,
           "Customers", "Company", Types.VARCHAR,
                  "All Unfinished Furniture", null);

 SimpleQueryInParam[] paramSet = {inParam};

 SimpleQueryFileInfo levelChartInfo = new
 SimpleQueryFileInfo("jdbc:hsqldb:woodview",
             "org.hsqldb.jdbcDriver", "sa", "",
             "SELECT Categories.CategoryName, Employees.LastName,
 Sum(Order_Details.Quantity) AS Sum_Of_Quantity FROM Products,
 Employees, Categories, Order_Details, Orders WHERE (Orders.OrderID =
 Order_Details.OrderID) AND (Employees.EmployeeID = Orders.EmployeeID)
 AND (Products.CategoryID = Categories.CategoryID) AND (Products.ProductID
 = Order_Details.ProductID) GROUP BY Categories.CategoryName,
 Employees.LastName HAVING (Employees.LastName =:LastName)");


 levelChartInfo.setInParam(paramSet);

 ColInfo levelOneChartColInfo = new ColInfo(-1, 0, -1, 2);

 try {

  rootChart.createDrillDownChart("TestDrillDownChart", QbChart.VIEW2D,
  QbChart.COL, levelChartInfo, false, null,
  levelOneChartColInfo, null, new int[] {0});
  rootChart.updateDrillDownCharts();

 } catch (Exception ex) { ex.printStackTrace(); }

return rootChart;

}
```

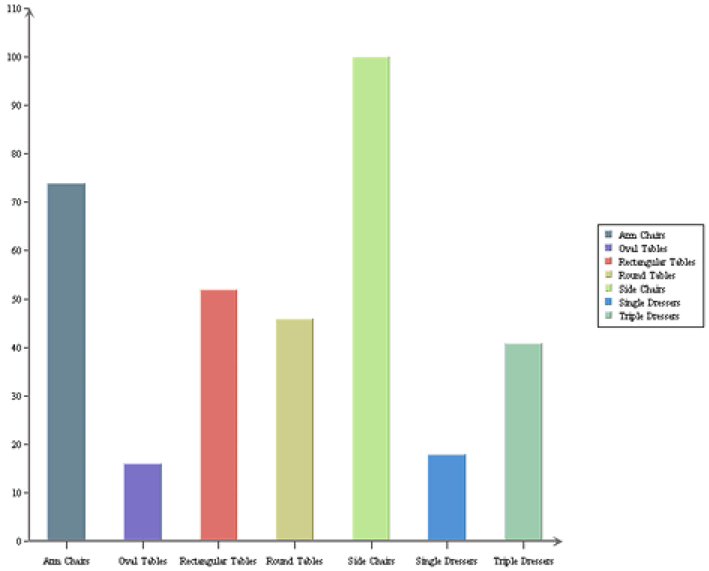
Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ParameterDrillDownChart.zip]

When the above code is run as a Java Web Start Application, it will produce a top-level chart shown below:



Generated top-level chart

Depending on the link clicked, you will see a chart similar to the one shown below:



Generated chart

Please note that this is a default chart that is generated without formatting of any kind.

When you generate parameter drill-down charts without using the EspressManager, you MUST have a sub directory called `drillTemplates` under the working directory of the `.class` file.

3.C.14.2. Data Drill-Down Charts

Parameter drill-down charts allow charts to be generated from different data sources. With data drill-down, the same data source is used throughout the different levels of the charts. The top-level presents a summary of the data. You can click on a data point to bring up another chart showing some detailed information about that particular data point.

Only the column, bar, line, pie, area, overlay, radar, dial, stack column and stack bar supports the data drill-down functionality.

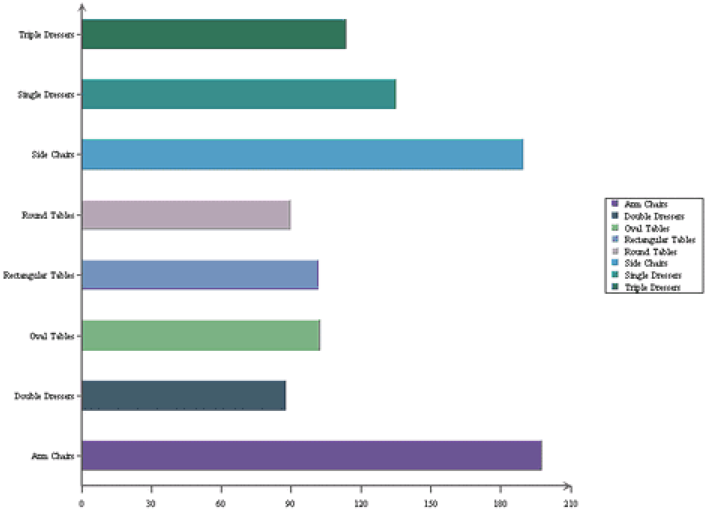
To create a data drill-down chart, you need to create a chart object first before specifying the drill-down properties.

Given below is an example of a data drill-down chart. The database against which the query is run is WoodView HSQL, so you will need to add database HSQL JDBC driver (`hsqldb.jar`) to your classpath. The driver is located in the `<EspressReportInstall>/lib` directory.

```
Component doDataDrillDownChart(Applet applet) {

 //Do not use EspressManager
 QbChart.setEspressManagerUsed(false);

 // Begin Code : Creating Chart 1  - the Root Chart
 DBInfo rootInfo = new DBInfo(
   "jdbc:hsqldb:woodview",
   "org.hsqldb.jdbcDriver",
   "sa",
   "",
   "select Categories.CategoryName, Products.ProductName,
 Products.UnitsInStock from Categories, Products where Categories.CategoryID
 = Products.CategoryID order by Categories.CategoryName;");

 ColInfo rootColInfo = new ColInfo();
 rootColInfo.category = 0;
 rootColInfo.value = 2;
 QbChart chart = new QbChart(applet, QbChart.VIEW2D, QbChart.BAR, rootInfo,
 rootColInfo,
   null);

 // End Code : Creating Chart 1  - the Root Chart

 try {

  // Begin Code : Creating Chart 2  - the sub Chart

  IDrillDown chartDrillDown = chart.gethDrillDown();
  chartDrillDown.setAggregateOperator(IDrillDown.SUM); // Set the
 Aggregation
  chartDrillDown.addDrillDown(QbChart.AREA, 1, -1, -1, true); //
 addDrillDown(chartType, category, series, sumby, is2DChart)
  chartDrillDown.setDrillName("DrillDownChart"); // Set the drill template
 name
 } catch (Exception ex)
 {
  ex.printStackTrace();
 }
 // End Code : Creating Chart 2  - the sub Chart

 return chart;

}
```

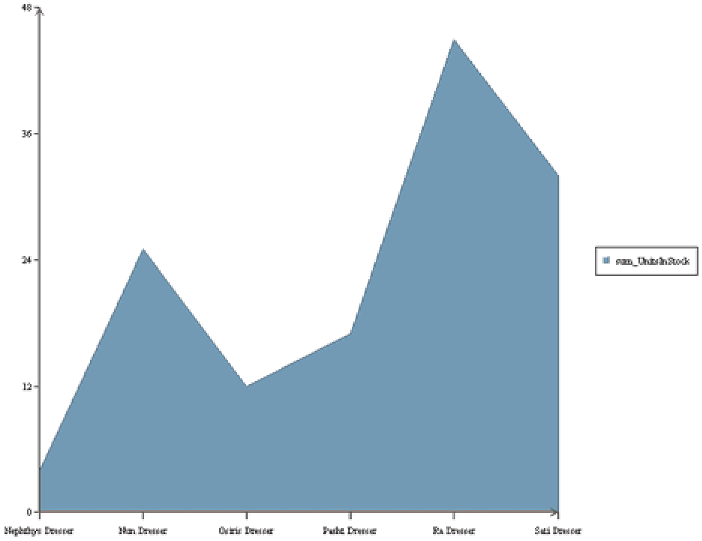
Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/DataDrillDownChart.zip]

When the above code is run as a Java Web Start Application, it will produce a top-level chart shown below:



Generated top-level chart

Depending on the link clicked, you will see a chart similar to the one shown below:



Generated chart

Please note that this is a default chart that is generated without formatting of any kind.

3.C.14.3. Dynamic Data Drill-Down Charts

In data drill-down charts, you have to pre-define the column-to-axis mapping for each drill-down level. Dynamic data drill-down gives you the flexibility to select the column-to-axis mapping for drill-down charts when you are viewing the chart. Only the top-level summary chart needs to be built and the aggregation specified for the drill-down.

Only the column, bar, line, pie, area, overlay, radar, dial, stack column and stack bar supports the dynamic data drill-down functionality.

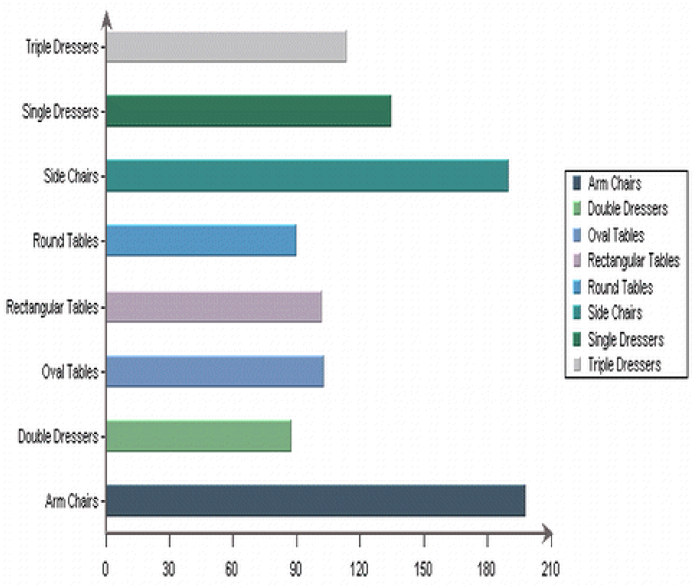
To create a dynamic data drill-down chart, you need to create a chart object first before specifying the drill-down properties.

Given below is an example of a data drill-down chart. The database against which the query is run is WoodView HSQL, so you will need to add database HSQL JDBC driver (`hsqldb.jar`) to your classpath. The driver is located in the `<EspressReportInstall>/lib` directory.

```
Component doDynamicDataDrillDownChart(Applet applet) {

 //Do not use EspressManager
 QbChart.setEspressManagerUsed(false);

 // Begin Code : Creating Chart 1  - the Root Chart

 DBInfo rootInfo = new DBInfo(
   "jdbc:hsqldb:woodview",
   "org.hsqldb.jdbcDriver",
   "sa",
   "",
   "select Categories.CategoryName, Products.ProductName,
 Products.UnitsInStock from Categories, Products where Categories.CategoryID
 = Products.CategoryID order by Categories.CategoryName;");

 ColInfo rootColInfo = new ColInfo();
 rootColInfo.category = 0;
 rootColInfo.value = 2;
 QbChart chart = new QbChart(applet, QbChart.VIEW2D, QbChart.BAR, rootInfo,
 rootColInfo,
   null);

 // End Code : Creating Chart 1  - the Root Chart

 try {

  // Begin Code : Creating Chart 2  - the sub Chart

  IDrillDown chartDrillDown = chart.gethDrillDown();
  chartDrillDown.setAggregateOperator(IDrillDown.SUM); // Set the
 Aggregation
  chartDrillDown.setDynamicDrillDown(true); // Turn on dynamic data drill-
down
  chartDrillDown.setDrillName("DynamicDrillDownChart"); // Set the drill
 template name

 } catch (Exception ex) {
  ex.printStackTrace();
 }

 // End Code : Creating Chart 2  - the sub Chart

 return chart;
}
```

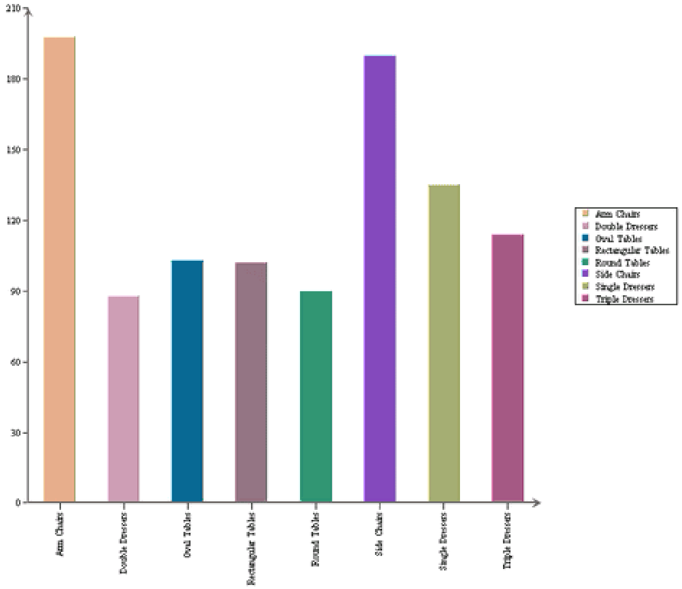
Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/DynamicDataDrillDownChart.zip]

When the above code is run as a Java Web Start Application, it will produce a top-level chart shown below:



Generated chart

Depending on the link clicked, you will see a chart similar to the one shown below:



Generated chart

Please note that this is a default chart that is generated without formatting of any kind.

3.C.15. Adding a Chart to a Report

A chart can be programmatically added to the report in one of three ways :

- Creating a chart template in Designer and adding the template to the report component;
- Creating a chart (that uses report data as its data source) using the API and adding that chart to the report component;
- Creating a `QbChart` object using the API and adding that object to the report component;

3.C.15.1. Adding a Chart Template

To add a chart, you will need to define a `ReportChartObject` object. Within the `ReportChartObject` object, you will have to pass in the location of the template (TPL) file that you are using. You can also specify the dimensions and alignment. If the report is exported to a static format (i.e. DHTML), you can also specify the image type (JPEG, GIF, PNG, etc) of the exported chart. The default alignment is the center of the cell and default image type is JPEG. You can then add the object to the desired location in the report.

Note

You can only use a TPL template file and the chart uses report data (even if the template uses an independent data source) as its data source.

The following example, which can be run as a Java Web Start Application, shows how to add an existing chart template to a report:

```
ReportChartObject chartObject = new ReportChartObject();
String chartLocation0 = new String("AddingChartTemplate.tpl");
chartObject.setText(chartLocation0);
chartObject.setWidth(7.40);
chartObject.setHeight(4.90);
chartObject.setY(0.10);
chartObject.setImageType(QbReport.PNG);
report.getTable().getFooter().addData(chartObject);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/AddingChartTemplate.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/AddingChartTemplate.pdf]

3.C.15.2. Adding a Chart that uses Report Data

To add a chart using this method, you must create the chart object using the `ChartObject` class. A `ChartObject` object contains the chart properties (i.e. the chart mapping, the dimension and chart type). Then a `ReportChartObject` object is instantiated/created, and the newly created `ChartObject` object is passed to it using the `setChart(IChart)` method. The `ReportChartObject` is then finally added to the desired location in the report.

The following example, which can be run as an applet or application, shows how to add a chart, that uses report data as its data source, to a report:

```
ColInfo chartColInfo = new ColInfo();
chartColInfo.category = 0;
chartColInfo.value = 1;

ChartObject chartObj = new ChartObject(report, // QbReport

      ChartObject.VIEW2D, // Dimension
      ChartObject.BAR, // Chart Type
      false, // Do Transpose
      null, // Transpose Columns
      chartColInfo, // Chart Mapping
      "AddingChart.tpl"); // Template

ReportChartObject chartObject = new ReportChartObject();
chartObject.setChart(chartObj);
chartObject.setWidth(7.40);
chartObject.setHeight(4.90);
chartObject.setY(0.10);
chartObject.setImageType(QbReport.PNG);
report.getTable().getFooter().addData(chartObject);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/AddingReportDataChart.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/AddingReportDataChart.pdf]

3.C.15.3. Adding a Chart that uses an Independent Data Source

To add a chart using this method, you must create the chart object using the `QbChart` class. A `QbChart` object contains the chart properties (i.e., the chart mapping, the dimension and chart type). Then a `ReportChartObject` object is instantiated/created, and the newly created `QbChart` object is passed to it using the `setChart(IChart)` method. The `ReportChartObject` is then finally added to the desired location in the report.

This scenario differs from the one in the previous section in that the data for the chart is not coming from the report. The chart data is independent of the report's.

The following example, which can be run as a Java Web Start Application, shows how to add a chart, that uses report data as its data source, to a report:

```
ColInfo chartColInfo = new ColInfo();
chartColInfo.category = 0;
chartColInfo.value = 1;

// Data Source for chart
String chartQuery = "SELECT ProductName, Sum(Order_Details.Quantity) " +
     "FROM Products, Order_Details " +
     "WHERE (Order_Details.ProductID = Products.ProductID) " +
     "GROUP BY Products.ProductName " +
     "ORDER BY Sum(Order_Details.Quantity) DESC";

DBInfo chartDatabaseInfo = new DBInfo("jdbc:hsqldb:database/
woodview", "org.hsqldb.jdbcDriver",
         "sa", "", chartQuery);

QbChart chartObj = new QbChart((Applet)null, QbChart.VIEW2D, QbChart.COL,
 chartDatabaseInfo,
        false, chartColInfo, "AddingChartTemplate.tpl");

// Add ChartObject to report
ReportChartObject chartObject = new ReportChartObject();
chartObject.setChart(chartObj);
chartObject.setWidth(7.40);
chartObject.setHeight(4.90);
chartObject.setReportDataUsed(false);
chartObject.setY(0.10);
chartObject.setImageType(QbReport.PNG);
report.getTable().getFooter().addData(chartObject);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/AddingIndependentDataChart.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/AddingIndependentDataChart.pdf]

Chapter 4. Internationalization

4.1. Internationalizing EspressReport

EspressReport provides many different features that allows you to generate reports for just about any locale and language. Different internationalization features can require different system or setting configurations depending on your specific requirements.

4.1.1. Specifying Locales

EspressReport allows different time zones and locales reports and charts. They are not limited to the locale on the machine where they are created. The locale can be set through API right before the report or chart is run.

4.1.1.1. Locale-Specific Formatting

For data formatting (date and numeric), EspressReport allows you to set locale-specific formatting for report elements. Locale-specific formatting allows data elements to be displayed in the correct format for the particular locale that is being used for the report. Locale-specific options are available in the data formatting dialog for Report Designer (see Section 1.5.7.3 - Data Formatting for Formulas and Column Fields).

The locale for report and charts, as well as the time zone, can be set at run-time through the API. Note that this only effects the date, the time, and the data formatting.

4.1.1.2. Setting Time Zones and Locales Using the API

To set the time zone or locale, you can use the methods in QbReport/QbChart (applies to the entire report/chart) or use `LocaleDateTimeFormat` and `LocaleNumericFormat` objects (applies to a specific object in the report/chart). The following code fragments shows how to do this with two different approaches.

```
report.setLocale(Locale.UK);
report.settime zone(time zone.gettime zone("GMT"));
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/LocaleReport.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/LocaleReport.pdf]

Here, the formats are applied to the entire report. Alternatively, you can apply the format to specific data columns as follows:

```
LocaleNumericFormat currencyFormat =
 LocaleNumericFormat.getCurrencyInstance();
currencyFormat.setLocale(Locale.UK);
int columnIndex = 2; //column 2 is the "Unit Price" column
report.getTable().getColumn(columnIndex).setDataFormat(currencyFormat);

LocaleDateTimeFormat dateFormat = LocaleDateTimeFormat.getDateInstance();
dateFormat.settime zone(time zone.gettime zone("GMT"));
report.getPageFooter().getData(1).setDataFormat(dateFormat);
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/LocaleReportObject.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/LocaleObject.pdf]

4.1.2. Language and Encoding

EspressReport includes a set of features that removes limitations caused by language differences. By utilizing a simple to use interface, it is possible to translate all text, buttons, and menus within EspressReport to a different language. Through some basic configuration, you will be able to import, view, and type characters in a foreign language, as well as save and export reports in that language.

4.1.2.1. EspressReport Language Translation

Internationalization is supported through the use of an `.xml` file, `Language.xml` (located in the `<EspressReport installation directory>`). A GUI is provided to work with the `Language.xml` file and replace the lines of English with those of another language.

The `Language.xml` file has the following structure:

```
<Product name="REPORTDESIGNER" dir="quadbase/reportdesigner/designer">

      <File name="ReportMenubar.java">
            <CODE>File</CODE>
            <TEXT>File</TEXT>
            <CODE>New</CODE>
            <TEXT>New</TEXT>
            ... </File>

</Product>
```

The file has the following tree structure: Product (Directory) → File → text. You can easily search the product, file, or text you want to translate and simply replace the `Text` between `<Text>` and `</Text>` with the translation. EspressReport will replace `<Code>` with `<Text>` in the Java Swing UI and/or in the EspressReport web application. The translation GUI will list all products and each product will list codes that need translation. A same code (for example, `File`) may be listed under several products to maintain the completeness of each product. However, the translation for a code is duplicated across all products. The same code cannot have two different translations for different products.

The `Language.xml` file is located in the install directory. It is recommended that you make a copy (of `Language.xml`) and then use the GUI provided (on the copy) to put in the translation without touching the xml file directly. To utilize the user interface, navigate to the EspressReport root directory and enter the following command:

```
java -classpath "./lib/ReportAPIWithChart.jar;.;"
 quadbase.internationalization.TranslateWizard -file:<fileName> -
enc:<encoding>
```

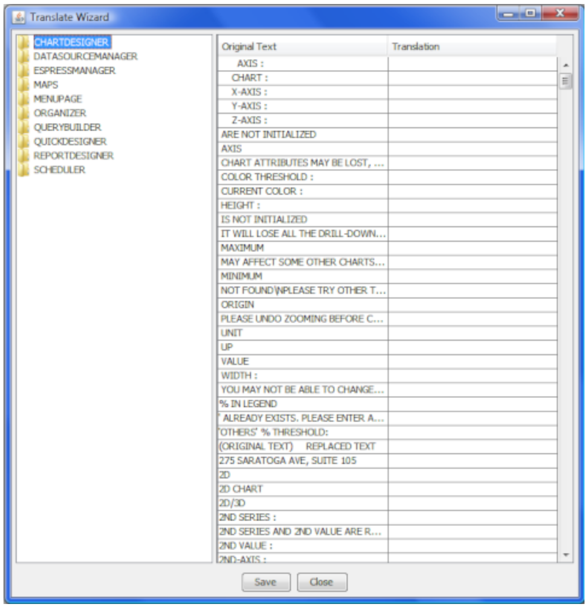
where `<fileName>` represents the xml file that will store the translation and `<encoding>` represents the encoding for the translation. For correct results, the proper encoding must be specified. If the file name and encoding are not provided, the default file name (`Language.xml`) and default encoding (`Cp1252`) will be used.

An example of the above command is given below:

```
java -classpath "./lib/ReportAPIWithChart.jar;.;"
 quadbase.internationalization.TranslateWizard -file:Chinese.xml -enc:gbk
```

Where `gbk` is the encoding used to save `Chinese.xml` file.

When the Translate wizard is started, it appears as below:

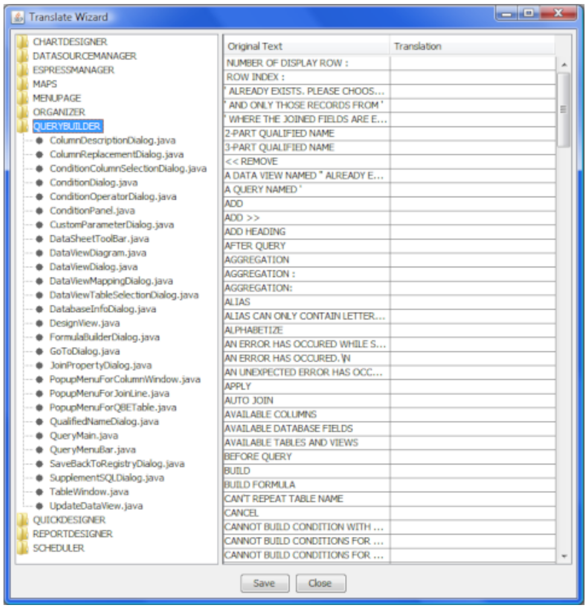


- CHARTDESIGNER
- DATASOURCEMANAGER
- ESPRESSMANAGER
- MAPS
- MENUPAGE
- ORGANIZER
- QUERYBUILDER
- QUICKDESIGNER
- REPORTDESIGNER
- SCHEDULER

Original Text	Translation
AXIS :	
CHART :	
X-AXIS :	
Y-AXIS :	
Z-AXIS :	
ARE NOT INITIALIZED	
AXIS	
CHART ATTRIBUTES MAY BE LOST, ...	
COLOR THRESHOLD :	
CURRENT COLOR :	
HEIGHT :	
IS NOT INITIALIZED	
IT WILL LOSE ALL THE DRILL-DOWN...	
MAXIMUM	
MAY AFFECT SOME OTHER CHARTS...	
MINIMUM	
NOT FOUND\NPLEASE TRY OTHER T...	
ORIGIN	
PLEASE UNDO ZOOMING BEFORE C...	
UNIT	
UP	
VALUE	
WIDTH :	
YOU MAY NOT BE ABLE TO CHANGE...	
% IN LEGEND	
' ALREADY EXISTS. PLEASE ENTER A...	
'OTHERS' % THRESHOLD:	
(ORIGINAL TEXT) REPLACED TEXT	
275 SARATOGA AVE, SUITE 105	
2D	
2D CHART	
2D/3D	
2ND SERIES :	
2ND SERIES AND 2ND VALUE ARE R...	
2ND VALUE :	
2ND-AXIS :	

Translate Wizard

You can see the various codes (available for translation) by selecting appropriate Product. Products are shown in a tree on the left that contains all the files available for translation. For example, on the image below you can see expanded *QUERYBUILDER* product. You can either translate entire product by selecting the product node or just select a file that you want to translate.



- CHARTDESIGNER
- DATASOURCEMANAGER
- ESPRESSMANAGER
- MAPS
- MENUPAGE
- ORGANIZER
- QUERYBUILDER
  - ColumnDescriptionDialog.java
  - ColumnReplacementDialog.java
  - ConditionColumnSelectionDialog.java
  - ConditionDialog.java
  - ConditionOperatorDialog.java
  - ConditionPanel.java
  - CustomParameterDialog.java
  - DataSheetToolBar.java
  - DataViewDiagram.java
  - DataViewDialog.java
  - DataViewMappingDialog.java
  - DataViewTableSelectionDialog.java
  - DatabaseInfoDialog.java
  - DesignView.java
  - FormulaBuilderDialog.java
  - GoToDialog.java
  - JoinPropertyDialog.java
  - PopupMenuForColumnWindow.java
  - PopupMenuForJoinLine.java
  - PopupMenuForQBETable.java
  - QualifiedNameDialog.java
  - QueryMain.java
  - QueryMenuBar.java
  - SaveBackToRegistryDialog.java
  - SupplementSQLDialog.java
  - TableWindow.java
  - UpdateDataView.java
- QUICKDESIGNER
- REPORTDESIGNER
- SCHEDULER

Original Text	Translation
NUMBER OF DISPLAY ROW :	
ROW INDEX :	
' ALREADY EXISTS. PLEASE CHOOS...	
' AND ONLY THOSE RECORDS FROM '	
' WHERE THE JOINED FIELDS ARE E...	
2-PART QUALIFIED NAME	
3-PART QUALIFIED NAME	
<< REMOVE	
A DATA VIEW NAMED " ALREADY E...	
A QUERY NAMED '	
ADD	
ADD >>	
ADD HEADING	
AFTER QUERY	
AGGREGATION	
AGGREGATION :	
AGGREGATION:	
ALIAS	
ALIAS CAN ONLY CONTAIN LETTER...	
ALPHABETIZE	
AN ERROR HAS OCCURED WHILE S...	
AN ERROR HAS OCCURED. \N	
AN UNEXPECTED ERROR HAS OCC...	
APPLY	
AUTO JOIN	
AVAILABLE COLUMNS	
AVAILABLE DATABASE FIELDS	
AVAILABLE TABLES AND VIEWS	
BEFORE QUERY	
BUILD	
BUILD FORMULA	
CAN'T REPEAT TABLE NAME	
CANCEL	
CANNOT BUILD CONDITION WITH ...	
CANNOT BUILD CONDITIONS FOR ...	
CANNOT BUILD CONDITIONS FOR ...	

Insert Translation

The English texts are listed on the left and you can enter the translation on the right. The translation can be entered by clicking on the cell and typing in the text. The translation can be saved (to the file specified by the command that started the wizard) by clicking on the *SAVE* button. You can navigate to the previous screen by clicking on the *PREVIOUS* button. Note that if a translation for a code has already been set, it will appear for every instance of the code.

Note that more options exist in the Translate Wizard than is required by EspressReport. For example, the ORGANIZER product does not exist in EspressReport and hence any changes made in this product will not show up.

You can use `Language.xml` (or the copy you modified) in EspressReport by adding the `-file` and `-enc` arguments to the java application or java JNLP (More about Applets in JNLP: Section 1.2.7 - Run Applets in WebStart with JNLP file) that starts EspressManager, ReportDesigner and/or Scheduler.

4.1.2.1.1. Upgrading Language File

When the user upgrades to a newer version, the `Language.xml` file needs to be upgraded as well. The language upgrade program will copy over the translations from the previously customized `Language.xml` file and append the additional entries in the new version. To use the language upgrade program, navigate to the EspressReport directory from a console window and use the following command:

```
java -classpath ".;.\lib\ReportAPIWithChart.jar"
 quadbase.internationalization.UpgradeLanguageXMLFile -from:oldfile -
to:newfile -enc:encoding
```

It may be necessary to replace the semicolon with colon and backslash with slash on non-Window environments.

Oldfile refers to the customized `Language.xml` from a previous version of EspressReport, newfile refers to the `Language.xml` included in the upgraded EspressReport installation, and the `enc` is the language encoding used in the translation. After running the program, the resulting file will be named `Language.xml` and you can open it with Translate Wizard to further translate any new entries.

4.1.2.2. Displaying Foreign Characters

Foreign characters can be easily displayed in Designer and various Viewers. To display foreign characters in a report/chart, you will need to have fonts for that language installed in your system. Then, in the report/chart, you can set the font for the object that contains foreign characters to the appropriate system font.

Another option is to modify the font.properties file in the JVM so that foreign characters are supported in the default JVM fonts. For Sun JVMs, the font.properties file is located under the `jre/lib` directory. The different language files have names like `font.properties.ru` for Russian, `font.properties.zh` for Chinese, etc. To change the language settings for the JVM, rename the current `font.properties` file to back it up and change the name of the desired language file to `font.properties`. With the language settings changed in the JVM, the default fonts in EspressReport (Dialog, Serif, Monospaced, etc.) will display with foreign characters.

4.1.2.3. Entering Foreign Characters

In order to enter foreign characters into any EspressReport interface (for example, ReportDesigner), the following changes to the system settings are required:

- The *default locale* of your system must be set to the region for the language you want to use.
- The *input locale* for the system must also be set to the region for the language you want to use

Note that for Windows the settings can be accessed through *Regional Options* in the *Control Panel*.

In addition, the font settings in the JVM must be adjusted to the desired language following the instructions discussed in Section 4.1.2.2 - Displaying Foreign Characters.

Once these settings have been applied, foreign characters can be entered in labels, queries, formulas, and parameters in reports/charts.

4.1.2.4. XML Encoding

By default, EspressReport use UTF-8 character set for encoding when writing to XML. This includes data registry files, XML report templates, XML exports, and XML representations of global format information and font mapping. Please note that for most of users it is not necessary to change character set encoding, because UTF-8 fully supports all languages.

This encoding can be changed for other languages by adding a run-time parameter to ReportDesigner and EspressManager. For ReportDesigner, you can do this by modifying the `ReportDesigner.bat/sh` file, or modifying the applet page used to launch the ReportDesigner.

To change the XML encoding, add the following argument to the `<EspressReportInstallDir>\ReportDesigner.bat` or `.sh` file: `-xmlEncoding:Encoding`. For example `-xmlEncoding:ISO-2022-JP` would set the encoding for the Japanese character set.

If you are running ReportDesigner through an applet, you will need to add the following parameter to the applet page: `<PARAM NAME="xml_encoding" VALUE="ISO-2022-JP">`.

This parameter also needs to be set for the server. For more information about server settings, please see Section 1.2.4 - Starting EspressManager.

4.1.2.5. Exporting With Foreign Characters

Most of the EspressReport export formats will be automatically generated with the UTF-8 character encoding. In order to view the exported reports/charts, the client will need to have the appropriate system fonts installed. The only exception to this is the PDF export. PDF format does not depend on client fonts for viewing. Therefore, foreign language fonts will need to be embedded into the PDF document in order for characters to display properly. To do this, you will need to use the PDF Font Mapping feature. The Font Mapping can be set in Designer and in API.

If you wish to export the report/chart to a locale different from the one that is set on the machine, you can specify a different encoding before doing the export. This is done using the following method in QbReport:

```
public void setExportEncoding(String encoding);
```

where `encoding` is a character encoding supported by the Java 2 platform. For example, the character encoding for English (Windows Latin-1) is `Cp1252`. Please note that you have to use the canonical name used by the `java.io` and `java.lang` APIs.

You can also specify the character set used in the DHTML exports using the following method in QbReport:

```
public void setHTMLCharSet(String charset);
```

where `charset` is a valid character set. For example, a character set for English used in DHTML documents is `UTF-8`.

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/ExportForeign.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/ExportForeign.html]

4.1.2.6. Specifying Encoding for Text Data Files

By default, EspressReport use system encoding for any text data files. The encoding for input text data files can be set via API using the following block of code:

```
report.getInputData().setDataFile("sample.dat", false, "ASCII");
```

Full Source Code [https://data.quadbase.com/Docs71/help/manual/code/src/DataFileEncoding.zip]

Exported Results [https://data.quadbase.com/Docs71/help/manual/code/export/DataFileEncoding.pdf]

The above block use the following method found in `quadbase.reportdesigner.util.IInputData`:

```
public void setDataFile(String fileName, boolean isDataSorted, String
 encoding);
```