EspressChart®

Version 7.2

EspressChart 7.2 User's Guide

Quadbase

Quadbase Systems Inc.

EspressChart User's guide

Welcome to the EspressChart Online User's Guide.

This documentation is available in two versions - PDF and HTML. In both versions of this document, all chapter references (for example: Chapter 10 - EspressChart Chart API) are also active links. Clicking on them will open the relevant chapter.

EspressChart Users guide consists of several elements:

QuickStart Guide — Quick overview of EspressChart and a simple guide for beginners. This chapter will guide you through some basic steps like setting up a data source and creating your first chart in EspressChart.

User's Guide — Complete EspressChart documentation.

API Documentation — Chart API documentation. Consists of several elements.

- **API documentation chapter** — Chapter 10 - EspressChart Chart API - The main API documentation chapter with working code examples.
- **Javadoc** — Complete JavaDocs for Chart API. Located in the `<EC_Install_Directory>/help/javadoc/apidocs` directory. Can be also accessed through the following link: Chart API JavaDoc [https://data.quadbase.com/Docs72/ec/help/apidocs/index.html]
- **Appendixes** — More specific API code examples (chart-type specific settings etc...)

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.

QuickStart

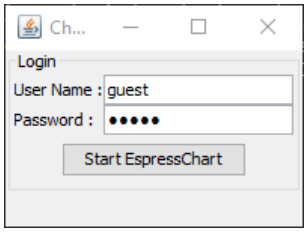
This guide will take you through the basics of setting up a data source and creating charts using Chart Designer. This guide assumes you are running EC locally on a Windows NT/2000/XP machine or newer.

If you're not running on a Windows machine you can still use this guide; however, you won't be able to use the sample Access database. There is a text file that you can use instead to create the same chart.

Q.1. Start Chart Designer

To start Chart Designer you will first need to start EspressManager. To start EspressManager, execute the `EspressManager.bat` or `EspressManager.sh` file that is in the root directory of your installation. For Windows installations, you can also start EspressManager from the Start menu if you selected to create shortcuts when installing the software.

Once EspressManager is running, you can launch Chart Designer by executing the Designer .bat or .sh file in the root directory of your installation. For Windows installations, you can also start Chart Designer from the Start menu if you selected to create shortcuts when installing the software. When Chart Designer is started, a login window will appear prompting you for a user name and a password. Enter **`guest`** as the user name, and leave the password field blank.



Click Start EspressChart and Chart Designer will open as a new window.

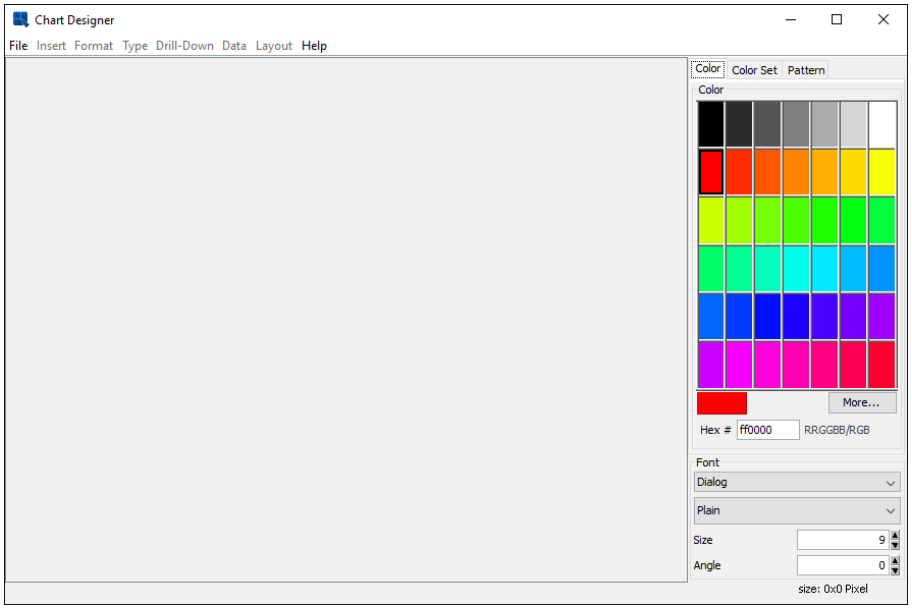
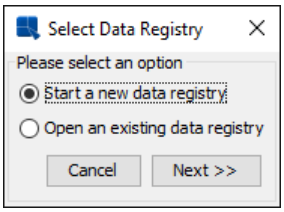


Chart Designer main window

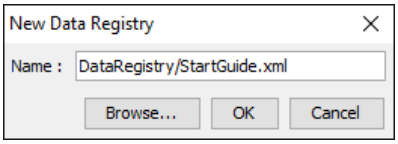
Q.2. Start a new chart & setup a data source registry

To start a new chart, select File → New. A dialog will appear asking if you would like to create a new data source registry, or use an existing one. A data registry is an XML file that stores database connections, queries, and file locations for text and XML sources. Select *Start a new data registry* and click *Next*.

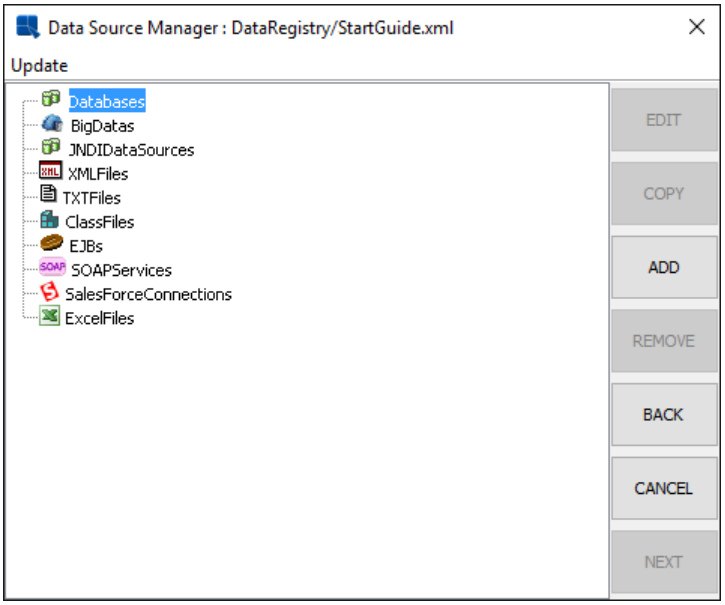


Select Data Registry dialog

You will then be prompted to enter a name for your data registry. You can name it anything you want and an XML file of the same name will be created in the DataRegistry directory. After you have specified a name, the Data Source Manager will open. Since this is a new registry there are no nodes under the data source types in the window.



New Data Registry dialog

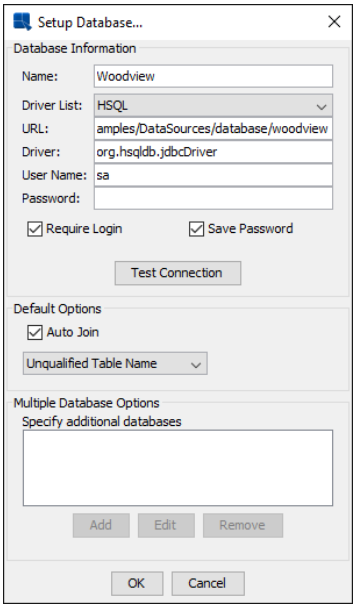


Data Source Manager

Q.3. Set up Data Sources in the Registry

The first data source to set up is a database. EspressChart comes with two sample databases. One is HSQL which is a pure Java application database. The other is a MS Access Database. Both contain the same data.

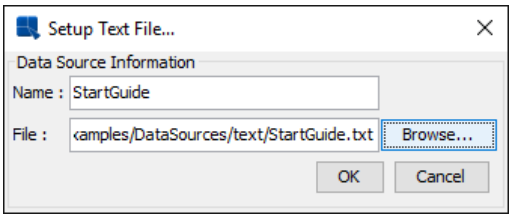
Start Chart Designer and open your data registry again. From the Data Source Manager, click on the *Databases* note in the left-hand frame, and click the *Add* button. A dialog will then appear prompting you to enter the connection information for the new database. Enter **`Woodview`** as the name of the Database, 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 user name and leave the password blank.



Setup Database dialog

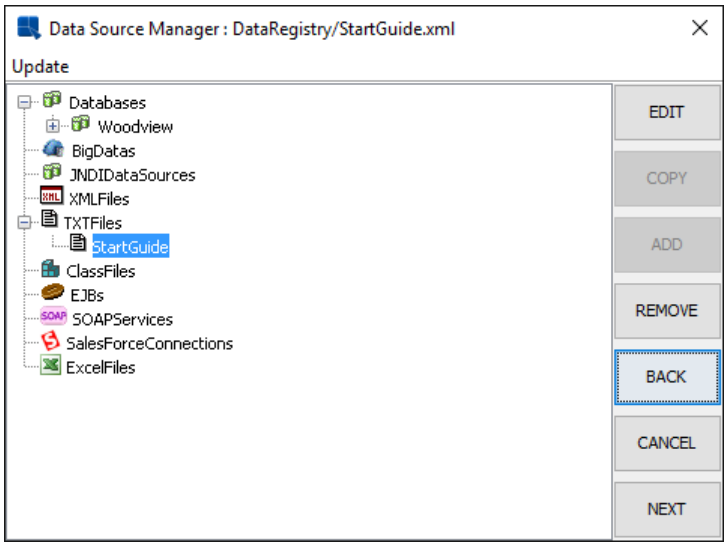
Leave the auto-join and table name properties alone and click the *Test Connection* button to make sure you've entered the information correctly. Then click *OK* to bring back up the data source manager window, where there will be a new node under *Databases* for Woodview.

To add a text data source, click on the *TXTFiles* node in the left-hand frame of the Data Source Manager and click *Add*. A dialog will appear prompting you to specify a display name for the data source and the location of the text file. Enter any name you would like and then press the *Browse* button. Browse to `help/examples/DataSources/text/` and select `StartGuide.txt`.



Data Source Manager

Clicking *OK* will bring back up the Data Source Manager where you will see a new node under *TXTFiles* for the text data source that you have just entered.

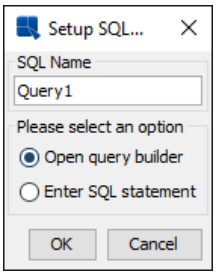


Data Source Manager

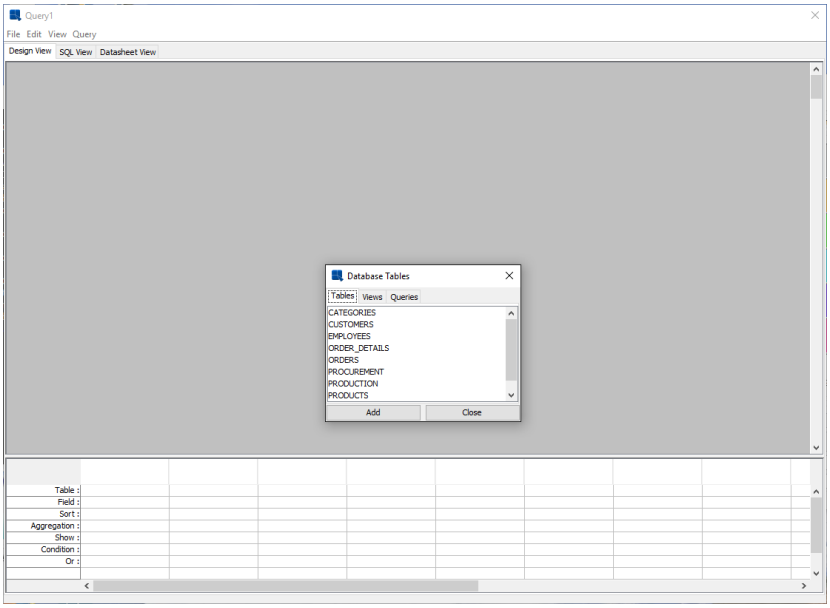
There is an additional sample text file under `help/examples/DataSources/text/` called `Sample.dat`. This expanded sample file contains data for each data type. You can use this file to explore the different chart types and options.

Q.4. Build a query

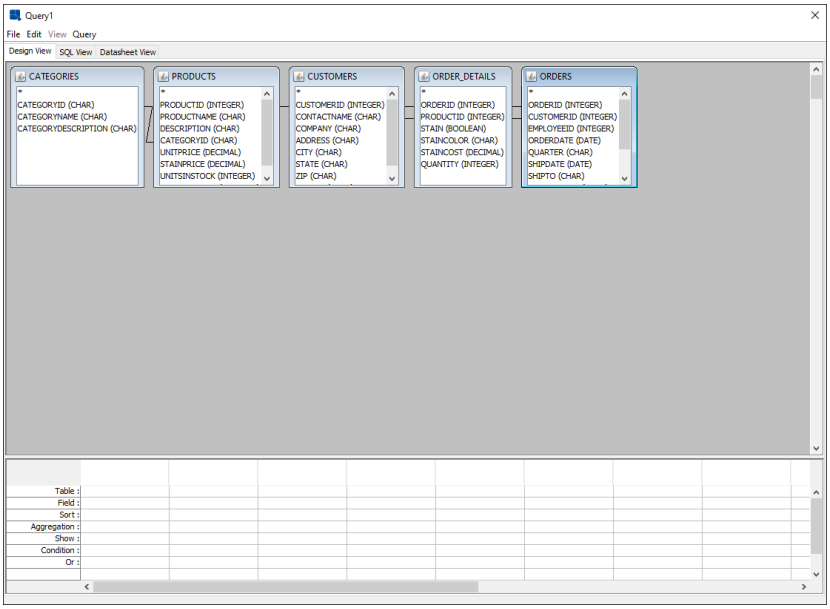
To create a new query, click to expand the *Woodview* node in the left-hand frame of the Data Source Manager. Two sub-nodes will appear, one called *Queries* and one called *Data Views*. Select the *Queries* node and click *Add*. A dialog will appear prompting you to specify a name for the query, and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



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

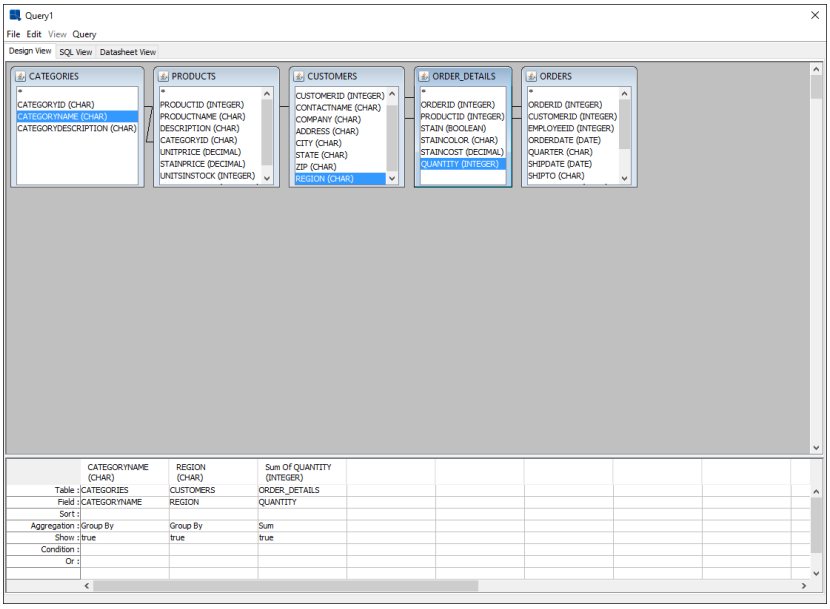


From the *Tables* tab add the Categories, Products, Customers, Order Details, and Orders tables to the query by selecting each and clicking the *Add* button. As the tables are added they will appear in the top half of the query builder window. The tables will be auto-joined as indicated by the black lines connecting them. Once you have added the tables, click the *Close* button on the tables window.

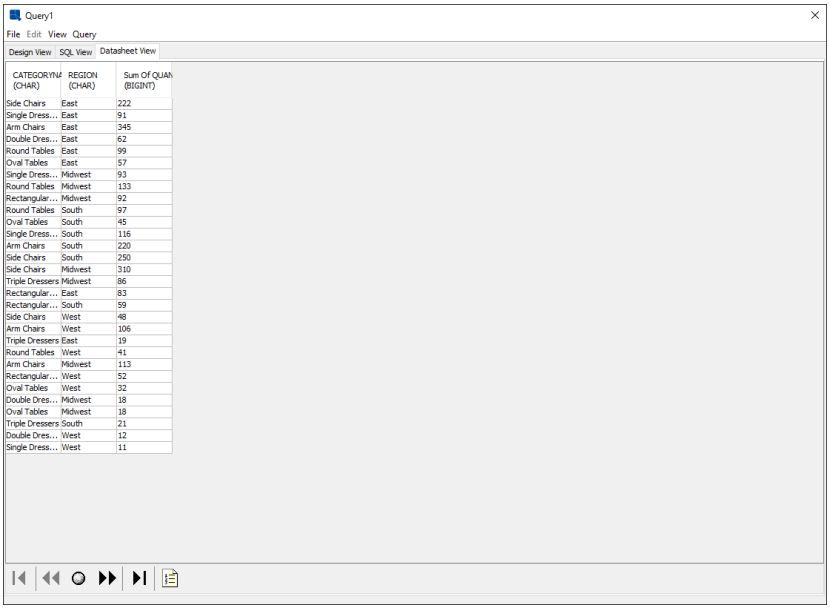


To add fields to the query you can either double-click the field from within the table window, or you can double-click on the *Table* and *Field* fields in the lower-half or QBE (query by example) portions to make selections from a drop-down menu. Using either method, add the following fields to the query: CategoryName from the Categories table, Region from the Customers table, and Quantity from the Order_Details table.

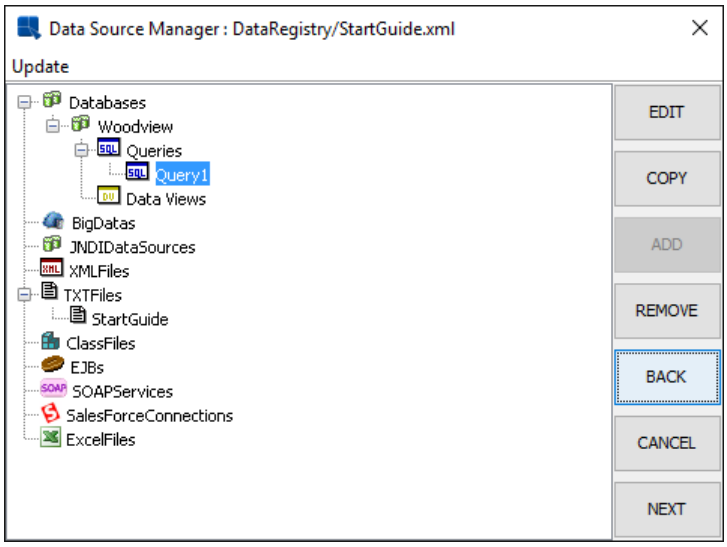
Once you have added the fields, double-click in the *Aggregation* field under the CategoryName column. Select *Group by* as the aggregation type. When you click elsewhere in the QBE frame all the other columns will default to *Group by* as the aggregation. Double-click the *Aggregation* field under the Quantity column and change the aggregation to *Sum*.



Once you have entered the fields, you can preview the results of the query by clicking on the *Datasheet View* tab. The query will run and you will see the total sales volume broken down by product category and sales region.

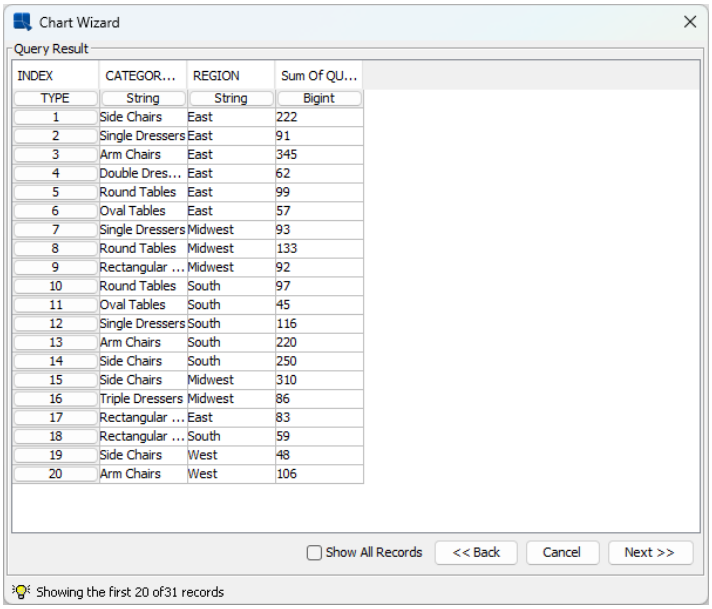


Once you have finished building the query, select File → Done. The Query Builder will close and the Data Source Manager will come back up. There is now a node under *Queries* for the newly created query.

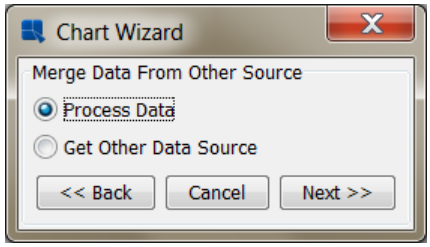


Q.5. Build a chart

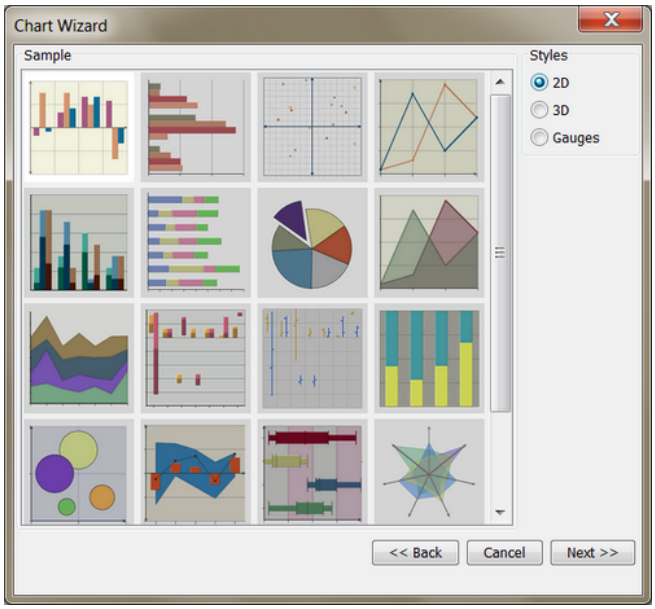
To build a chart, you must first select the data source that you would like to use. Select either the query that you created or the StartGuide text file and click the *Next* button. A new window will open showing the first twenty records of data.



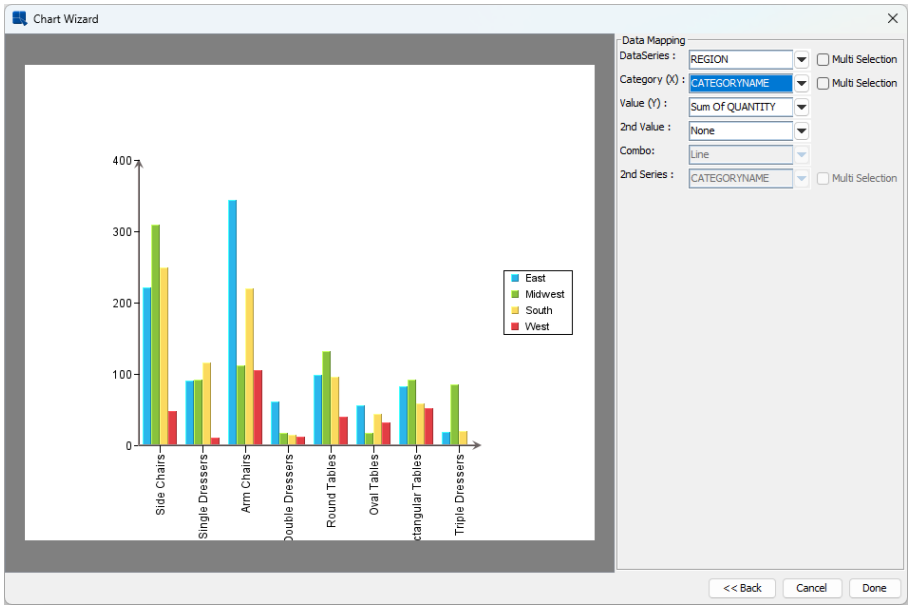
From this dialog click *Next* again. A dialog will appear asking you to process data or to get another data source. Select *Process Data* and click *Next*.



The next screen in the chart wizard allows you to select the chart type that you would like to use. You can toggle between two-dimensional and three-dimensional chart types using the radio buttons. Select a two-dimensional column chart as the type that you would like to use and click *Next*.



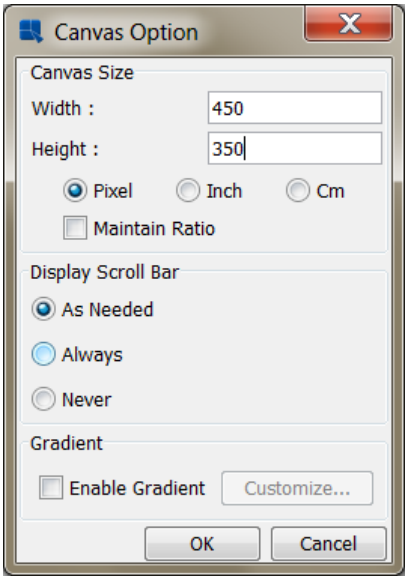
The next screen allows you to set data mapping for the chart. Data mapping is the process by which the columns of data from your data source are mapped to the elements of the chart. By default the first column from left to right (CategoryName) will be mapped as the data series, the second (Region) column as the categories, and the third (Quantity) column as Value. To modify the mapping, select Region as the data series and CategoryName as the categories. Leave the Value option the same.



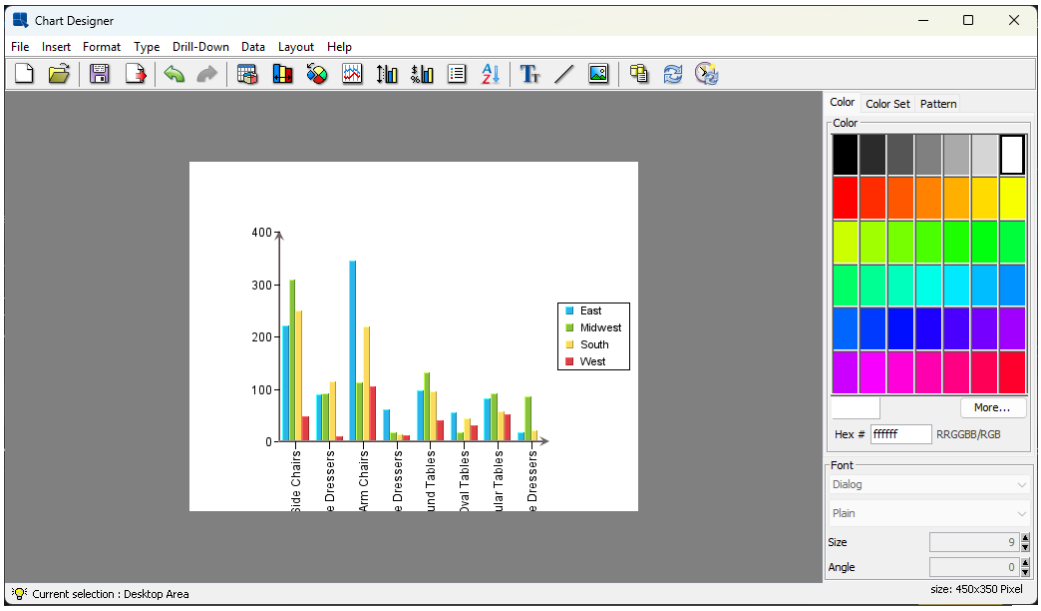
Once you have finished setting the mapping options, click *Done* and you will go to the Chart Designer window where you can customize the chart.

Q.6. Customize chart properties

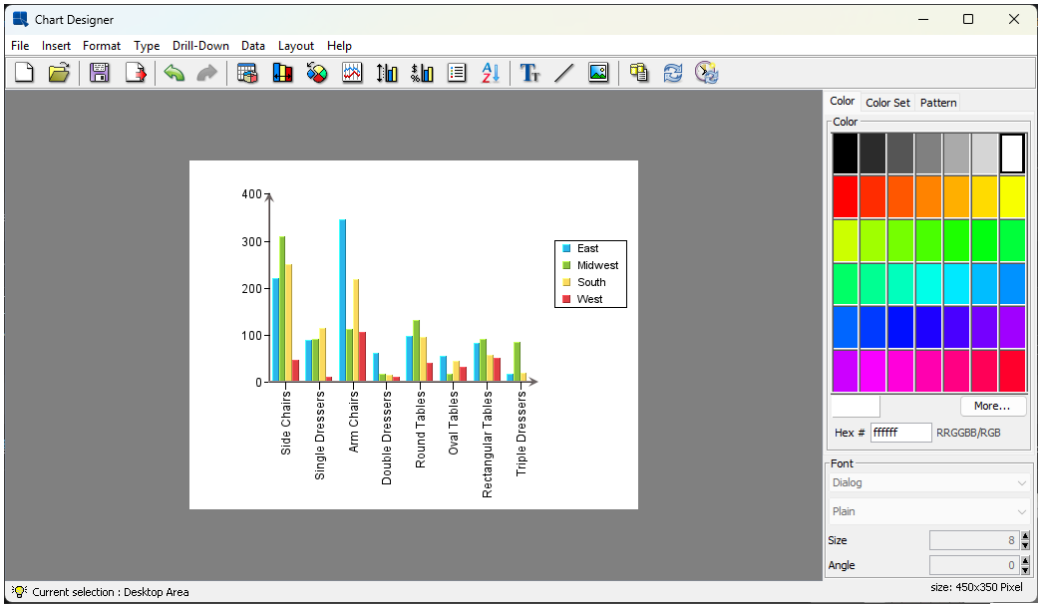
The default canvas size for charts is 600 x 500 pixels. To adjust the size of the chart canvas, select *Canvas* from the Format menu. This will bring up a dialog allowing you to resize the chart canvas in either pixels, inches, or centimeters.



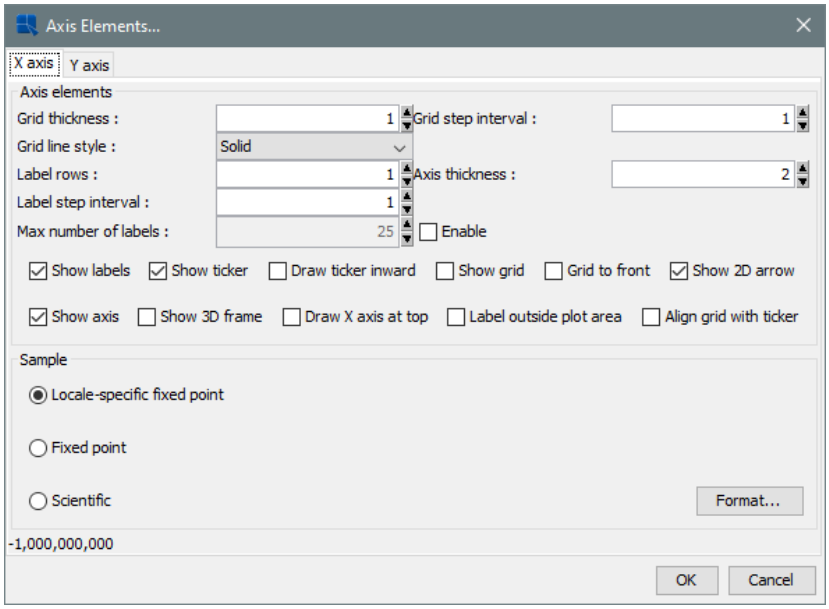
Uncheck the *Maintain Ratio* button, change the canvas dimensions to 450 x 350 pixels and click the *Ok* button. The canvas will now resize in the designer.



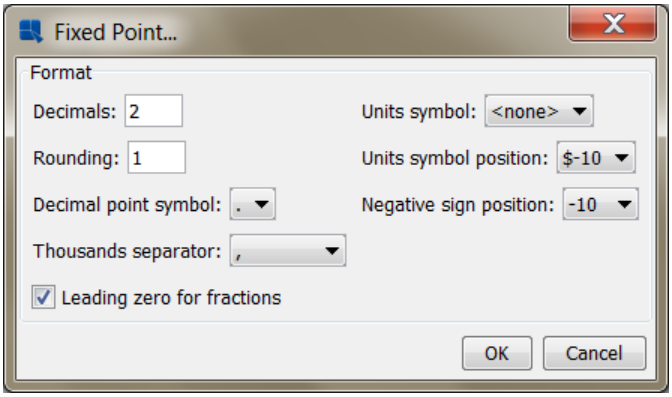
Now that the chart canvas has been shrunk, the chart plot will now appear small and portions of the X-Axis labels may be truncated. This can be adjusted by resizing the chart. You can click and drag on the chart plot to move it and then right-click and drag to resize it. You can also click and drag to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



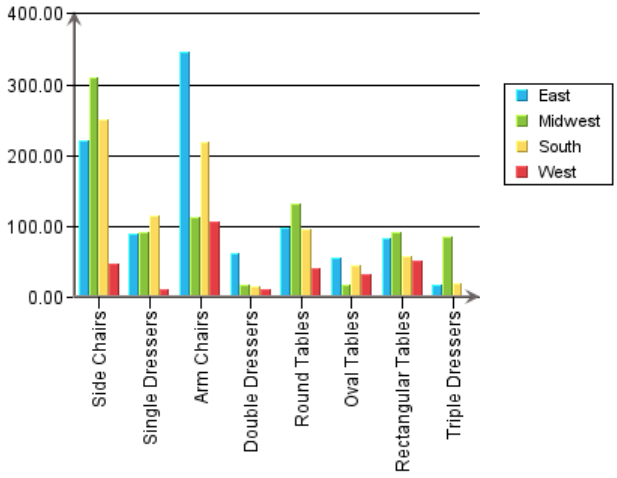
Next you can modify the format of the axis labels. To do this click the Axis Elements button on the toolbar:  . This will bring up a tabbed dialog allowing you to set different options for each chart axis. Click on the *Y Axis* tab to bring up options for the value axis.



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 the numeric data. Select the number of decimals as 2 and click *OK*.

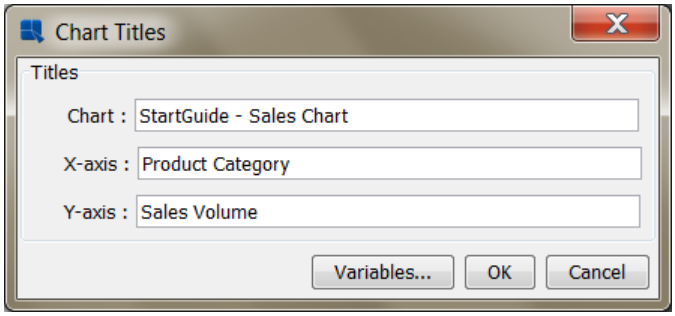


Click *OK* again to dismiss the axis elements dialog and you will see the specified changes reflected in the chart.

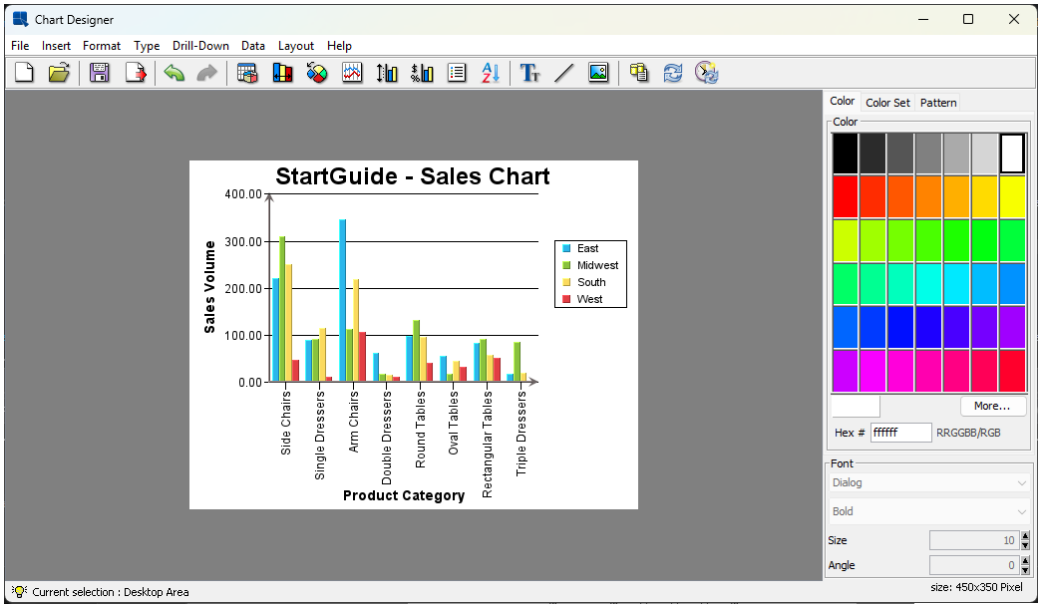


To modify the colors of the chart elements, you can click the *Randomize Color* button. This will cycle through various color combinations for the data elements. You can also change the color for any element by clicking to select it (the hint at the lower left-hand corner of the design window will indicate the currently selected element) and then picking a new color from the color panel.

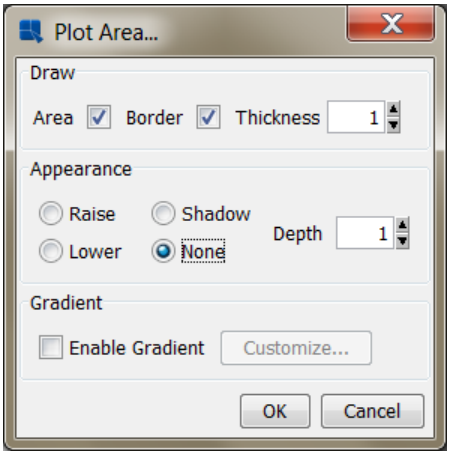
Next, you can add titles to the chart. To do this, select Insert → Titles. This will bring up a dialog allowing you to enter titles for the chart as well as each of the axes.



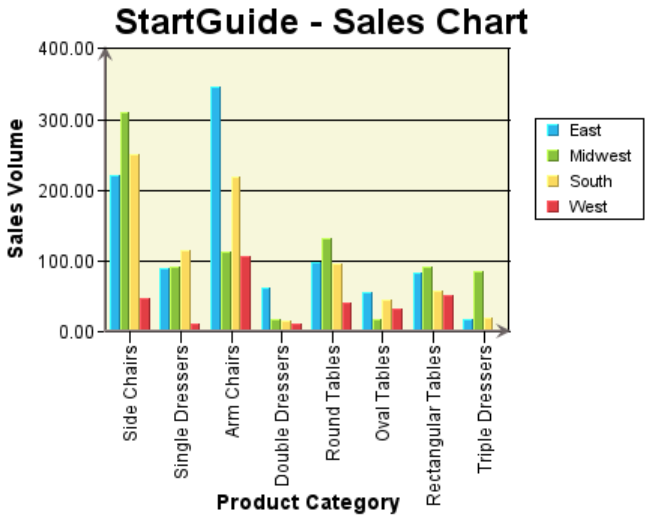
Enter any titles that you would like for the various elements and click *OK*. The titles will then be added to the chart. Titles are placed automatically, but you can manually adjust their positions by clicking and dragging the text on the chart canvas.



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

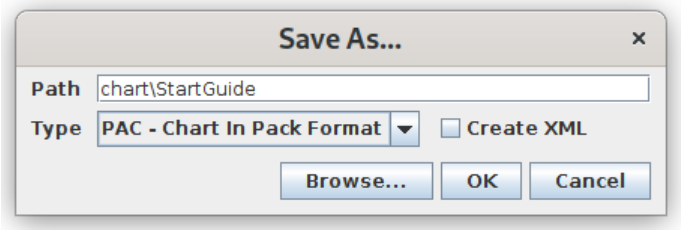


Select to draw both the plot area and the border with a thickness of 1. Specify *None* for the appearance. Once you have specified the options, click *OK* and the plot area for the chart will be modified. You can change the background color of the plot area, by clicking to select it, and then modifying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



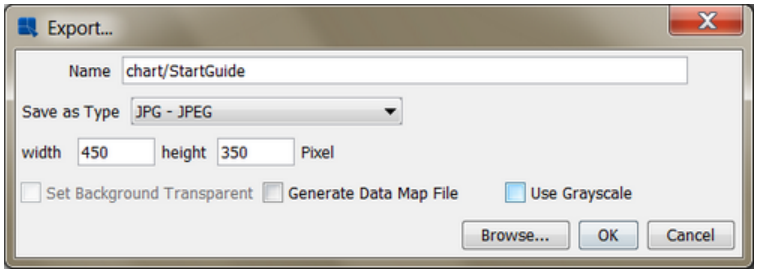
Q.7. Save and export the chart

Once you have finished customizing the chart, you can save the chart definition by selecting File → Save As. This will bring up a dialog prompting you to specify the file name as well as several other save options.

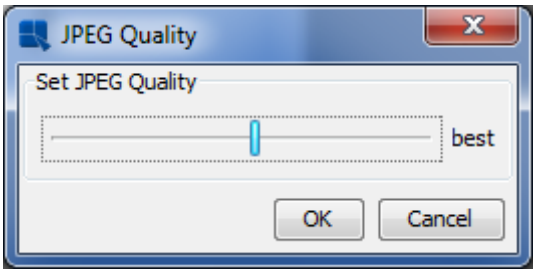


Select to save the chart in CHT format and specify a file name for the chart. This will save the chart with its data in the root directory. You can specify a different location if you like.

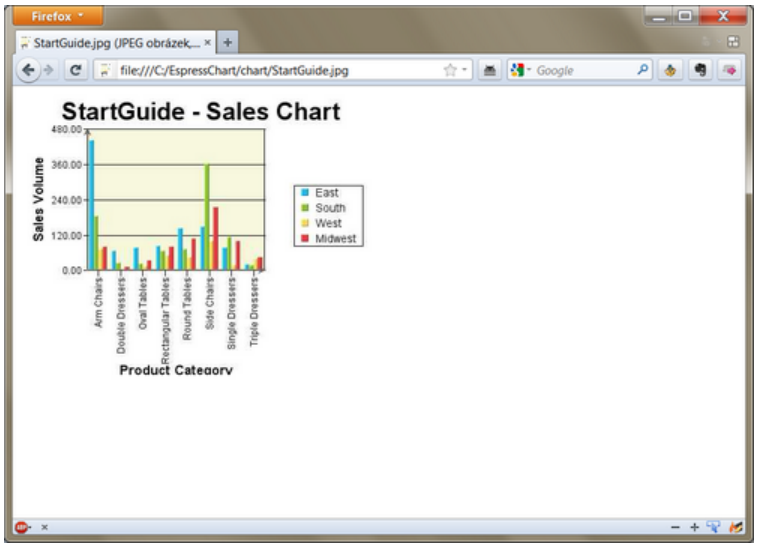
Once you have saved the chart, you can export it by clicking the Export button on the toolbar:  . This will bring up a dialog prompting you to specify export options for the chart.



Specify a name for the generated image file and select JPEG as the export format. Click *OK* and you will be prompted to specify the image quality for the generated image.



Specify the maximum quality and click the *OK* button. A JPEG file will be written in the root directory of the installation. You can preview the image by opening the generated file in your Web browser.



Now you've covered some of the basic functionality of the Chart Designer. Please see the User's Guide for more detail on any of the features discussed in this guide.

Chapter 1. Overview/Introduction

Welcome to EspressChart. EspressChart is a powerful set of tools for creating and deploying dynamic charts. EspressChart makes it easy to create polished, compelling charts and to render them in virtually any environment, platform, or configuration. EspressChart includes a powerful design interface with extensive data connection tools that allows users to query a data source and to plot a chart in an easy-to-use point and click environment. EspressChart also includes a robust, object-oriented API that allow users to easily incorporate dynamic charts and the Chart Designer into applications, JSPs, and servlets. EspressChart has been certified 100% Pure Java™ and uses no native libraries, allowing charts to be rendered on virtually any platform. EspressChart can directly connect to JDBC data sources (including Excel spreadsheets) and can also draw data from text, XML files, or even EJBs. For even more flexibility, users can pass data sets to a chart directly as an argument or array. At run-time, charts can be rendered in a number of popular image formats including GIF, JPEG, PNG, SVG, & SWF, or displayed in applets for full interactivity.

1.1. Using this Guide

The EspressChart User's Guide is roughly broken into three sections. The first section covers background information including installation, architecture, setup, and configuration of all the EspressChart components. The second section covers charting basics, using the chart designer, and many of the features found in EspressChart. It is recommended that you look over this section to gain some familiarity with EspressChart's basic features and terminology even if you are planning to develop charts completely programmatically. The third section deals with programming and deploying charts. It covers the Chart Viewer applets, the Chart API, integration with servlets/JSPs, and deployment options/configurations. The API section in this guide contains an overview, some modes of usage, and an explanation of how to use some of EspressChart's features in the API. There are also two additional resources included in your installation:

API How To This provides a more extensive API tutorial. It details specifically, with sample code, how to do many of the basic chart creation/manipulation functions in the API. The guide can be found in your EspressChart installation in the following location: `<EspressChartInstallDir>/help/api_HowTo/index.html`

API JavaDoc The complete API JavaDocs are included in your EspressChart installation in the following location: help/apidocs/index.html

For sample charts, and API code (including servlet & JSP examples) look in the `<EspressChartInstallDir>/help/examples` directory.

1.2. EspressChart Components

EspressChart is composed of four main components:

Chart Designer The Chart Designer is an interactive application that allows users to create and edit charts in a visual point and click environment. It can be run locally as a application or loaded as a applet(More in: Section 2.6 - Run Applets in WebStart with JNLP file). Most chart properties can be easily set and modified in the designer without writing code. Finished charts can be exported directly to a number of static formats for viewing or printing. Charts can also be saved as template (.tpl) files. A template stores the chart attributes (colors, size, etc), but none of the chart data. Every time a template is loaded, it will retrieve the latest data from the source, allowing charts to be updated without programming. Templates can also be applied to other charts or chart objects (API), allowing formatting information to be carried from one chart to another.

Chart Viewer Chart Viewer is a applet(More in: Section 2.6 - Run Applets in WebStart with JNLP file) that allows charts to be viewed remotely. It enables viewers to rotate, resize, zoom, and pan a chart. Viewers can click on individual data points to either recover the data associated with the data point or to link to another chart.

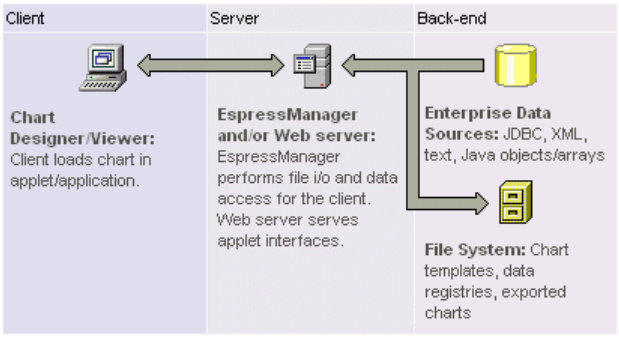
Chart API Chart API is an easy-to-use application programming interface that enables you to create, customize, and deploy charts within your application (and JNLPMore about Applets in

JNLP: Section 2.6 - Run Applets in WebStart with JNLP file), on either the server-side or the client-side. It provides a Java package to a set of advanced two and three-dimensional charting library functions. Charts can be constructed with as little as one line of code.

EspressManager EspressManager provides a “back end” to EspressChart. It provides the functions required to accessing and querying databases. When using the Chart Designer, EspressManager provides a seamless connection between the query utilities and the database. EspressManager also provides data buffering, file I/O, and user authentication. Although EspressManager is required when running the Chart Designer, charts can be deployed using the API or the Chart Viewer without using it.

1.3. EspressChart Architecture

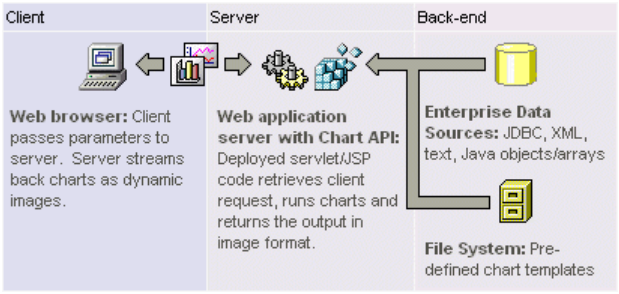
EspressChart has a number of different configurations in which it can run both at design-time and at run-time. At design-time the Chart Designer GUI interface, which includes data access tools, can be loaded as a Java Web Start Application in a client server architecture. The following diagram illustrates EspressChart running with the Chart Designer at design-time:



EspressChart Designer Architecture

When Chart 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(More in: Section 2.6 - Run Applets in WebStart with JNLP file)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 Chart Designer.

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



EspressChart in a Servlet Environment

When running in an application server, the Chart API can be used to generate charts within servlets and JSPs and stream the generated chart to the client browser as an image. Clients can also view charts by loading the Chart Viewer JNLP (More about Applets in JNLP: Section 2.6 - Run Applets in WebStart with JNLP file) in a web page.

In addition to running in a client-server environment, EspressChart can be run in a thick-client application. The Chart API can be used in conjunction with the Chart Viewer to show/generate charts in an application interface. In this configuration, EspressManager does not need to be running, as the whole process can be contained on the

client. Note that the diagrams above do not represent all possible deployment configurations. For more detailed information about deploying EspressChart, please see Chapter 12 - Deployment

Chapter 2. Installation/Configuration

There are four versions of the EspressChart installer, one for Windows, one for Unix, one for Mac OS X, and a Pure Java version. Although EspressChart itself is pure Java, the different versions make it easy to install EspressChart on the platform of your choice.

2.1. Running the Installer

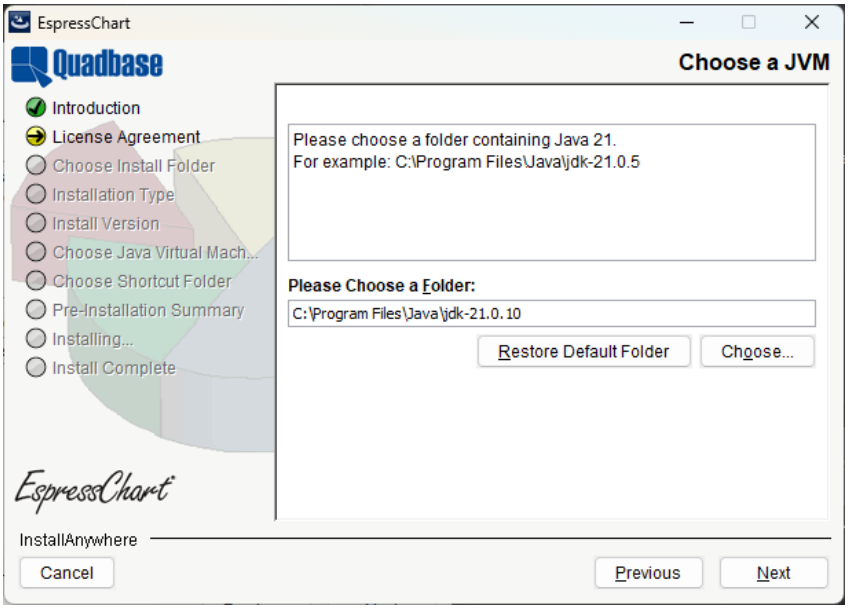
Windows Version — To start the Windows installer, run the `installEC.exe` file and the installer will launch.

Unix Version — To start the Unix installer, run the `installEC.bin` file and the installer will launch.

Mac OS X Version — To start the Mac installer, double click the `InstallEC.zip` file to extract the `InstallEC.app` file. Double-click on `InstallEC.app`, and the installer will launch.

Pure Java Version — To start the pure Java installer, a Java Virtual Machine, the equivalent of Java 21 or higher, must already be installed on the machine where EspressChart is to be installed. Make sure that the JVM is included in your path (or move the `installEC.jar` file to the same directory as the JVM). From a command prompt navigate to the directory where you have placed the `installEC.jar` file, and type the following command: `java -jar installEC.jar`. The installer will then launch.

Once the installation program has launched and you agree to the license agreement, the first option presented asks you to specify the directory containing Java 21. Select the folder where your Java 21 JDK is installed, for example `C:\Program Files\Java\jdk-21.0.5`.



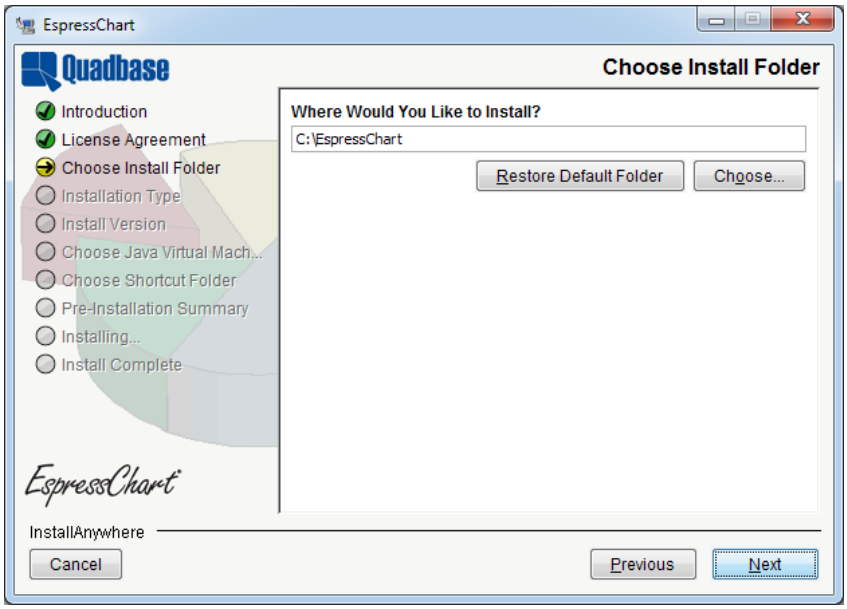
Choose Java Location Dialog

Note

In order to develop and execute Java code using the EspressChart 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.

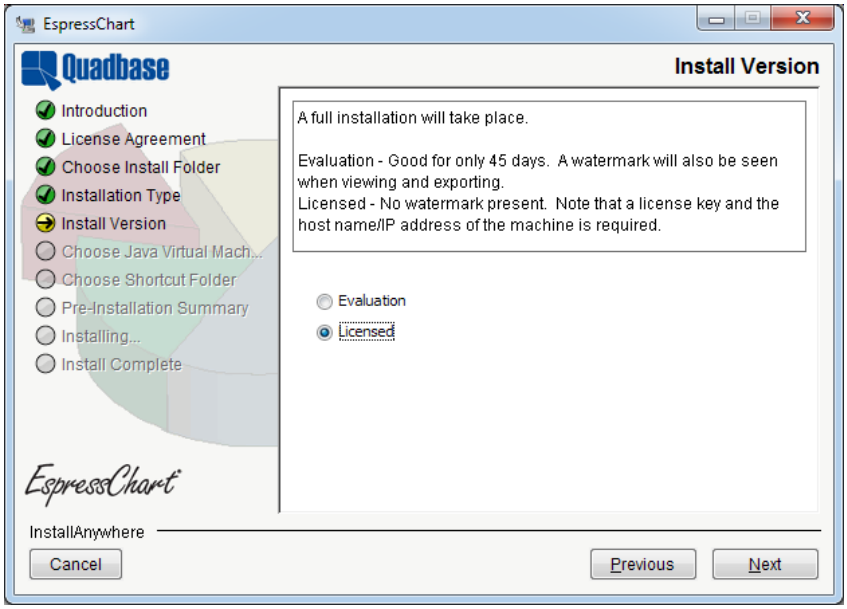
After you specify the location of your Java 21 installation, the installer asks you to choose the directory where EspressChart will be installed.

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



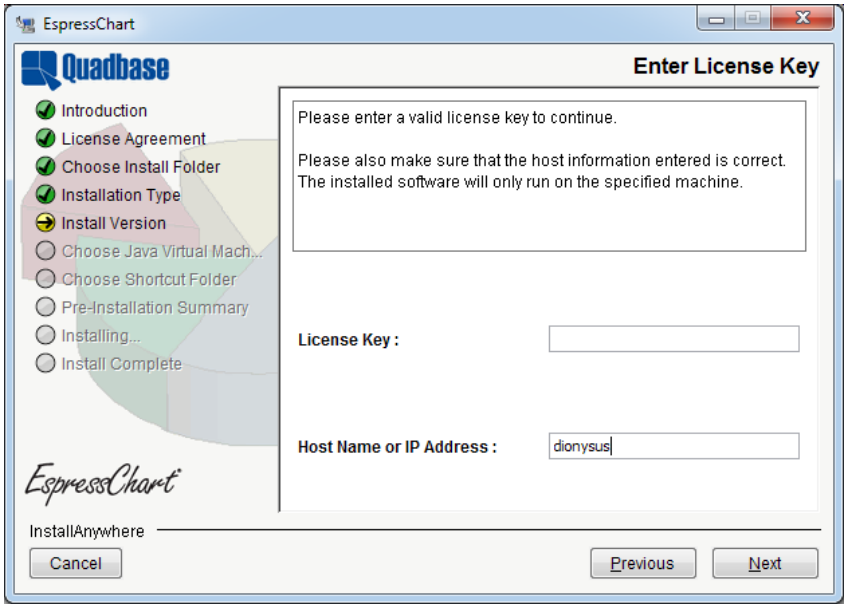
Choose Location Dialog

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



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 EspressChart.



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 EspressChart. 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 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 EspressChart 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.

2.2. Configuration

The configuration for EspressChart 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
- Print charts from applets using **Ctrl**+**J** or **Ctrl**+**P**
- Add/delete/modify Chart Designer 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: `<EspressChartInstallDir>/userdb/config.txt`. A sample configuration file is shown below:

```
[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
```

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 it 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.

The `[webroot]` section contains a path to the webroot of the web server. EspressManager use this when a chart is exported from Chart Viewer (for printing using **Ctrl**+**J** or **Ctrl**+**P**). For more information about this option, see Chapter 9 - Chart Viewer.

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.

2.2.1. Increasing maximum memory heap size

JNLP applications have a maximum memory heap size set by their launch configuration. To increase the heap size, modify the Java VM arguments in the JNLP file.

2.3. Starting EspressManager

Before ChartDesigner can be started, EspressManager must be running. In most cases, EspressManager should be started on the web server, but it is possible to use EspressChart on a stand-alone machine by running both EspressManager and ChartDesigner locally. The assigned IP number will be 127.0.0.1. This will create a log file `<EspressChartInstallDir>/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 ChartDesigner user machine.

To start EspressManager, run `EspressManager.bat` or `EspressManager.sh` file in the EspressChart 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.

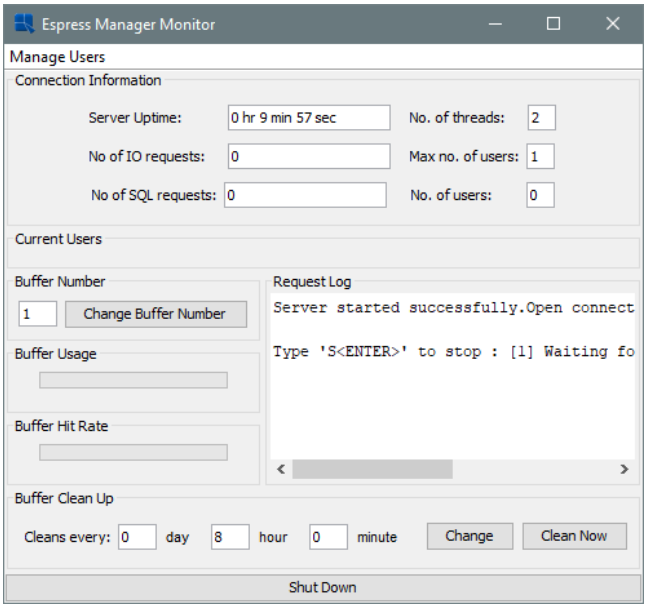
You can there also Add/delete/modify users.

"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.

-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.



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, EspressChart 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 ChartDesigner when it is launched via the API. Normally, when ChartDesigner 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 ChartDesigner when it is called from the API. With this argument turned on, every time a `QbChartDesigner` 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 ChartDesigner from the API, please see Section 10.5.6 - Calling Chart Designer from Chart 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 `QbChartDesigner` object. For more information about calling ChartDesigner from the API, please see Section 10.5.6 - Calling Chart Designer from Chart API.

-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 chart will continue, but the data will be read from the paging files. This feature prevents the system from running out of memory when running charts.

-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 chart.

-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 chart is generated.

-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 4.2.2.2 - Parameterized Queries.

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.

2.3.1. 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 EspressChart installed to C:\EC72

2.3.1.1. 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 the closing tag of the "Host" sections.

```
<Context path="/EC72" docBase="C:/EC72"><Manager pathname=""/></Context>
```

Example of the code placement in server.xml:

```
        <Valve className="org.apache.catalina.valves.AccessLogValve"
 directory="logs"
               prefix="localhost_access_log" suffix=".txt"
               pattern="%h %l %u %t &amp;quot;%r&amp;quot; %s %b" />
 <Context path="/EC72" docBase="C:/EC72">
  <Manager pathname=""/>
 </Context>

      </Host>
```

2.3.1.2. 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. Add the following lines on the 2nd line of the .bat file.

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

If you want to use a particular Java for starting the Tomcat, also add the following line:

```
set "JRE_HOME=C:\Program Files\Java\jdk-21.0.5"
```

Example of the first three lines of the startup.bat file:

```
@echo off
set "JRE_HOME=C:\Program Files\Java\jdk-21.0.5"
set "CATALINA_HOME=C:\Tomcat"
```

For Linux operating systems

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

Type the following two lines into the newly created <EC 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.

2.3.1.3. File launch settings

To make changes in the launching file, open `EspressChart.jsp` and find the `application-desc` section near the end of the file.

Make sure the servlet argument points to the same Tomcat context path configured in `server.xml`.

```
<application-desc main-class="quadbase.chart.designer.InitFrame">
    <argument>-enc:UTF-8</argument>
    <argument>-servlet:http://127.0.0.1:8080/EC72/servlet</argument>
</application-desc>
```

If the Tomcat context path is configured as `/EC72`, the servlet URL should end with `/EC72/servlet`.

In the next step, edit `designer.bat` or `designer.sh` and append the following parameter at the end of the file: `-servlet:http://your-server-ip:your-port/EC72/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 Chart Designer, Chart Viewer, and any application/servlet/JSP using the chart API. For more information about EspressChart deployment, please see Chapter 12 - Deployment.

2.4. Starting Chart Designer

Chart Designer can be run in two ways, as a stand-alone application or through a JNLP file loaded from a web browser. In either case, you must have EspressManager running in order to start the Chart Designer. Please see previous section for instructions on how to start the EspressManager.

Running as application

To start the Chart Designer as application, execute `Designer.bat` or `Designer.sh` file located in root directory of the EspressChart installation. For Windows installations, you can use program shortcuts from Start menu or desktop, depending on the options that were specified during installation. A dialog will appear prompting you to enter a username and password. 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 EspressChart* button and the application will start.

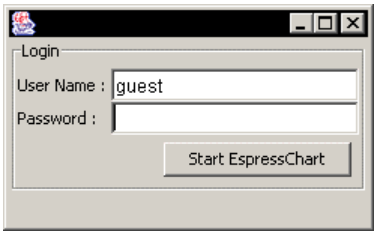


Chart Designer Login Window

Running through a Browser

If you're running Chart Designer remotely, make sure that EspressManager is running on the remote machine, then type the URL for EspressChart in your browser, i.e. **`http://machinename/espresschart/index.html`**. The page will contain a dialog prompting you to log on to EspressManager. If the `config.txt` file has been modified to set up users,

enter your username and password to log in. If the `config.txt` file has not been modified, enter the username **`guest`** with no password, then click the *Start EspressChart* button and the Chart Designer will launch.

Caution

Because EspressChart requires a JVM the equivalent of Java 21 or higher, you will need a compatible JNLP launcher in order to run Chart Designer via browser.

2.4.1. Connecting to EspressManager Running as a Servlet

If EspressManager is running as a servlet, as described in Section 2.3.1 - Starting EspressManager as a Servlet, you will need to make some modifications before ChartDesigner can be started successfully.

Stand-Alone

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

For example, **`-servlet:http://127.0.0.1:8080/EC72/servlet`** indicates that the EspressManager is running on localhost with port 8080 under the `/EC72` context. Below is an example of the modified designer.bat

set "JAVA_EXECUTABLE=C:\Program Files\Java\jdk-21.0.10\bin\java.exe" set PATH= %PATH%;.\lib "%JAVA_EXECUTABLE%" -Xmx256M -classpath ".\lib\axis.jar;.\lib\ax-is-commons-logging.jar;.\lib\axis-commons-discovery.jar;.\lib\axis-jaxrpc.jar;.\lib \ChartDesigner.jar;.\lib\ExportLib.jar;.\lib\FlashExport.jar;.\lib\hsqldb.jar;.\lib\jsqlparser.jar;. \lib\jsyntaxpane.jar;.\lib\qblicense.jar;.\lib\SVGExport.jar;.\lib\wsdl4j.jar;.\lib\xercesImpl.jar;. \lib\xml-apis.jar;.;.\lib\commons-codec.jar;.\lib\commons-collections.jar;.\lib \commons-compress.jar;.\lib\commons-io.jar;.\lib\commons-csv-1.12.0.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;.\lib\EspressAPI.jar;.\lib\bcprov-jdk18on.jar" quadbase.chart.designer.InitFrame %1 %2 %3 %4 %5 %6 %7 %8 %9 -servlet:http://127.0.0.1:8080/EC72/servlet

Browser

If you're running ChartDesigner through a JNLP application that is connecting to EspressMan-ager running as a servlet, make sure that the following `application-desc` argument tags are present near the end of the `EC72\EspressChart.jsp` file, which in our example, is shown below

The following argument tags need to be included within the element of `application-desc` of `jnlp` contents:

```
<application-desc main-
class="quadbase.chart.designer.InitFrame">
    <argument>-enc:UTF-8</argument>
    <argument>-servlet:http://127.0.0.1:8080/EC72/servlet</
argument>
</application-desc>
```

The `-servlet` argument indicates that the EspressManager is running as a servlet and specifies the URL of the EspressManager servlet.

Restart your Tomcat server.

To access the ChartDesigner, open your web browser and go to the following ad-dress: **`http://127.0.0.1:8080/EC72/index.html`**. You may either need to open

`EspressChart.jnlp` directly or save it to your download directory and run the jnlp file from there to open the ChartDesigner.

Tip

Sometimes the downloaded JNLP file cannot be launched after it is downloaded. In Windows 10/11, 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.

2.5. 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 from an older version, you may want to keep the old behavior, so your charts 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 `<EspressChart_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 the Chart Designer and Viewer, you have to edit `espressmanager.bat` and `designer.bat` files (if you use Windows), or `espressmanager.sh` and `designer.sh` files (if you use other OS) in your EspressChart 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 Chart Designer from HTML page, you also have to edit `index.html` file in your EspressChart 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

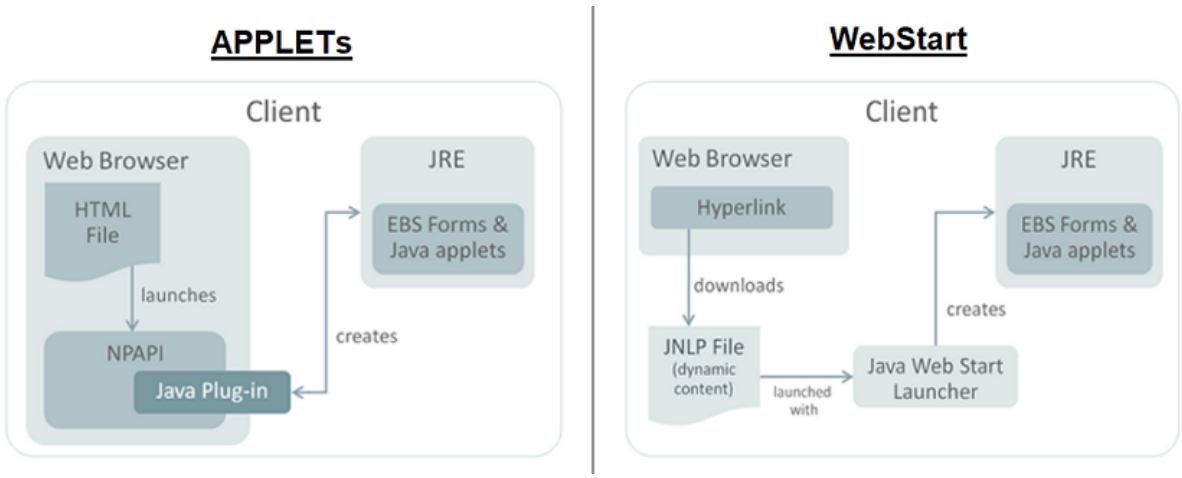
- behavior with the patch - Single color for categories feature is turned off by default
- new behavior has been introduced in version 6.3
- this feature can also be set using the `IDataPointSet.setSingleColorForCategories` method in API or in the Chart Options dialog in the Chart Designer

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

2.6. 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 WebStart technology. Both Applets and Java Web Start are marked as deprecated technologies from Java 9.



Applets vs WebStart

In this case, EspressChart is Java 21 compiled application and is best to use it with Java 21. 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 `application-desc` element in the JNLP file configuration file.

```
<application-desc main-class="quadbase.chartviewer.Viewer">
 <argument>-filename</argument>
 <argument>help/examples/ChartAPI/data/test.tpl</argument>

 <argument>-preventSelfDestruct</argument>
 <argument>true</argument>
</application-desc>
```

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

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 2.3.1 - 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 `application-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.

```
<application-desc main-class="quadbase.chartviewer.Viewer">
 <argument>-filename</argument>
 <argument>help/examples/ChartAPI/data/test.tpl</argument>

 <argument>-preventSelfDestruct</argument>
 <argument>true</argument>

 <argument>-comm_protocol</argument>
 <argument>servlet</argument>

 <argument>-comm_url</argument>
 <argument>http://127.0.0.1:8080</argument>

 <argument>-servlet_context</argument>
 <argument>EC72/servlet</argument>
</application-desc>
```

Note

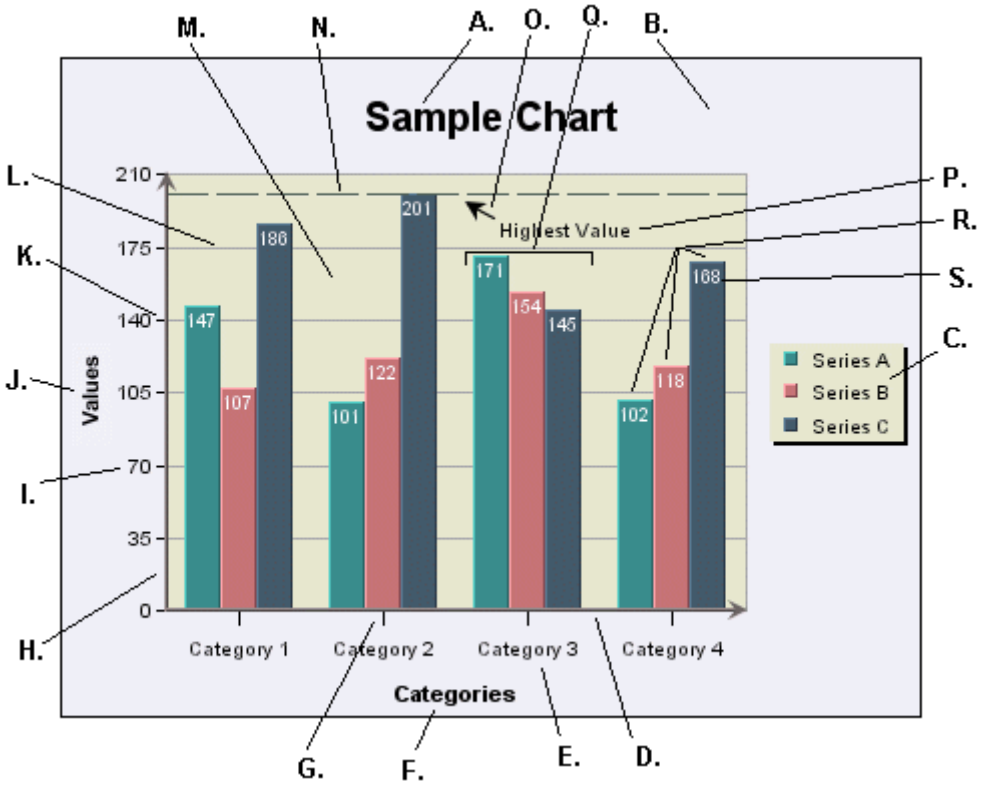
If you want to pre-fill the values automatically in a JSP file, the actual values highly depend on your server configuration. The previous examples are just illustrative.

Chapter 3. Charting Basics

This chapter covers some of the basics of EspressChart: from the different parts of a chart, to basic data mapping, to saving/exporting options. Everything discussed in this chapter is covered in greater details in later chapters.

3.1. 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 - 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. 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.

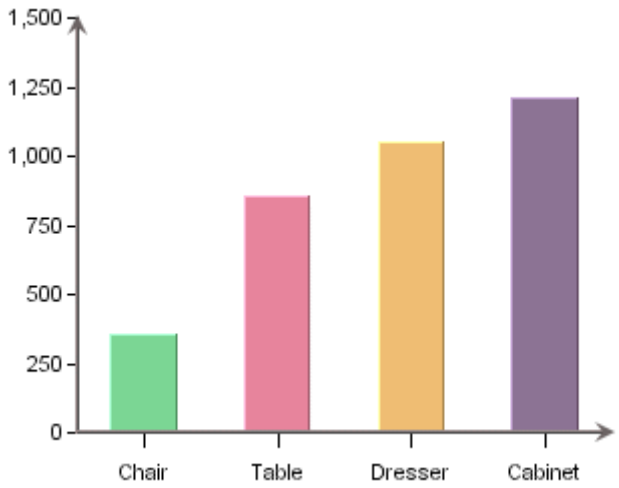
Chapter 4 - Working with Data Sources explains how to retrieve data from various sources.

A basic set of data might look something like this:

Product	Sales
Chair	362
Table	862

Product	Sales
Dresser	1052
Cabinet	1211

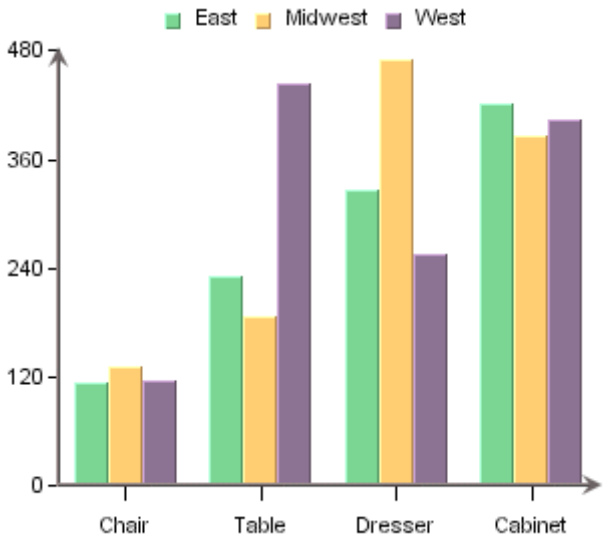
To plot this data in a chart, you would want to plot the **Sales** value for each entry in the **Product** column. Hence, the products are your category and the sales numbers are your values. In a chart you would map the **Product** column to the X (category) axis and the **Sales** column to the Y (value) axis. The resulting plot would look like this in a column chart:



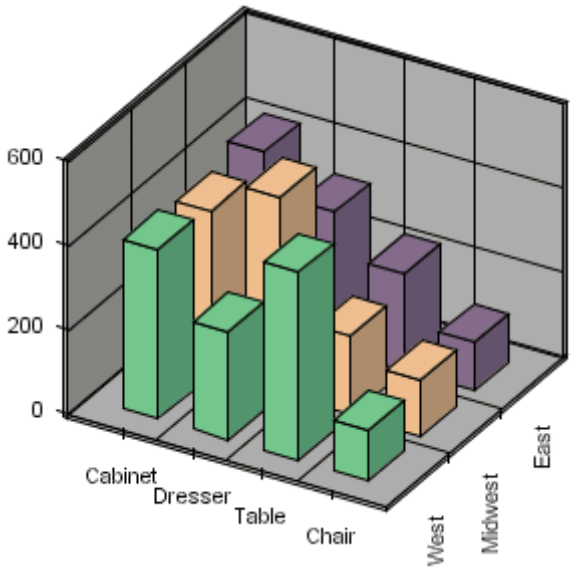
Here, a column is drawn to show the value for each distinct element in the category column. On top of the basic category values, additional information can be displayed in the form of a data series. For example, say that there is another element to our data that shows sales data not only for product, 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 Chapter 5 - Chart Types and Data Mapping.

3.3. Saving and Exporting Charts

There are several options available both for saving chart definitions and exporting charts as image files. More information on how to save and export charts, both in the Chart Designer and using the Chart API, can be found in Chapter 8 - Saving & Exporting Charts and Chapter 10 - EspressChart Chart API.

3.3.1. Saving Chart Definitions

There are two primary methods which you can use to store chart definitions created in EspressChart: either as chart or as template files.

Chart files 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. Hence, chart files are portable. Any time you open a chart file, it will open with the original data that was used to create the chart. After opening the chart, you can refresh the data from the source or change the chart's data source entirely.

Template files Template files save the chart in a binary file called `filename.tpl`. A template file stores only the chart definitions and stores only 10 records of data with the chart. Hence, any time a template file is opened it will try to connect to the original data source to retrieve the data. Because of this, template files can be less portable than chart files. Template files can also be used to pass chart attributes from one chart to another. To do this, you can apply a template to a chart. This will carry over many of the attributes of the template to the current chart. For more information about applying templates, please see Section 8.1.1 - Working with Templates.

In addition to saving chart definitions in binary format, chart files and template files can also be saved in XML format. These XML chart definition files can be modified outside of Chart Designer or Chart API.

3.3.2. Generating Image Files

Generally, charts can be deployed via the Web in one of two ways - either as applets or by generating image files. Applets can directly load chart definitions, either template or chart files. For more about using applets see the Chart Viewer section in Chapter 9 - Chart Viewer. For image files, EspressChart can render charts in the following formats:

GIF EspressChart can generate GIF images using one of two compression methods - RLE or LZW. The LZW method is faster and produces smaller files; however, its use is protected by patent. You must obtain a license from Unisys in order to unlock the LZW compression. By default, GIF files are generated using RLE compression.

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 the quality and compression of the file. The higher the quality, the larger the file.

PNG PNG is an image format that is less popular but it can be displayed in most browsers. It is a high-quality image with a smaller file size than JPEG. There are three different compression options available for this format.

SVG SVG (Scalable Vector Graphics) is a relatively new image format that 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 it allows the chart to be resized after export. Also, flash allows for high-resolution printing and produces a small file size.

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.

In addition to static images, EspressChart also allows the chart data to be exported as a text file, PDF, or an XML file.

Chapter 4. Working with Data Sources

The first step in designing a chart is to retrieve data you want to use. Within Chart 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.

4.1. The Data Source Manager

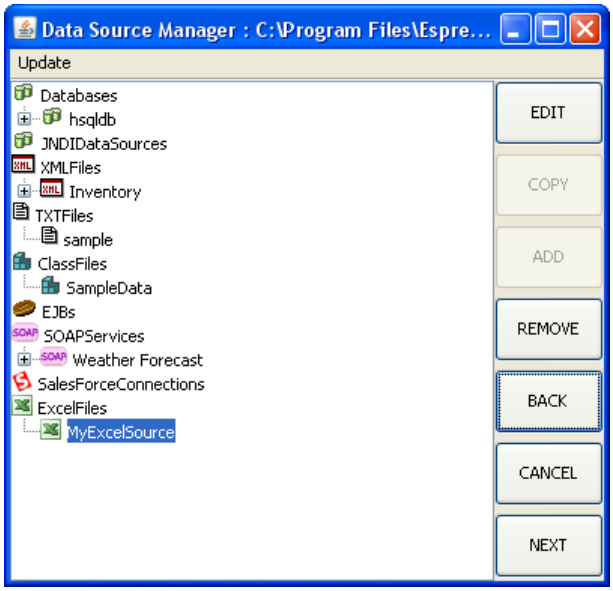
The Data Source Manager is an integrated utility that stores location and connection information for the data sources employed by a particular user. The information is stored in XML registry file. You can set up as many different data registry files as you like and there is no limit to the number of 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 charts since the data or the data source information is stored within the chart file.

Caution

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 chart.

4.1.1. Using Data Source Manager

When you start a new chart (by selecting File → New, or by clicking the *New* button on the toolbar), the chart wizard will launch. The Wizard is a step-by-step process that will guide you through the basic construction of your chart. The first dialog in the chart wizard will prompt you to specify the data registry file that you want 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.

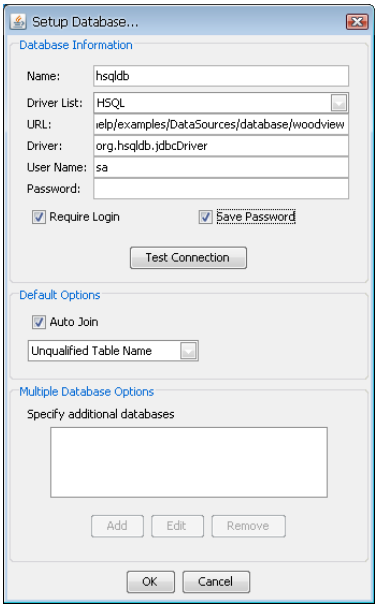
Cancel: This will cancel the wizard process.

Next: This will use the currently selected data source for the chart and proceed to the next step in the wizard.

4.2. Data from a Database

EspressChart can draw data from any JDBCcompliant database. In order to connect to a database via a 3rd party driver (other than the JDBC bridge), you will need to modify `EspressManager.bat` or `EspressManager.sh` file so that EspressManager will pick up the classes for the driver. Add the appropriate classes or archives to the `-classpath` argument in `.bat` or `.sh` file. 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.



Add Database Dialog

In order for EspressChart 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://machine-Name/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 recommend-

ed 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.

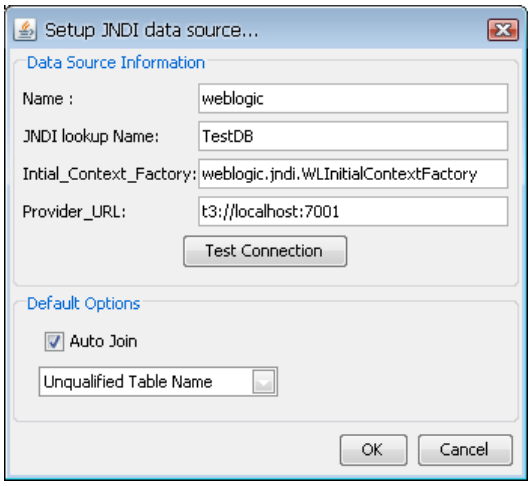
There are two sample databases included within the EspressChart installation. One is a HSQL (a pure Java application database) database and the other one is a MS Access database. Both contain the same data and are located in `help/examples/DataSources/database directory`. For details about how to set up connections to these sample databases, please see Section Q.3 - Set up Data Sources in the Registry.

4.2.1. JNDI Data Sources

In addition to connecting to databases via JDBC, EspressChart lets you use the JNDI (Java Naming and Directory Interface) to connect to data sources. In EspressChart, 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 charts between environments. If data sources in both environments are set up with the same lookup name, charts can be migrated without any changes.

To connect to a JNDI data source in the EspressChart, 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 2.3.1 - 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.



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.

4.2.2. Queries

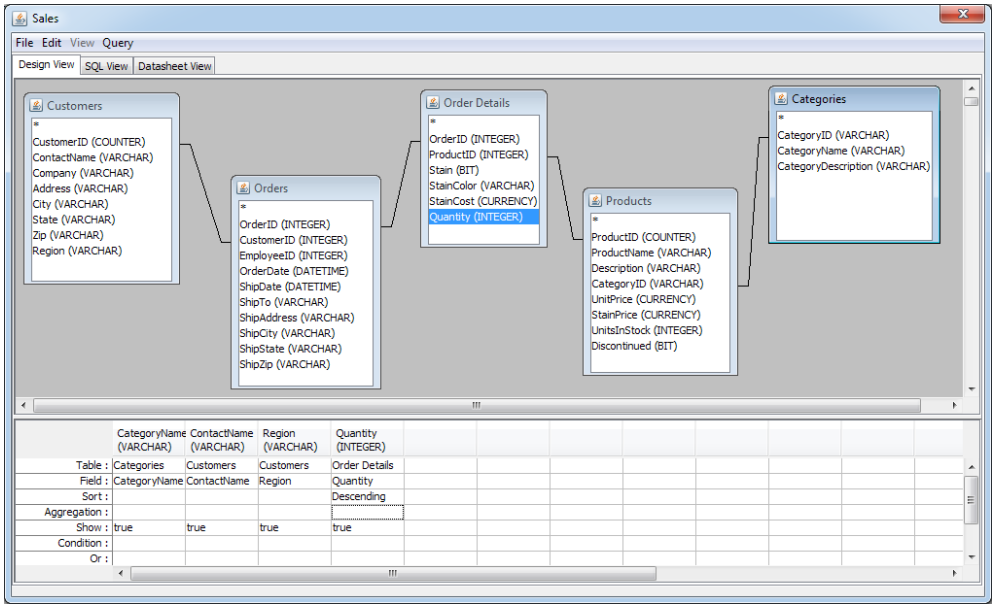
Once you add a database, a new node for your database will appear in the Data Source Manager window. When you expand the node, you will see two more nodes, one called *Queries* and one called *Data Views*. These are the two ways to retrieve data from your database. To create a new query, select the *Queries* 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.

4.2.2.1. 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.



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.

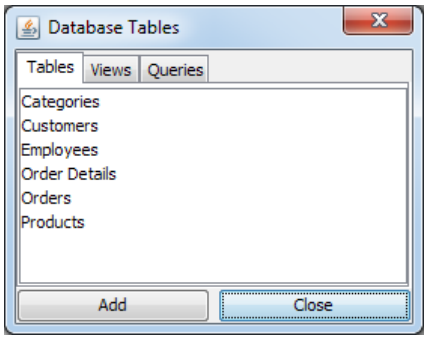
When you finish constructing the query, select *Done* from the File menu to return to the Data Source Manager.

4.2.2.1.1. 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

4.2.2.1.2. 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:

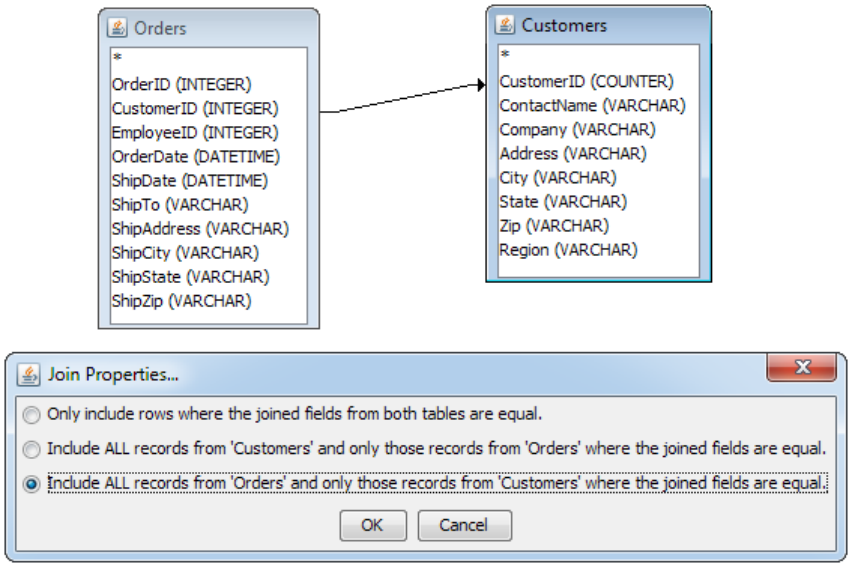
```
Select Orders.OrderID, Customers.CustomerName, Orders.Sales
From Customers, Orders
Where Customers.CustomerID = Orders.CustomerID
Order by Orders.OrderID;
```

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.



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

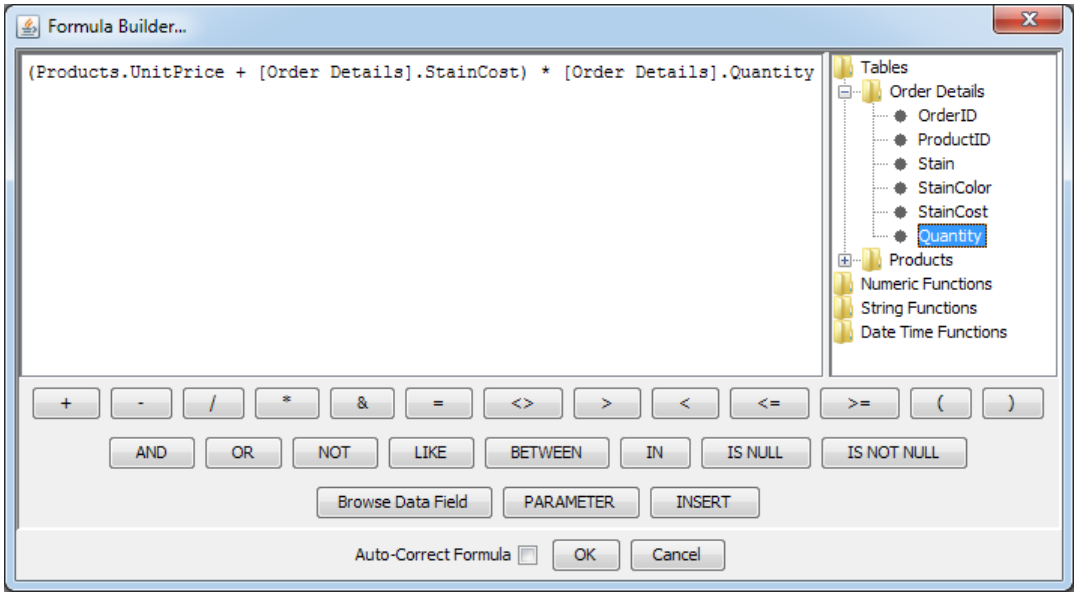
As you can see, all of the customer names have now been selected and null values have been inserted into the result set where there are no corresponding records. If you specify an outer join, the join line connecting the two tables in the Query Builder will become an arrow in the direction of the join.

4.2.2.1.3. 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.



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

EspressChart 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`, EspressChart will change the query condition

to `Categories.CategoryName='ARC'`. If you're using complex functions (i.e. database functions that take multiple string arguments), EspressChart 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.

4.2.2.1.4. Using Database Functions

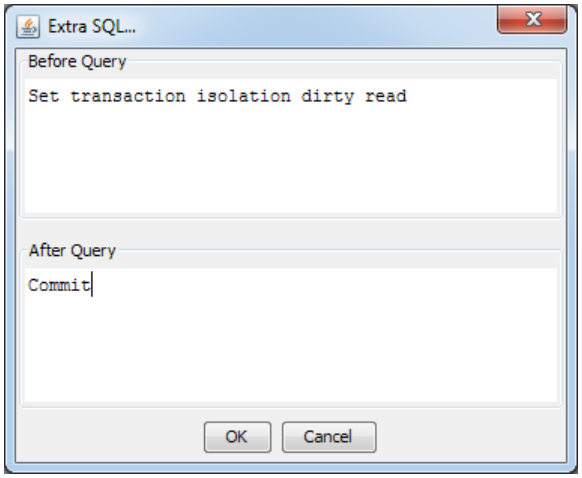
The formula builder component in the query builder allows you to use database specific functions when building a column or condition for the query. You can use the functions that are supplied or add your own to the interface.

EspressChart 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, EspressChart 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>
```

4.2.2.1.5. 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.



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.

4.2.2.1.6. 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.

4.2.2.2. Parameterized Queries

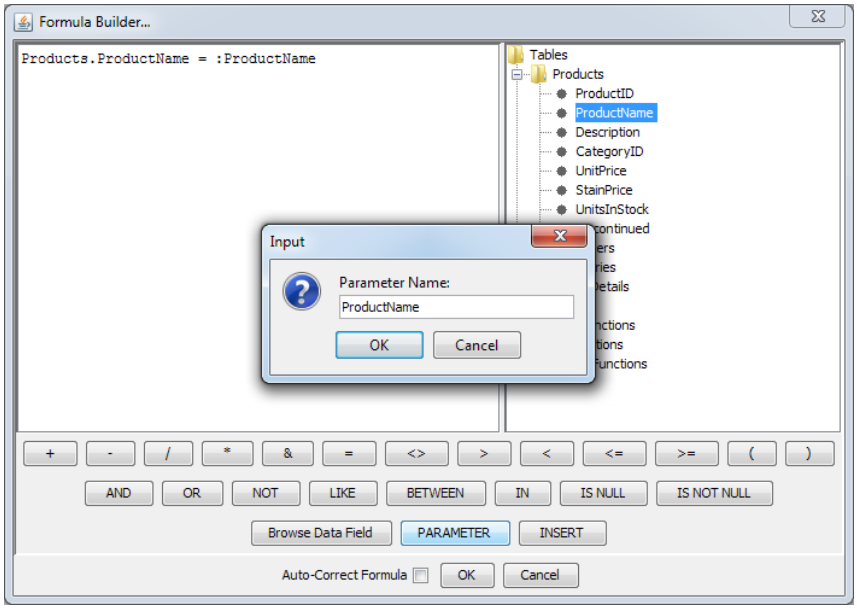
You can also use the Query Builder to design parameterized queries. This feature allows you to filter the data at run-time. Parameterized queries are also used for parameter drill-down in charts. For more information about drill-down charts, please see Section 7.3 - Parameter Drill-Down.

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.



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.

4.2.2.2.1. Multi-Value Parameters

EspressChart 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 chart 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.

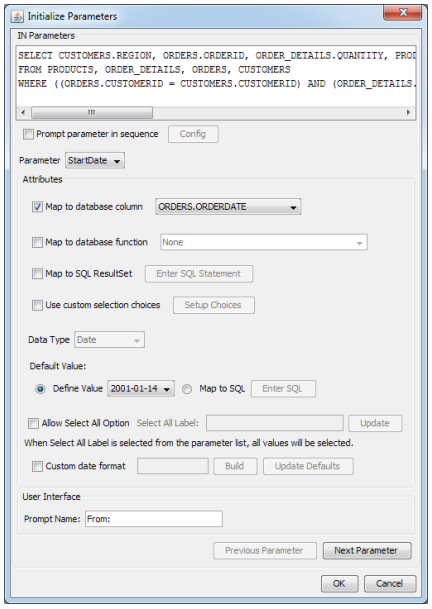
To create a multi-value parameter, place a parameter within an `IN` clause in an SQL statement. For example, the following query

```
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.

4.2.2.2.2. 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.



Initialize Parameter Dialog

From this dialog you can specify the following options:

Map to database column:

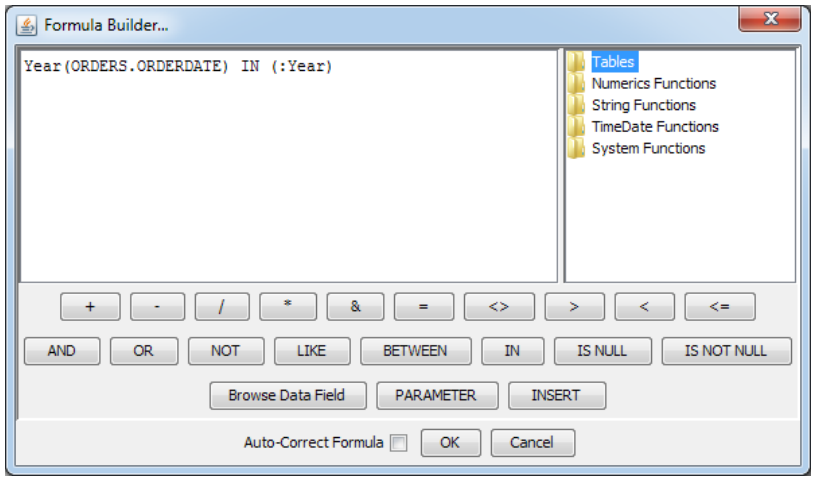
This option allows you to specify a column from the database whose values will be used for the parameter input. Selecting this option modifies the parameter prompt that the end user will get when previewing or running the chart in the chart Viewer. If you map the parameter to a database column, then 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 2.3 - Starting EspressManager

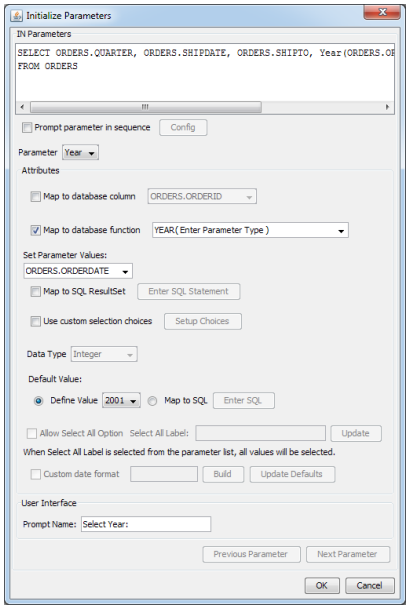
Map to database function:

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:



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.



Map Parameter to Database Function

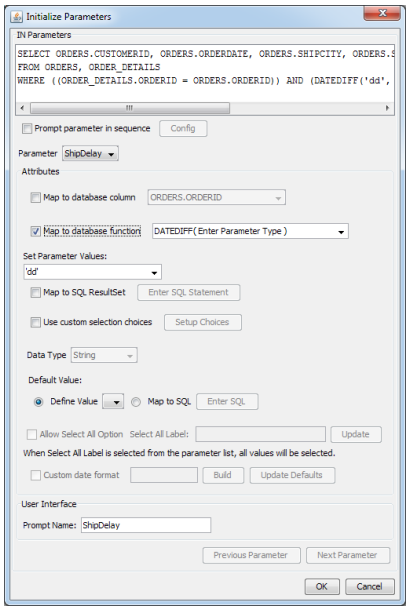
The list of custom functions is extracted from `DatabaseFunctions.xml` file located in `/userdb/` directory. Modify the .xml file if you wish to add a new database or custom functions. The new functions will appear in this list after you restart the program.

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.

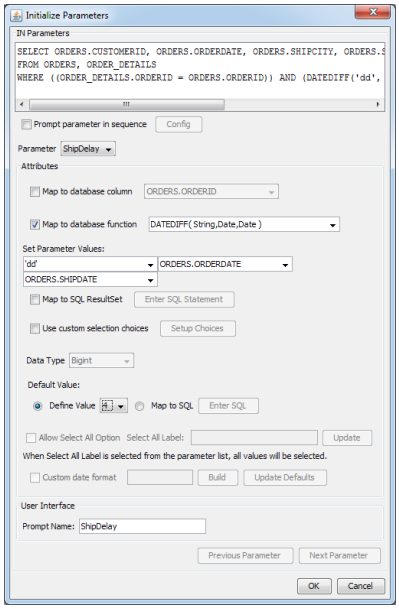
```
DATEDIFF('dd', ORDERS.ORDERDATE, ORDERS.SHIPDATE)
 >= :ShipDelay
```

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:



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.



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

chart. 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. An example is as follows:

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)))
```

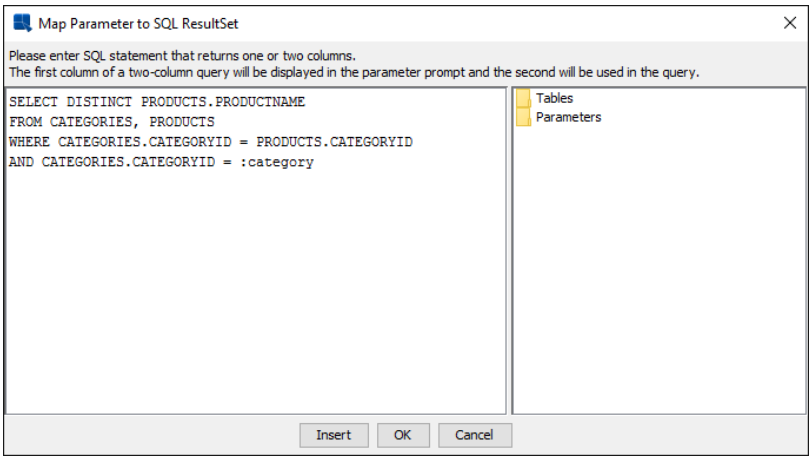
In the `config` prompt in `initialize parameter`, set the order for parameter prompting to `category` first, then `product`.

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
```

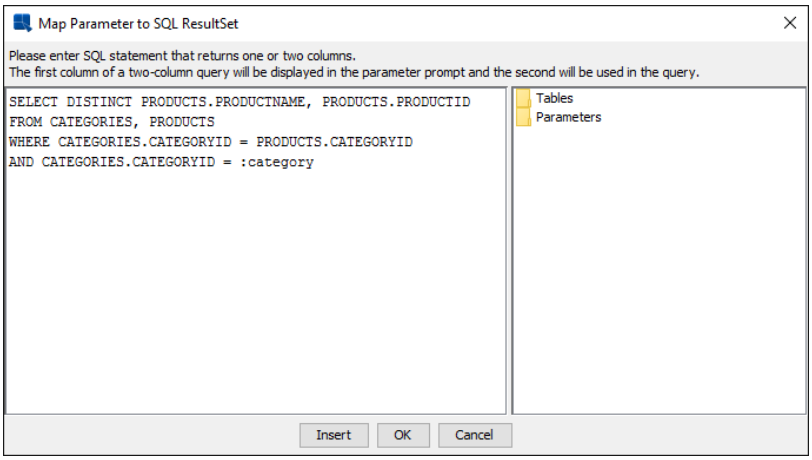


Select Statement for Product

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.

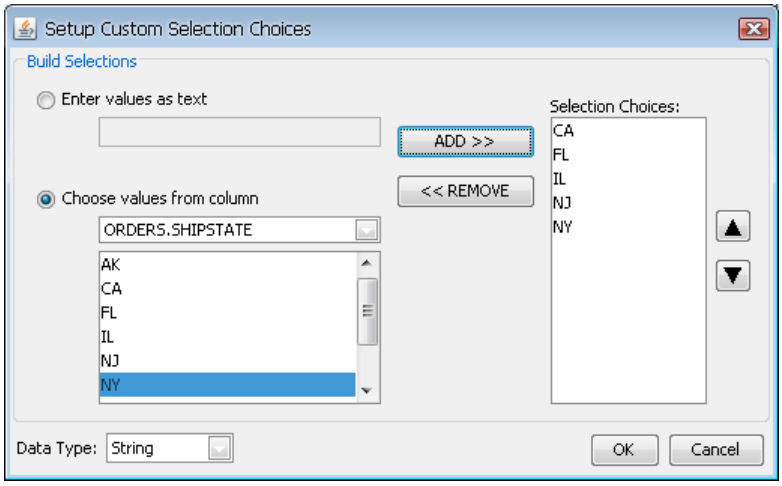
Suppose your database has a table with product ID as the primary key. When your end user wants to search for products from the database, they want to use the *product name* as parameter (therefore it is listed in the query as first) since a *product ID* could be just a cryptic code (therefore it is listed 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.



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.



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:

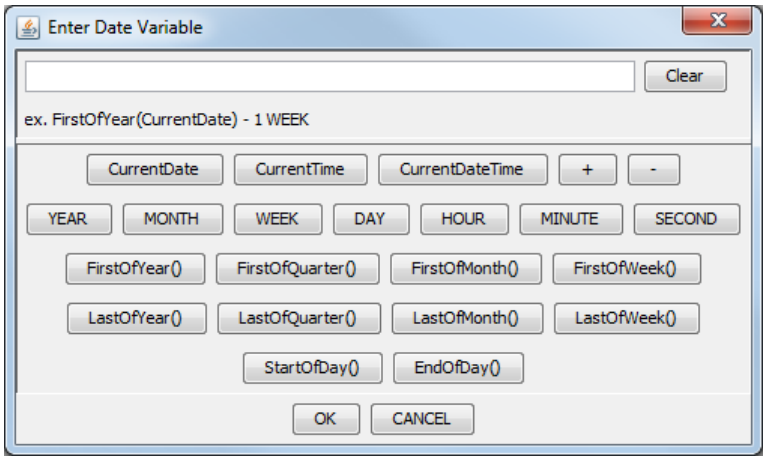
This allows you to specify a default value for the parameter. Although you don't have to specify a default value, it is recommended that you do so. If you do not supply a default value, you cannot open or manipulate the chart template without the data source present.

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 4.2.2.2.1 - 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.



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 chart can have the following default values:

```
StartDate: CurrentDate - 1 WEEK
EndDate: CurrentDate
```

This would indicate that every time the chart 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)`

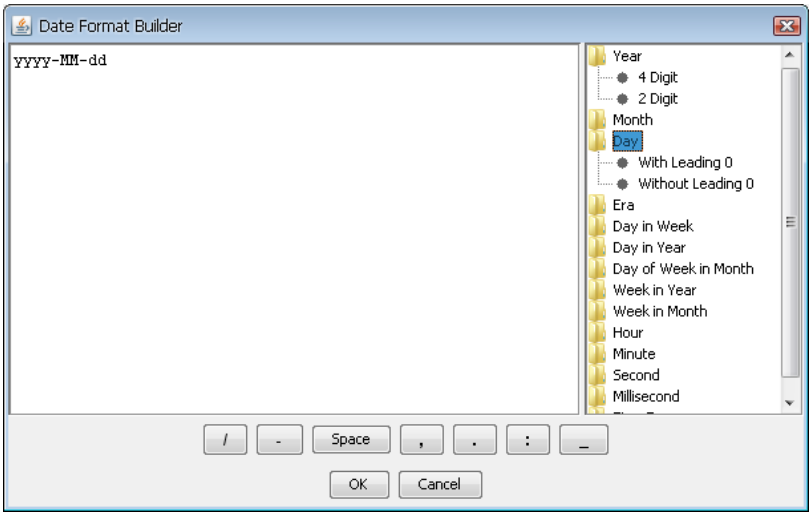
	This function returns a date of the first day of the year from the argument. For example, when the argument evaluates to `2012-08-14`, the function returns `2012-01-01`.
LastOfYear()	Argument format: `CurrentDate`, `CurrentDateTime`, e.g. `LastOfYear(CurrentDate)` This function returns a date of the 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 4.2.2.2.3 - 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.



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.

The following table illustrates the character combinations that you can use:

Character	Represents	Output(text/ number)	Example
G	era	text	AD
y	year	number	1996, 96
M	month in year	text or number (depends on length)	July, Jul, 07
d	day in month	number	10
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	milisecond	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 2nd 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

Character	Represents	Output(text/ number)	Example
z	time zone	text	Pacific Standard Time, PST

You can piece together almost any combination of these characters to produce a date expression in the format you want. The count of groups of characters determines the form the element will take. For text elements, 4 or more characters in a group will cause the full form of the element to be used. If less than 4 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, 4 or more in a group will display the full version, 3 will display the abbreviation, and 2 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 be displayed as entered. You can also insert words and expressions by enclosing them within single quotes (type two single quotes to insert an apostrophe as text).

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 chart, or open a chart that uses a parameterized query, the chart will load/start with the default values. You will be prompted to provide parameter values when you preview the chart.

4.2.2.2.3. 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.

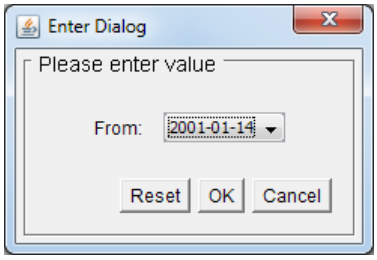
Note

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:

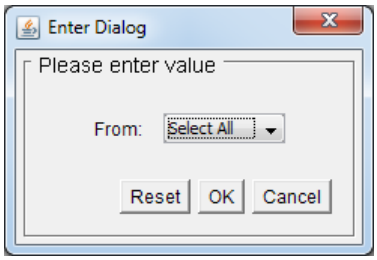
```
WHERE column = :Parameter
```

In such case, the parameter prompt dialog will not allow you to select more than one value.



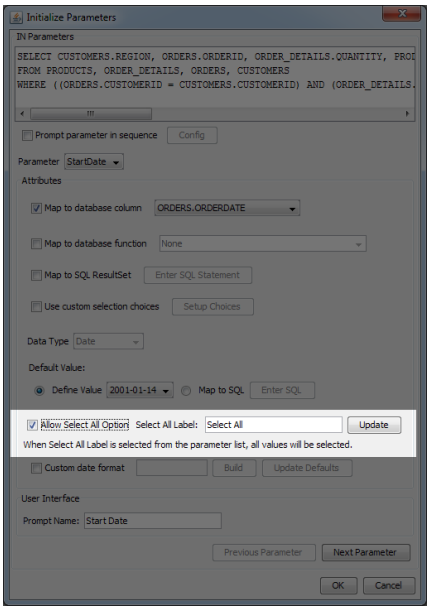
Typical single-value parameter

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).



Single-value parameter with the Select All functions enabled

The *Select all* feature can be enabled in the *Initialize Parameters* dialog (see Section 4.2.2.2.2 - Initializing Query Parameters for more details) by selecting the *Allow Select All Option* checkbox.



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

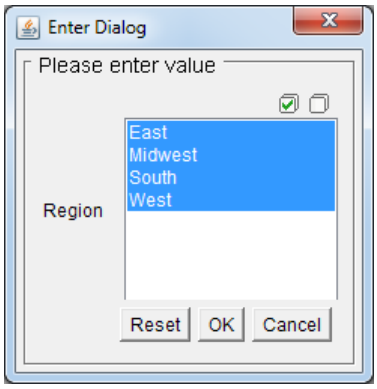
= equal to

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 performance 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 chart 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 EspressChart 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'))
```

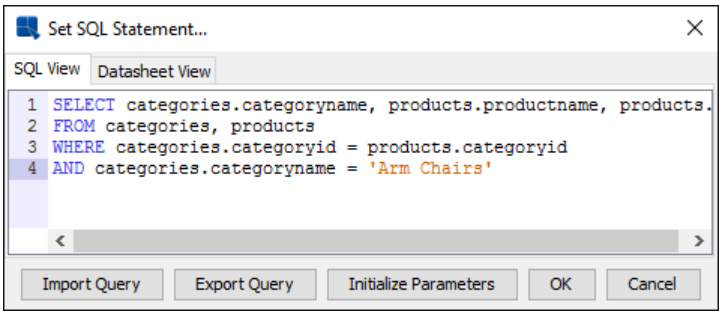
Note

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/].

4.2.2.3. 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.



Enter SQL Statement Dialog

To preview the result set, click on the *Datasheet View* tab.

4.2.2.3.1. 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.

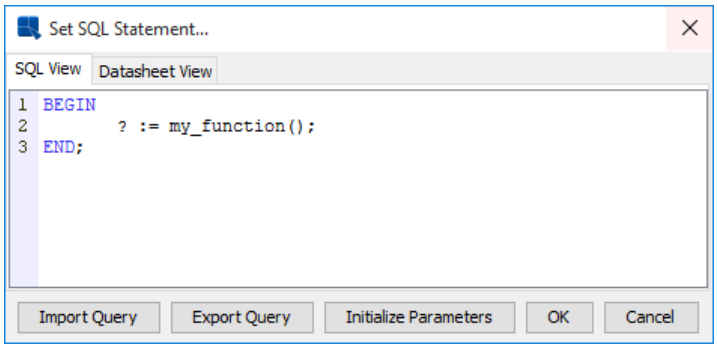
To access your Oracle stored procedures, the first step is to define a weakly typed `REF CURSOR` using the following PL/SQL statement:

```
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 ChartDesigner using a special PL/SQL like syntax. In the Set SQL Statement window, enter the following syntax to call the Oracle stored function:



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 ChartDesigner that a variable is reserved for the `OUT` parameter. The JDBC syntax for calling Oracle stored procedures is as follows:

```
( call ? := my_function() )
```

However, EspressChart 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 EspressChart 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

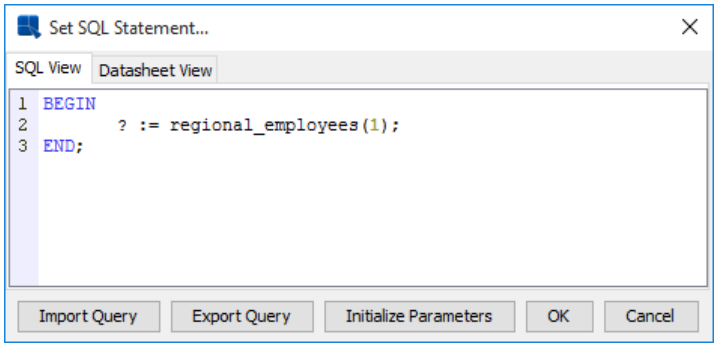
The table lists various corporate locations in a tree structure. The numbers of employees are stored in the leaf nodes (e.g. New York, London, etc.) and each node contains information about its immediate parent. Suppose you want to create a chart that displays the number of employees in a certain region and information about the separate branches within that region. For example, if the user inputs **`ID = 2`** (America), you want the chart 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:



Calling regional_employees function

Preview the results by clicking the *Datasheet View tab*.

SQL View | Datasheet View

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

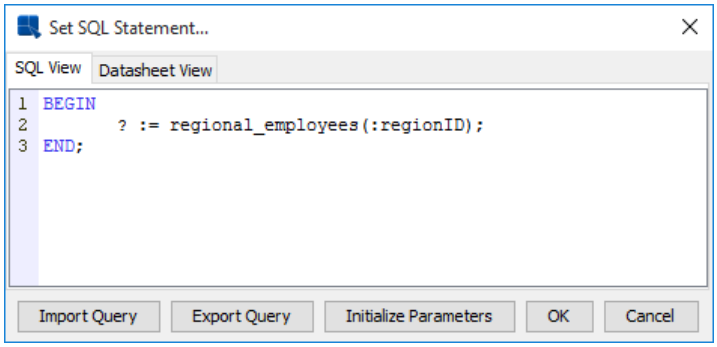
As seen in the results, the `CONNECT BY` clause traverses the tree recursively listing the American nodes together before listing the European nodes. If the user is only interested in European locations, they can enter **3** for the parameter and the following result set would return this:

SQL View | Datasheet View

ID	NAME	EMPLOYEES
3	Europe	25
7	London	14
8	Paris	11

Result set from regional_employees in Europe

To create a parameterized chart, use the `:param_name` syntax. The SQL parser in EspressChart 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:



Calling Oracle Stored Function using Parameter

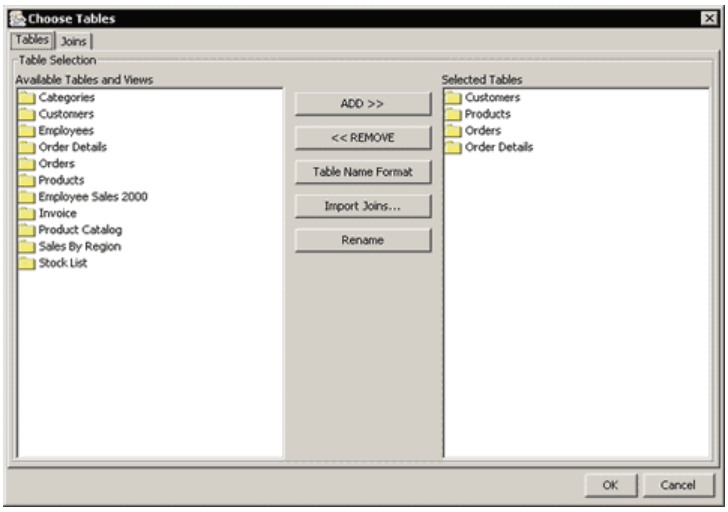
When using `IN` parameters, it is necessary to initialize the parameters prior to executing the query. It is especially important to set the correct default data type for executing stored procedures, because the parameters cannot be mapped to existing columns. You can find more information about initializing parameters in Section 4.2.2.2.2 - Initializing Query Parameters. To try this example, `<EspressChartInstall>\help\examples\data\locationHierarchyExample.sql` contains SQL commands to create `employee_table` as well as two stored functions.

4.2.3. Data Views

In addition to the query interfaces, EspressChart 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 chart.

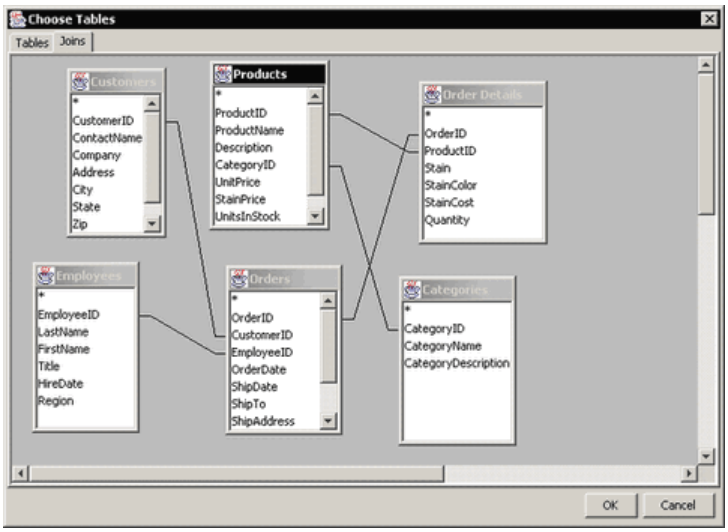
To create a data view, select the *Data Views* node in the Data Source Manager window and click the *Add* button. A new window will open allowing you to select the database tables you want to use for the data view.



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.

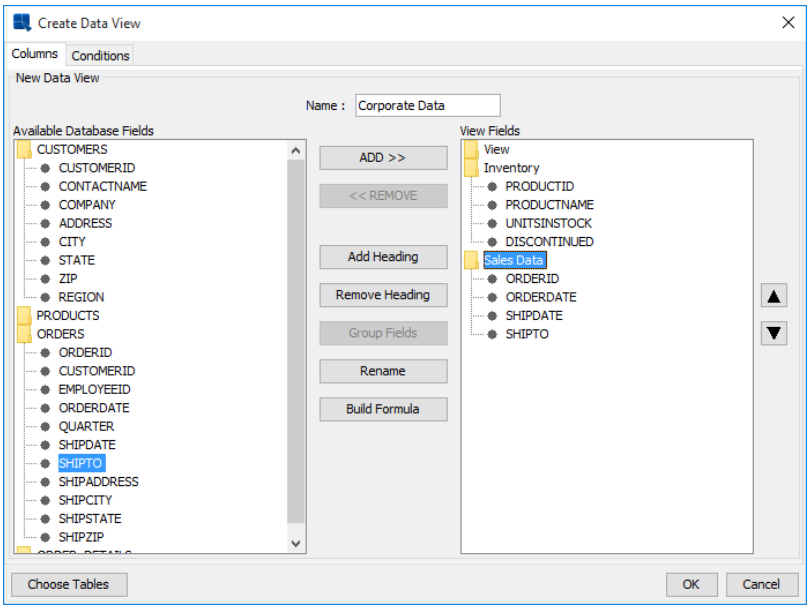
The *Joins* tab of this window allows you to specify the joins between the selected tables.



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 4.2.2.1.2 - 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.



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.

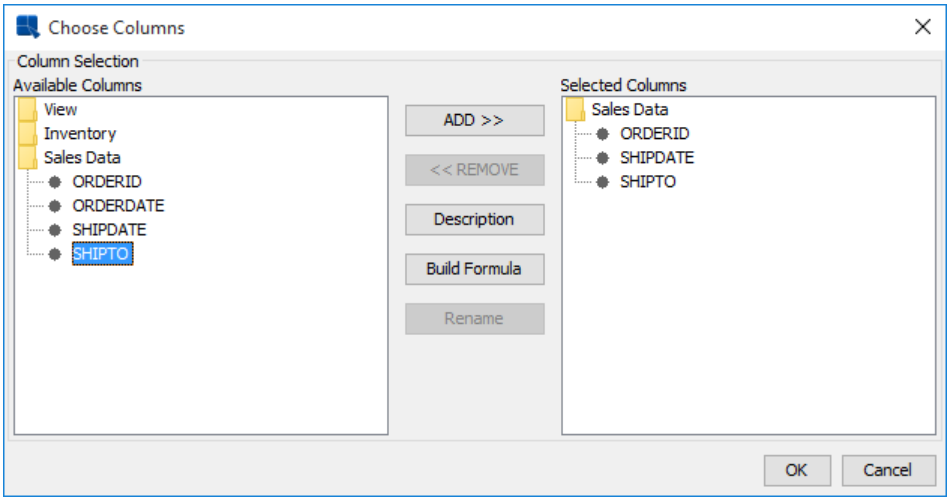
You can also group fields within the data view by adding headings. This allows you to create your own organizational structure of *virtual tables* that group data from different database tables under one heading. To create a heading, click the *Add Heading* button. You will then be prompted to specify 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 4.2.2.1.3 - Columns.

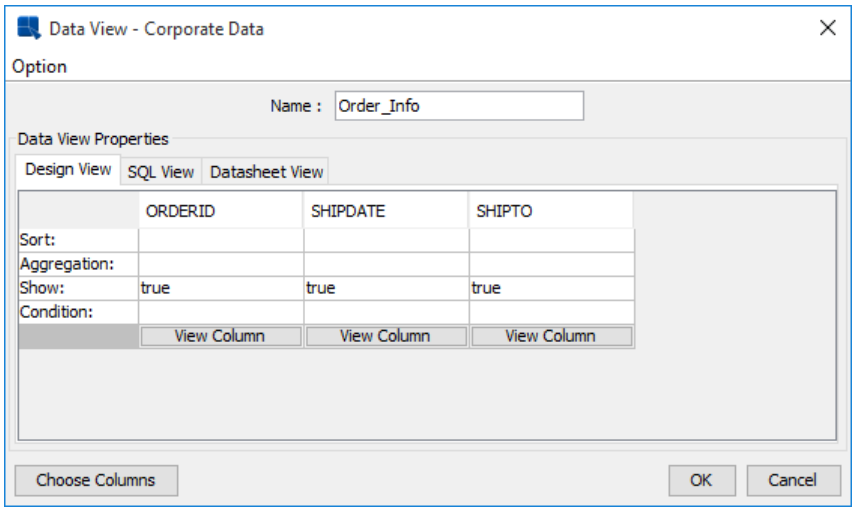
Once you finish creating the data view, click the *OK* button and the data view will be added to the Data Source Manager. Users can now use this view to construct ad-hoc queries.

When you design a chart 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 chart. 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



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 chart created from the data view will refer to the data view query for updating/modification.

4.2.3.1. Data View Parameters

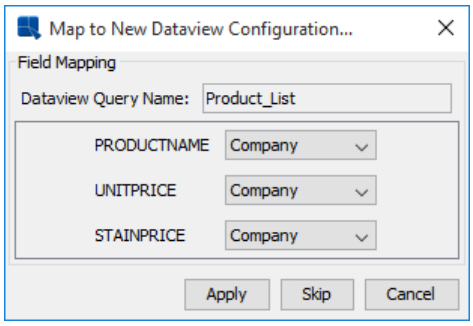
As with Query Builder, users can specify query parameters in Data Views. To add a parameter to a data view, select a data view in the Data Source Manager and click *View* to run the data view. After you select fields for the data view and you are in the conditions window, right-click in the *Condition* 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 4.2.2.2 - 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 chart wizard, or if you save the selections as a query. You can also initialize the parameter by selecting *Initialize Parameters* from the Option menu.

4.2.3.2. 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.



Update Query Fields Dialog

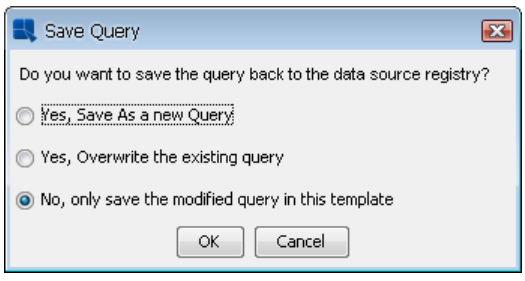
For each query, you will be prompted to change any fields that no longer match the data view structure. For each field, you can select a field from the data view to map it to, or remove the field from the query. If you do not want to change anything in the query, you can click the *Skip* button. The 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.

4.2.4. Editing Queries

If you select to build a chart 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 Chart Designer without having to go back to the Data Source Manager.

To modify a chart's query, select *Modify Query* from the Data menu in Chart Designer. If you have designed a query in the Query Builder, 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.

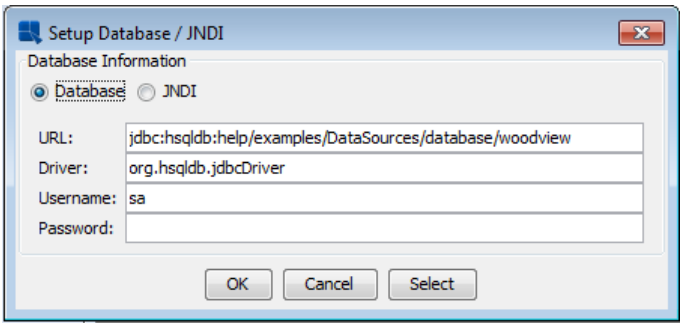


Saving Query Options

Once you specify the save options, the modified query will be applied to the chart. Note that if you made significant changes to the query, you may need to perform the data mapping for the chart again. For more information about data mapping, please see Chapter 5 - Chart Types and Data Mapping.

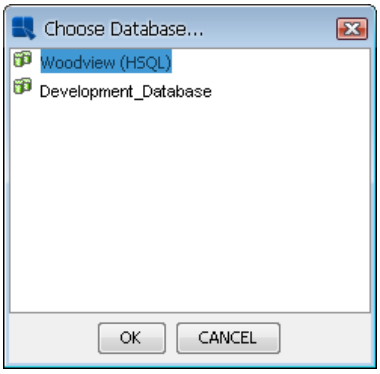
4.2.5. Editing Database Connections

If you have selected to build a chart using database data, you can also directly edit a template's database connection information from within the chart Designer. To modify the connection information in the chart, select Data → Modify Database. This will bring up a dialog allowing you to specify a different setting for the chart to use when connecting to the database or JNDI data source.



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.



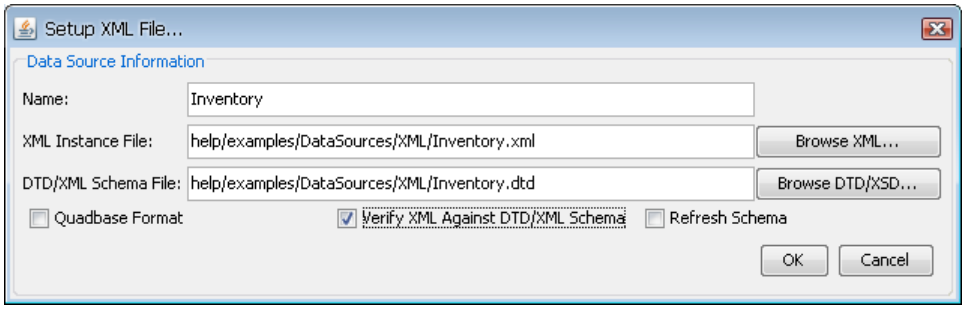
Select Database from Registry

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 and the changes will be applied.

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.

4.3. Data from XML and XBRL Files

In addition to relational databases, EspressChart 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.



Setup XML Data Source Dialog

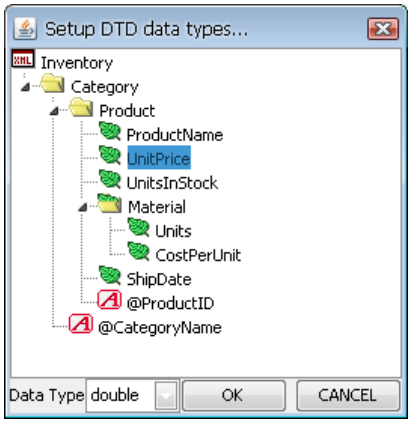
The first option allows you to specify a display name for the XML data source. The second option allows you to specify the location of the XML file from which you want to retrieve data. Note that you can also specify an XBRL file in this field. You can set up a data source that retrieves XML data from 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 EspressChart. For example, if you select to export a chart's data in XML format, you can read it back in using this format. For more on exporting to XML, see Section 8.2 - Exporting Charts. 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.



DTD Data Type Selection Dialog

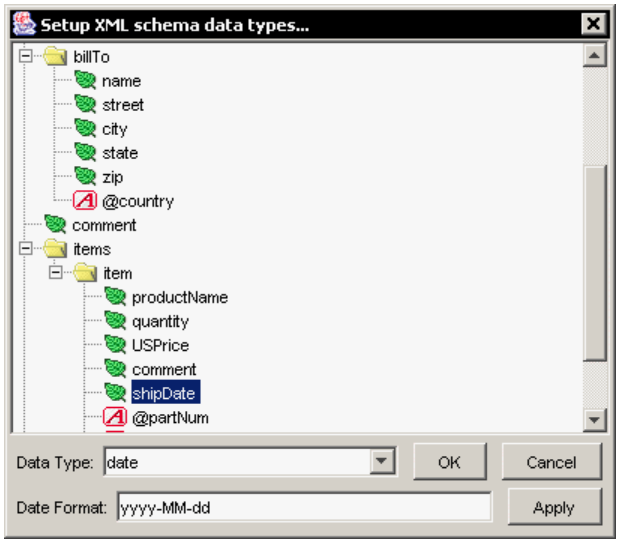
Because DTD files do not specify a correct data type for elements, all elements are considered to be Strings by default. To change the data type of an element, you have to select the element and pick a data type from the drop-down window at the bottom of the dialog. To ensure proper results when you 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.

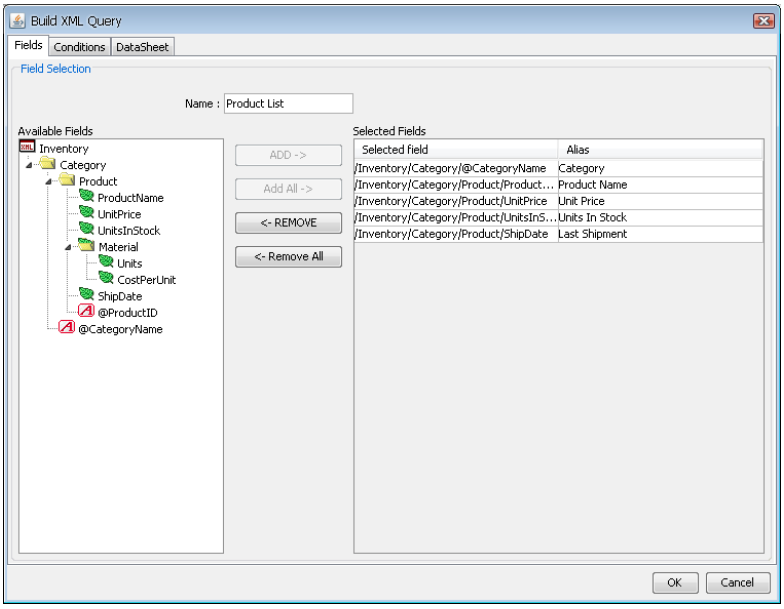


XML Schema Data Type Selection Dialog

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.

4.3.1. 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 EspressChart. 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.



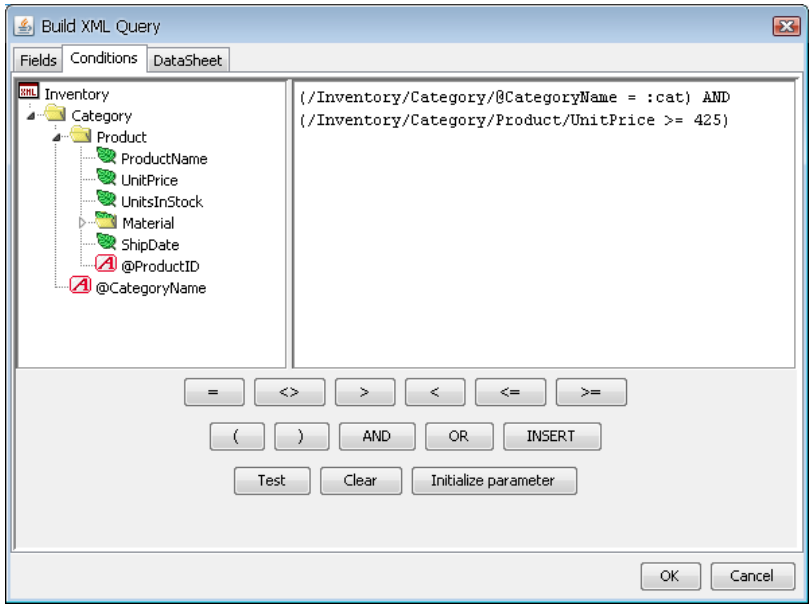
XML Query Field Selection Tab

The first tab in the XML query builder allows you to select fields/nodes from the XML file you want to use in the chart. 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 chart. 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.



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 4.2.2.1.6 - 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 chart.

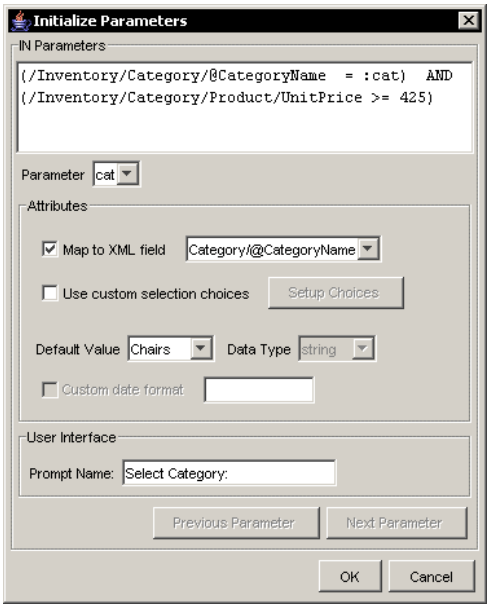
There is a sample XML file and a sample DTD/XSD included in the EspressChart installation. The files are located in `help/examples/DataSources/XML` directory of your installation and are called `Inventory.xml` and `Inventory.dtd`. There is also a sample servlet that allows you to stream XML data to the ChartDesigner. The servlet code and instructions are located in `help/examples/DataSources/XML/servlet` directory.

4.3.1.1. 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
```

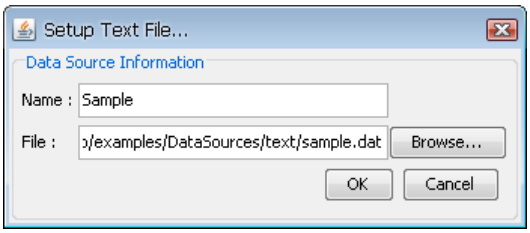
would place a dynamic filter on the query for the *CategoryName* attribute. XML parameters are initialized in the same way as query parameters. The initialization dialog will appear if you try to preview or close the query, or you can trigger it by clicking the *Initialize Parameters* button. The only difference is that instead of mapping to a database field, the parameter prompt can be mapped to a node in the XML file.



XML Parameter Initialization Dialog

4.4. Data from Text Files

EspressChart 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.



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.

4.4.1. Formatting Requirements for Text Files

There are certain formatting requirements for the data within a text file in order make it readable by EspressChart. 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).

4.4.2. 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 EspressChart
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

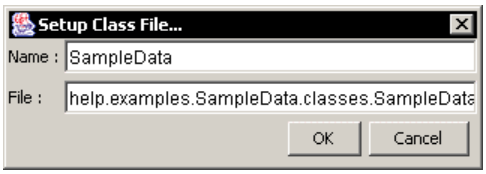
There is a sample text file included in the EspressChart installation. The file is located in `help/examples/DataSources/text` directory of your installation and is called `Sample.dat`.

4.5. Data from Class Files

For maximum flexibility, EspressChart allows you to design charts using object or array data by providing an interface to pass data to the chart 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 EspressChart.

For more information about this feature, please see Section 10.5.3.4 - Data from 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.



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 ChartDesigner. 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 ChartDesigner).

4.5.1. Parameterized Class Files

EspressChart provides an additional API interface, `IParameterizedDataSource`, that allows you to define chart parameters within the context of a class file data source.

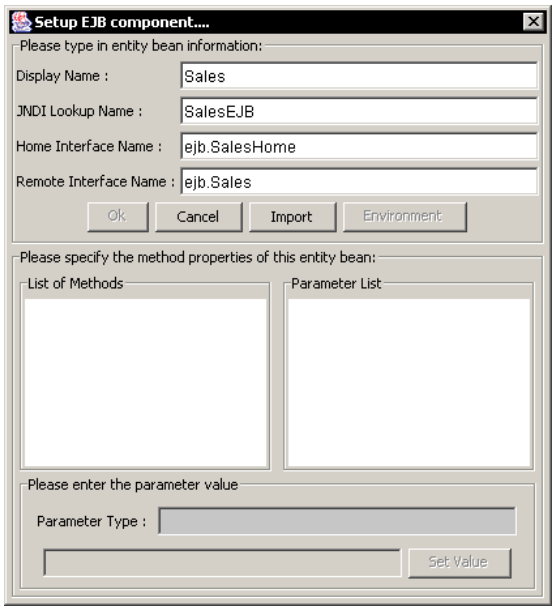
Parameters that are defined within the context of a class file work in the same way as query parameters. For more information about setting up a parameterized class file data source, please see Appendix 10.A.5 - Data Passed in your Custom Implementation.

4.6. 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 charting tool making redundant database connections. For this situation, EspressChart allows users to query data directly from an entity bean.

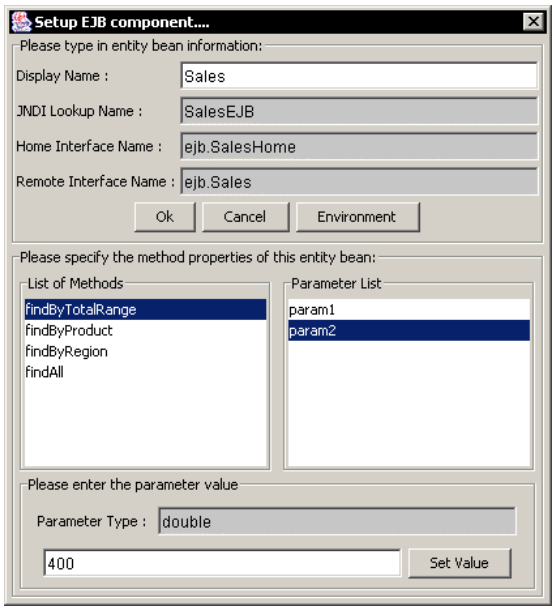
To add an EJB as a data source, the EJB must first be deployed in the application server and the client JAR file containing the appropriate stub classes must be added to your classpath (or the `-classpath` argument of the `EspressManager.bat` or `EspressManager.sh` file when using chart Designer). Select 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.



Add EJB Dialog

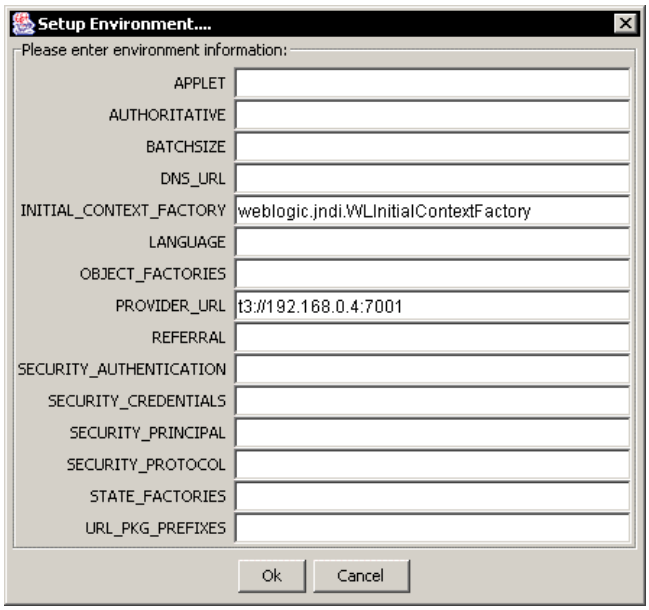
To connect to an EJB data source, you must provide the JNDI lookup name for the bean (this is specified when you deploy the bean). For EJB 1.1 users, specify the name for the home interface and the remote interface. For EJB 2.0 users, specify the local home interface and the local interface. Once you 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.



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.



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.

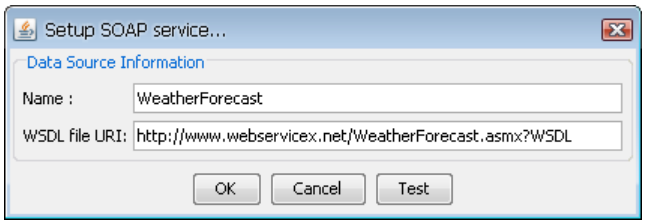
You can modify the parameter values by selecting your EJB source, and clicking the *Edit* button.

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 21 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.

4.7. Data from SOAP with WSDL support

EspressChart 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.

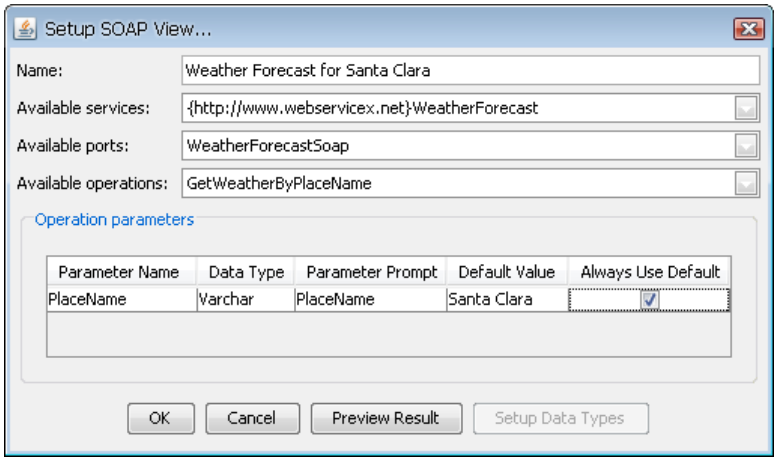
To set up a SOAP data source, select the *SOAPServices* node in the Data Source Manager and click the *Add* button. A dialog will open prompting you to specify options for the new SOAP data source.



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 EspressChart 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.

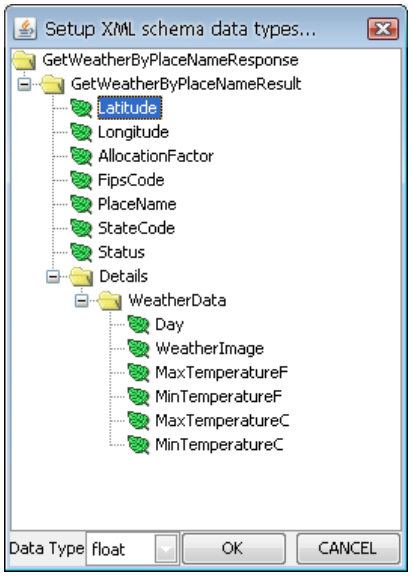
To add a new SOAP View, select an existing SOAP data source and click the *Add* button. A dialog will open prompting you for all the parameters necessary to make a SOAP query.



Setup SOAP View dialog

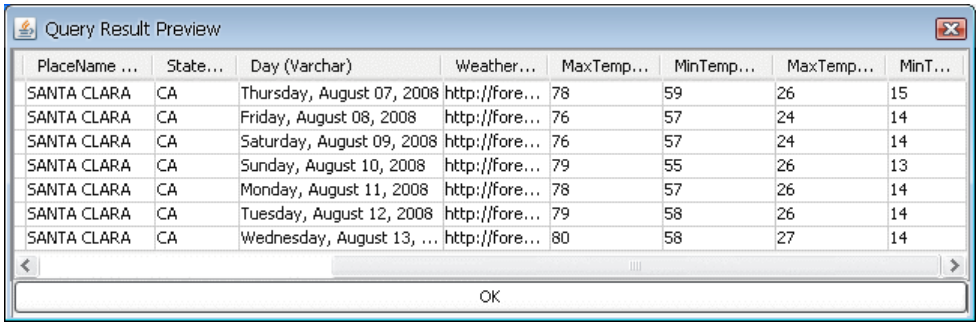
The first option in the dialog allows you to specify a display name in the Data Source Manager. Next, there are three drop-down menus in the dialog. The first drop-down menu contains all the SOAP services described in the WSDL file. Once you specify a service, the second drop-down list will be populated 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.



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 4.3 - 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.



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.

4.8. Data from Salesforce

The Salesforce data source is designed for existing Salesforce users who want to display their Salesforce data in EspressChart. 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 EspressChart 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:

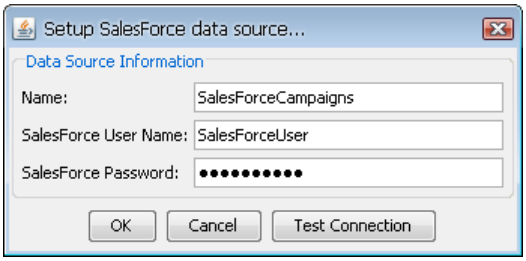
SOQL queries

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

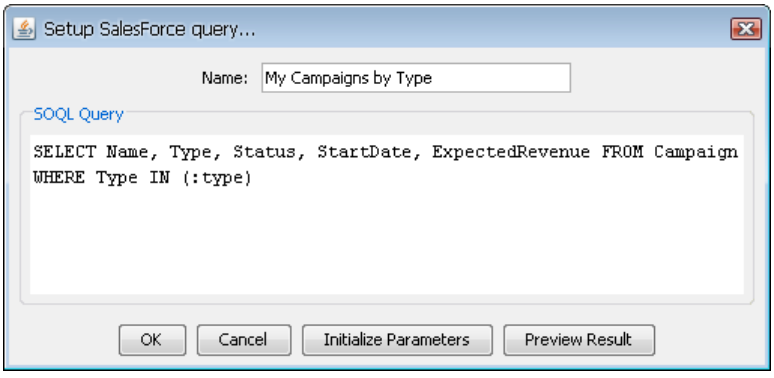
https://help.salesforce.com/s/articleView?id=sf.security_networkaccess.htm

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.



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.



Setup SalesForce Query Dialog

Please note that only child-to-parent relationship queries are supported in the current EspressChart 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
```

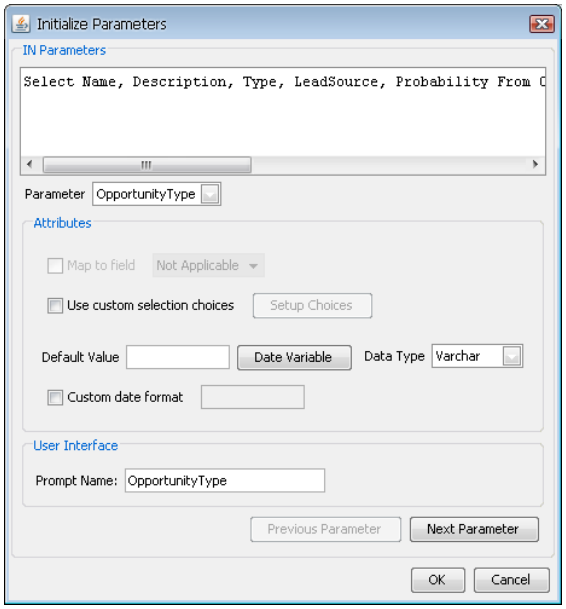
specifies a single value parameter called *CampaignName*. You would then be able to enter a campaign name at run-time and only retrieve data for that campaign.

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.



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.

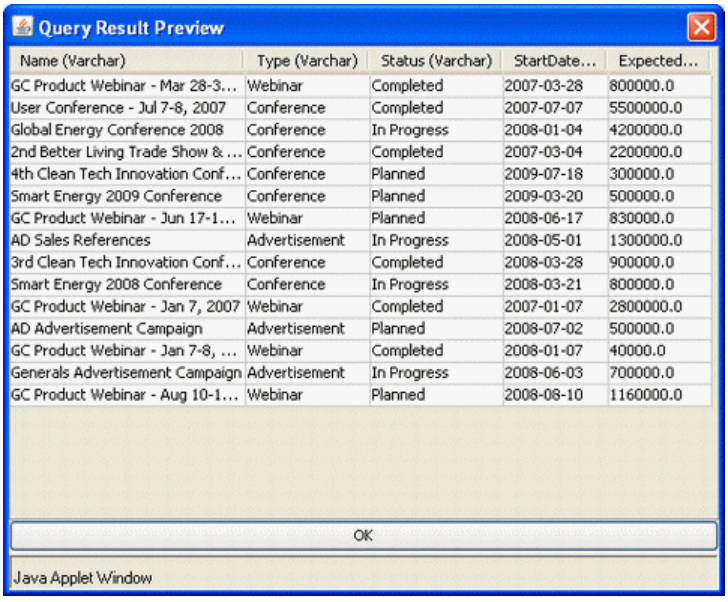
Use custom selection choices: Rather than having a drop-down menu with all the distinct column values, you can build a custom list of parameter values. To set up the list, select this

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 4.2.2.2.2 - 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



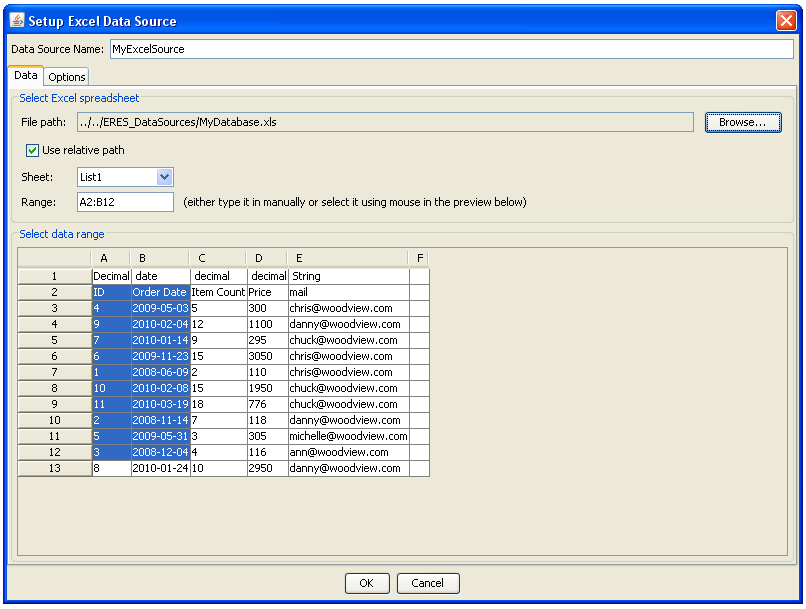
Query Result Preview Dialog

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.

4.9. Data from Excel files

EspressChart also allows you to design charts 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.



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 EspressChart 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 EspressChart 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.

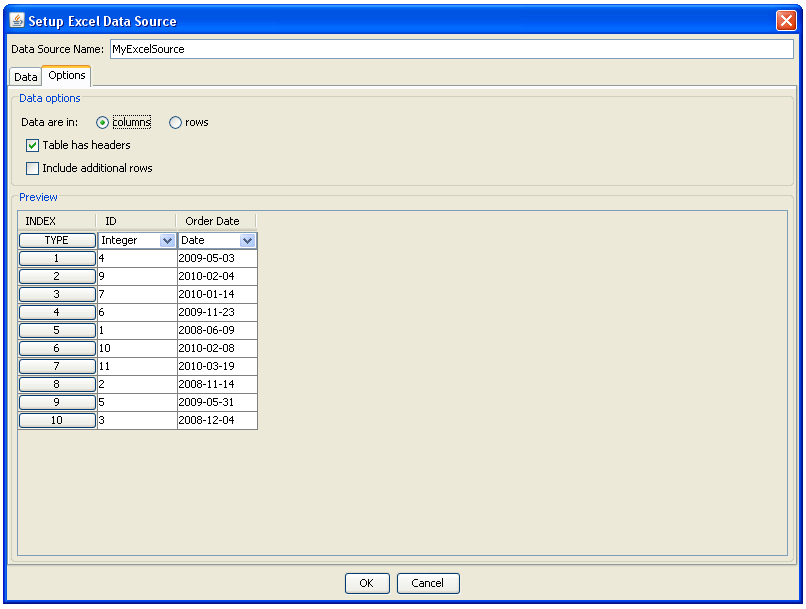
Please note that EspressChart can process both `*.xls` files and `*.xlsx` files. The `*.xlsx` files are used in Microsoft Excel 2007 and newer and they are based on Open XML.

EspressChart 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 dialog - Options

Click *OK* to save the data source when the configuration is finished.

4.10. Using Data for Charts

4.10.1. 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.

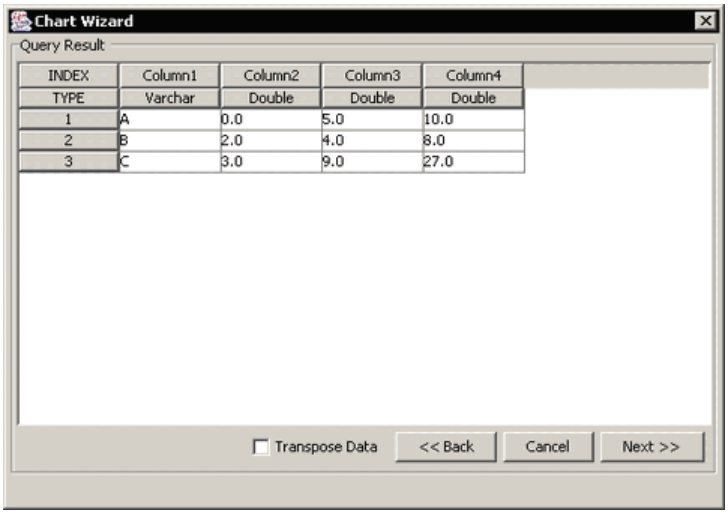
EspressChart allows you to construct charts from multiple data sources. Once you have selected your first source, and click *Next* from the data table window, you will be presented with a dialog asking if you would like to process the current data or select another data source.

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.

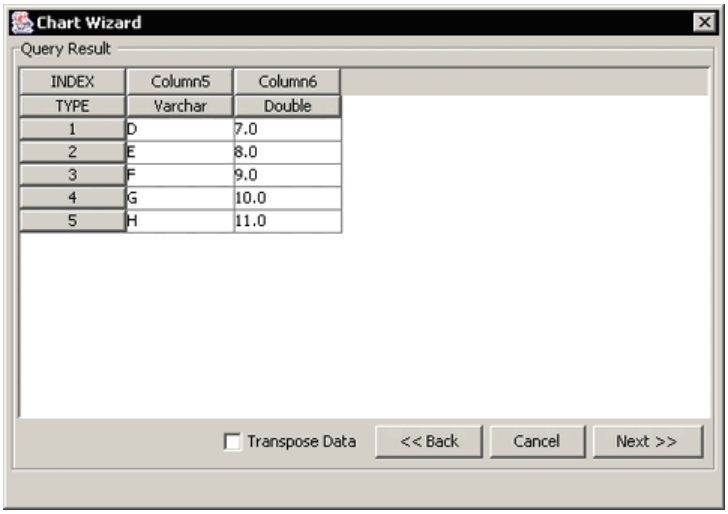
Multiple data sources are combined sideways in the data table. This means that the columns from the second (or third, fourth, etc.) data sources are placed to the right next to the columns from the first data source. If the columns from one data source have more rows than the other ones, null values will be placed in extra rows.

For example, assuming you want to use two data sources to create a chart. The data table generated by the first source will look like this:



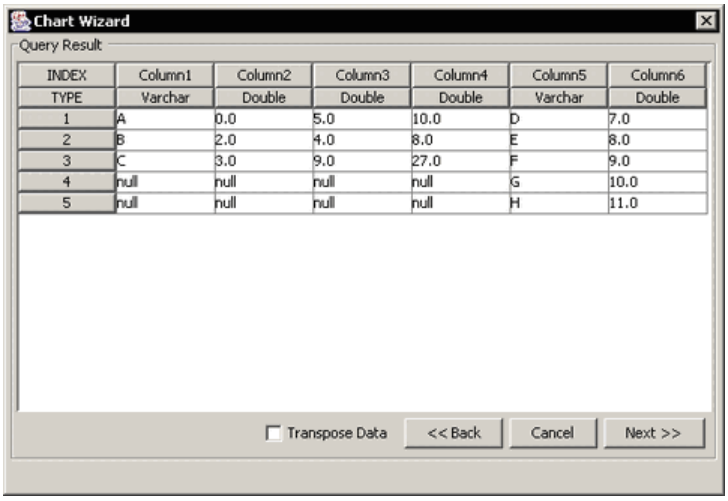
Data From the First Data Source

The data from the second source will look like this:



Data From the Second Data Source

When you combine two data sources, you will get the following data table:



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.

4.10.2. Change Data Source

At any point during chart design, you can select to change the template's data source. Since chart templates are saved with their data source information, you must use the option within Chart 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 4.11 - Data Source Updating). To change a template's data source, select *Modify Data Source* from the Data menu, or click the *Modify 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.

When changing the data source, you might need to re-map the data to the chart. For more information about layouts and data mapping, see Chapter 5 - Chart Types and Data Mapping.

4.11. Data Source Updating

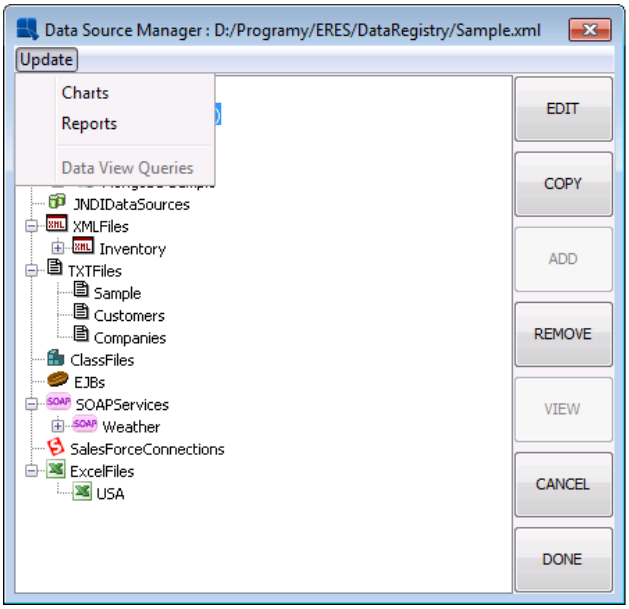
There are many circumstances where you will want to move a group of templates or a complete installation of EspressChart from one location to another. For example an application can move from development to a production environment. In each environment, the location and connection information for data sources may be different. In this scenario, updating chart 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, EspressChart allows you to quickly update a group of templates based on information in the data registry.

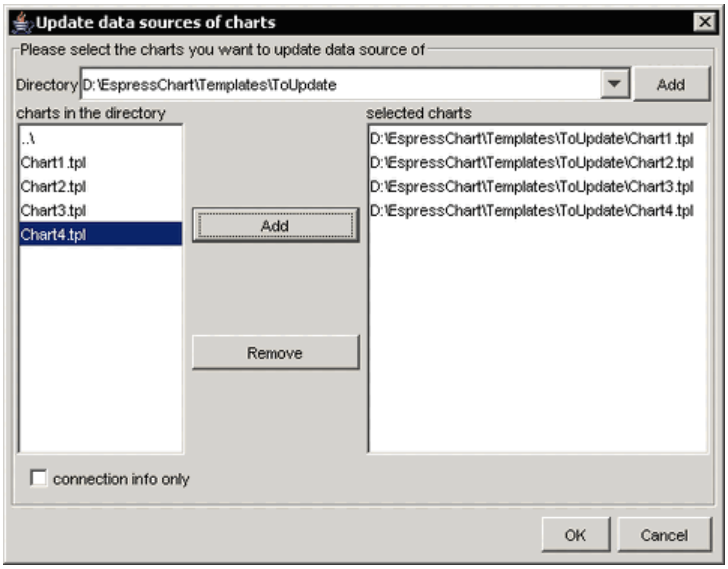
Although chart templates maintain an internal copy of the data source information (allowing them to be deployed independently), they also maintain information (location & source) about the data registry from which they were created. Hence, when you modify a chart's query (as detailed in Section 4.2.4 - 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 4.2.3.2 - Updating Data View Queries.

To use this feature, first make any modifications to the data registry that you want to propagate to the chart 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 charts 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 → Charts. This will bring up a dialog allowing you to select which charts you want to update.





Select charts for Data Source Updating

To select charts to update, first browse to the directory that contains the charts. After you select a directory, any chart 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.

A log file named `UpdateDataSources` will also be written in the root directory of the installation with the contents of the progress screen. Charts that fail the updating for various reasons can be updated manually using the option in Chart Designer.

Note

Only charts for the current data registry will be modified. If you select charts that do not retrieve their data from a source in the current registry, they will be ignored.

4.12. CData JDBC drivers

If you want to connect to a data source driver that doesn't ship with EspressChart, 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

4.12.1. 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>`

4.12.1.1. 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 4.12.2 - CData JDBC driver installation, Section 4.12.3 - Deploying the CData JDBC Driver in EspressChart and Section 4.12.4 - 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).

4.12.1.2. 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 4.12.2 - CData JDBC driver installation, Section 4.12.3 - Deploying the CData JDBC Driver in EspressChart and Section 4.12.4 - 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.

4.12.1.3. 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 4.12.2 - CData JDBC driver installation, Section 4.12.3 - Deploying the CData JDBC Driver in EspressChart and Section 4.12.4 - Using the CData JDBC drivers in DataSource Manager

4.12.1.4. 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 4.12.2 - CData JDBC driver installation, Section 4.12.3 - Deploying the CData JDBC Driver in EspressChart and Section 4.12.4 - Using the CData JDBC drivers in DataSource Manager

4.12.1.5. 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 4.12.2 - CData JDBC driver installation, Section 4.12.3 - Deploying the CData JDBC Driver in EspressChart and Section 4.12.4 - Using the CData JDBC drivers in DataSource Manager

4.12.1.6. 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

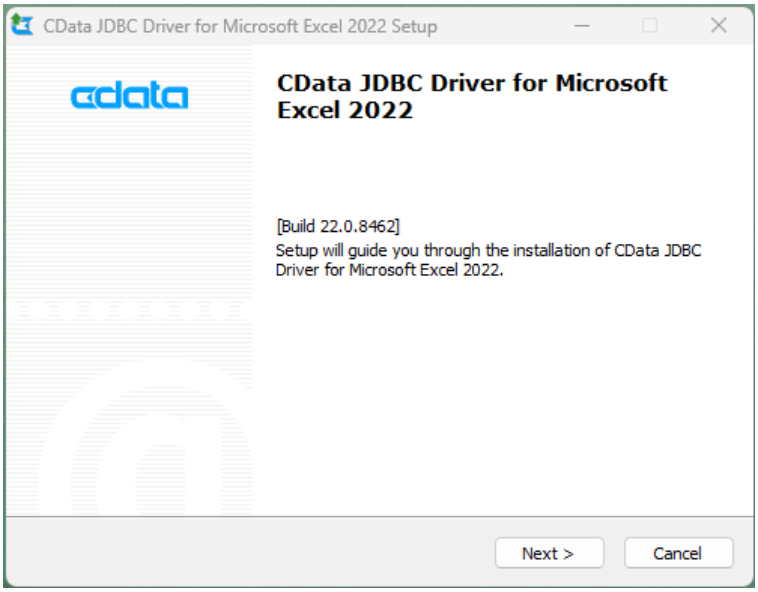
After downloading the driver, see the following chapters: Section 4.12.2 - CData JDBC driver installation, Section 4.12.3 - Deploying the CData JDBC Driver in EspressChart and Section 4.12.4 - Using the CData JDBC drivers in DataSource Manager

4.12.2. 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.

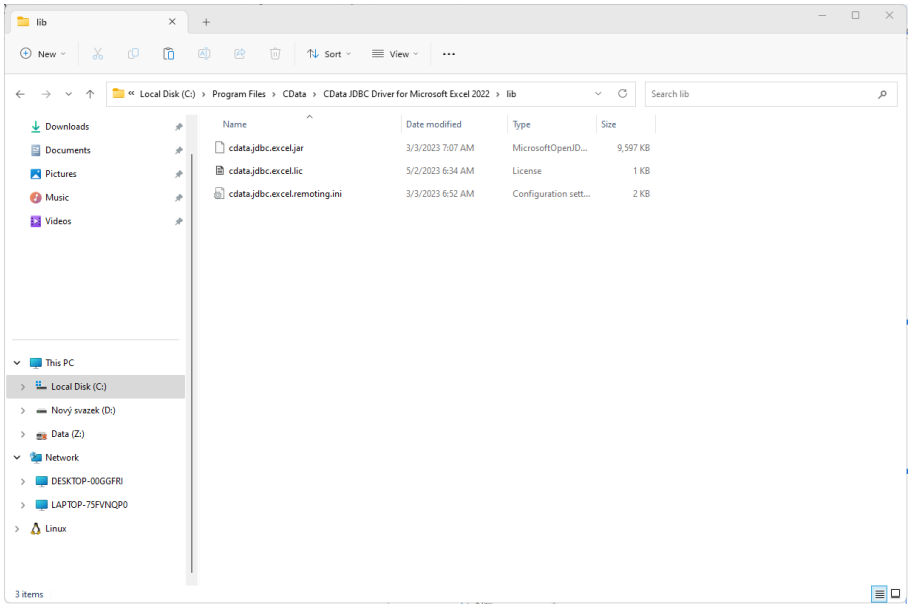
First, install the CData JDBC Driver of your choice. You will find the links to each supported CData JDBC driver in the chapters below.

After downloading the installer, proceed with the setup according to the dialogs.



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".

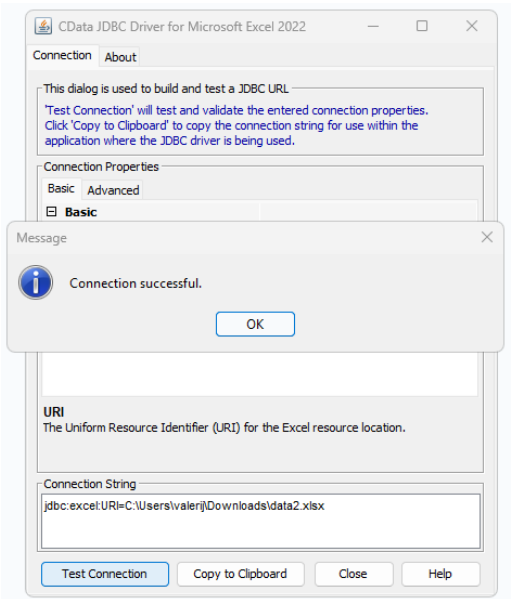


Tip

Executing "cdata.jdbc.excel.jar" will launch the connection test tool. You can use this tool to test or troubleshoot the CData driver.

Tip

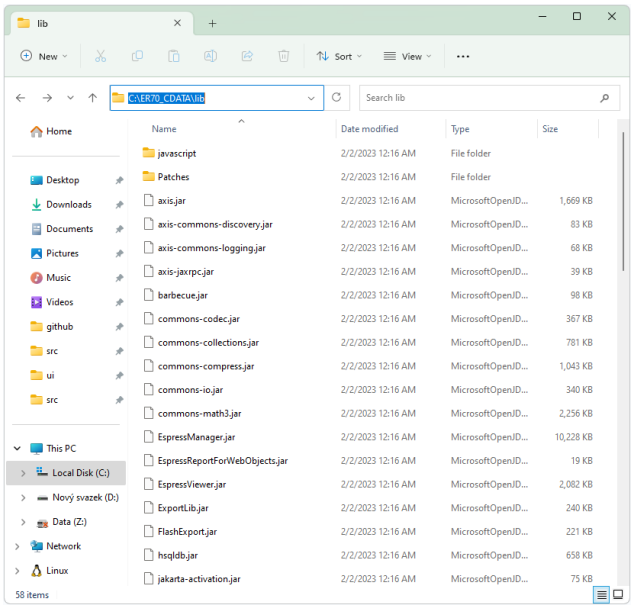
The connection test tool also displays you the connection URL that can be used in EspressChart Data Source Manager.



4.12.3. Deploying the CData JDBC Driver in Espress-Chart

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 EC/lib/

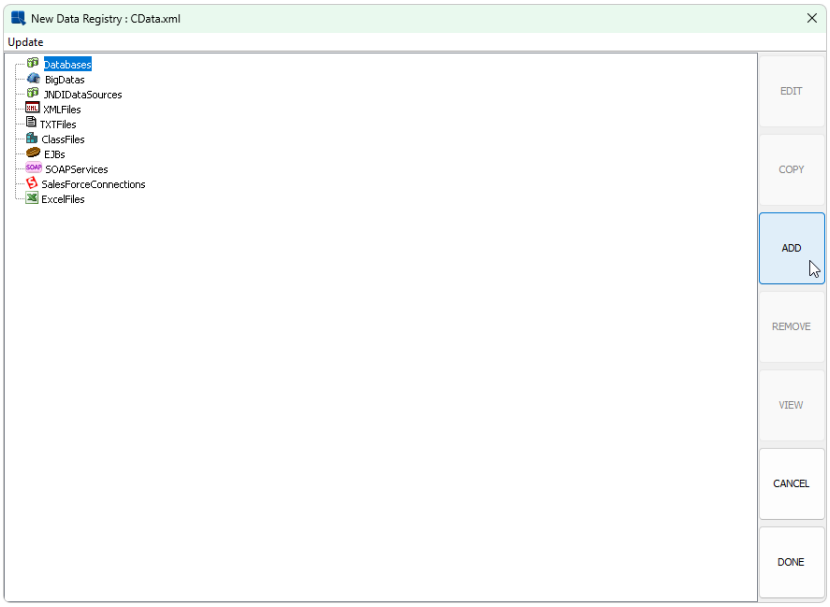


After you've copied the files, restart The EspressManager (if it is running).

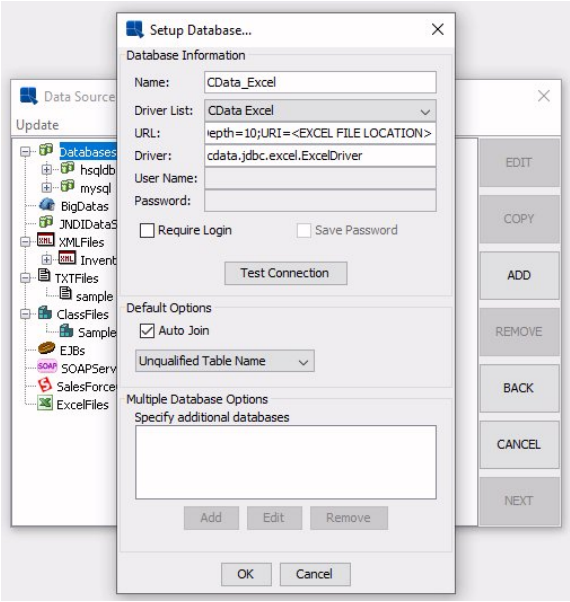
4.12.4. Using the CData JDBC drivers in DataSource Manager

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.



In the “Driver List:” drop-down menu, select “CData Excel” (or any other CData JDBC driver you might be installing).



In the “URL:” text field, replace the placeholders (like "EXCEL FILE LOCATION") with real values.

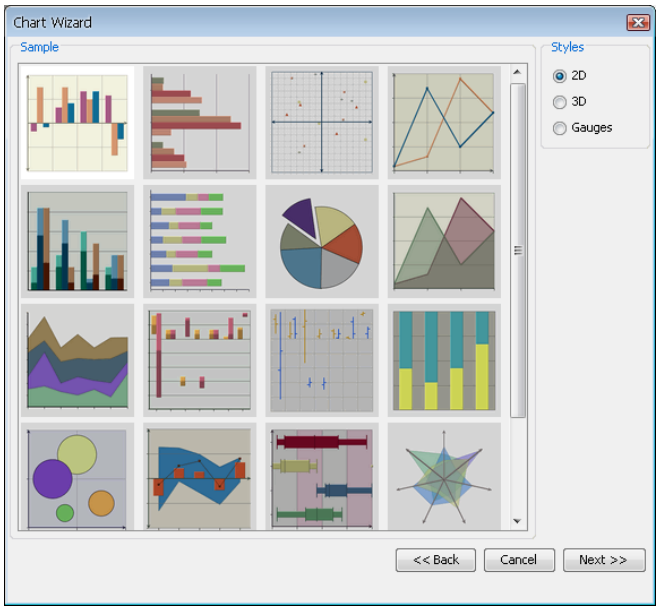
Tip

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.

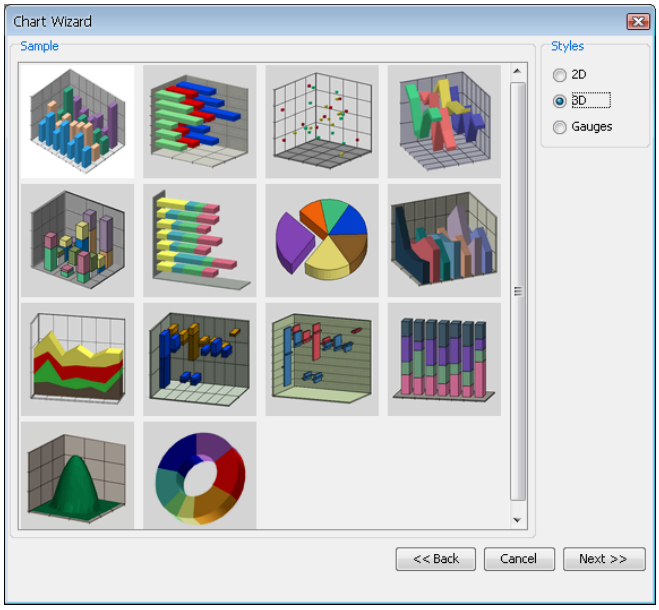
Chapter 5. Chart Types and Data Mapping

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:

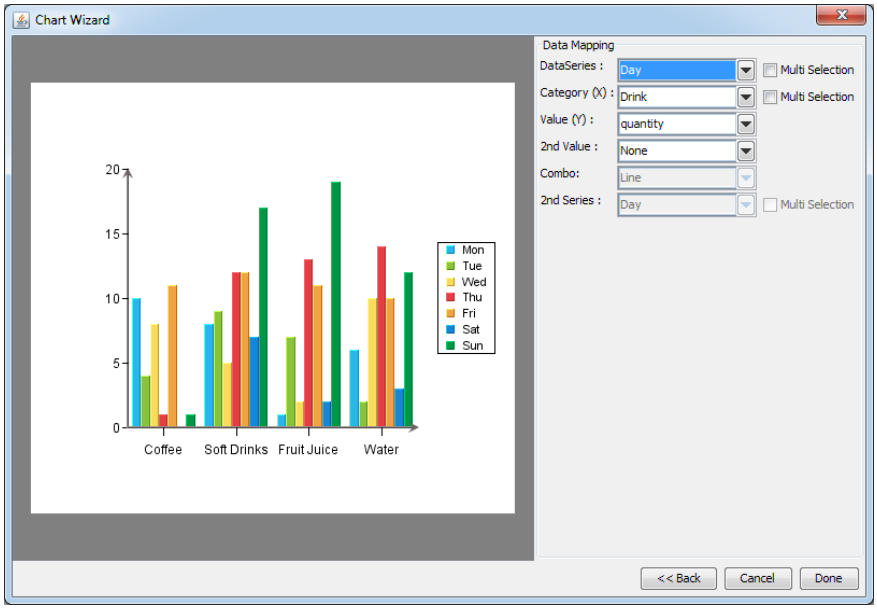
- Column
- Bar
- XY(Z) Scatter
- Line
- Stack Column
- Stack Bar
- Pie
- Area
- Stack Area
- High Low
- HLCO
- Percentage Column
- Doughnut
- Surface (Three-Dimensional Only)
- Bubble (Two-Dimensional Only)
- Overlay (Two-Dimensional Only)
- Box (Two-Dimensional Only)
- Radar (Two-Dimensional Only)
- 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 5.2 - Column Charts

For information on gauges, see Section 5.20.2 - Gauges

5.1. 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 - 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 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, EspressChart 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 5.2 - Column Charts.

5.1.1. Data Transposition

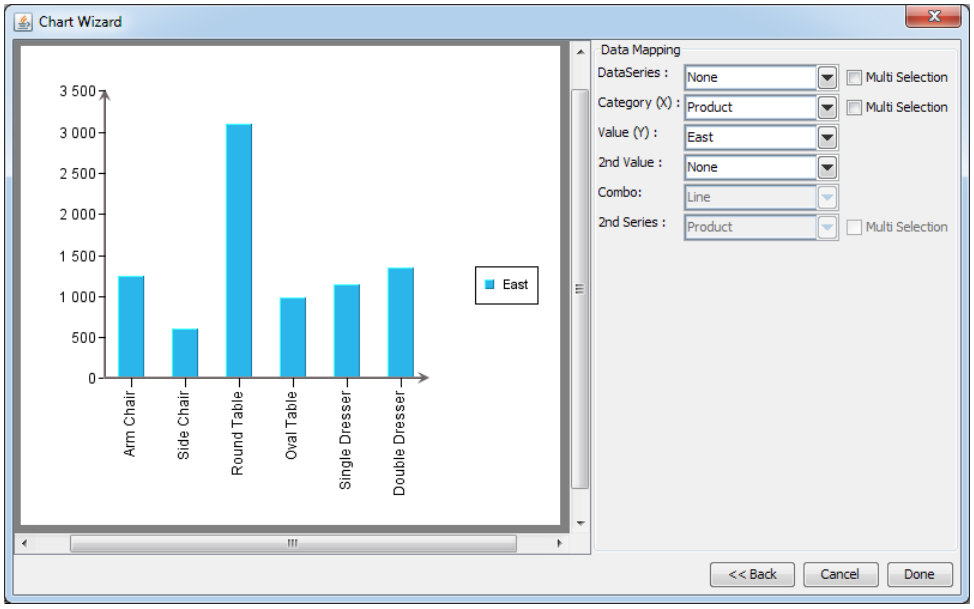
You can also set data transposing from the data mapping dialog. Transposing allows you to plot multiple data source columns in one chart without modifying the data source.

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

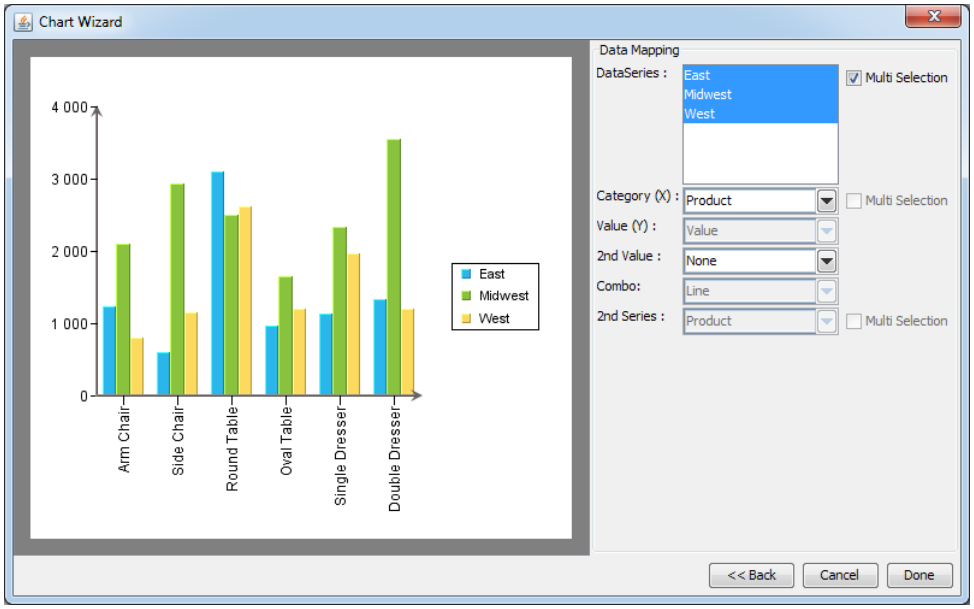
Example Data Source

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.



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:



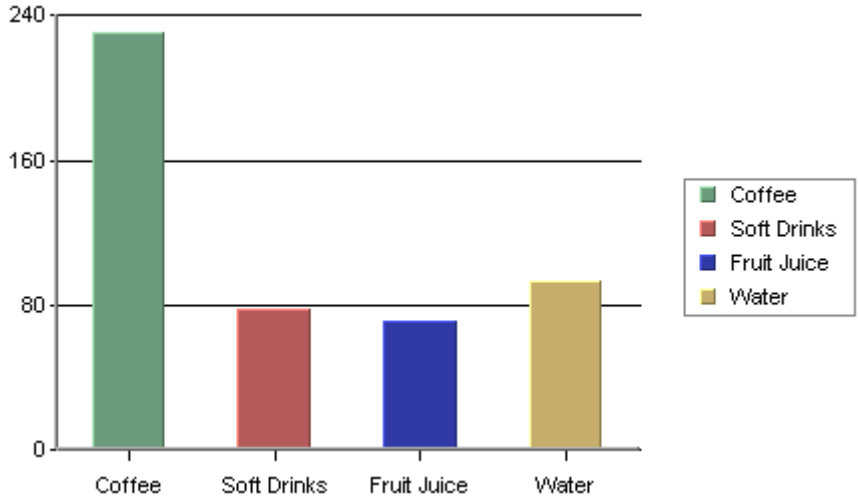
Default Chart Data Mapping

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.

5.2. Column Charts

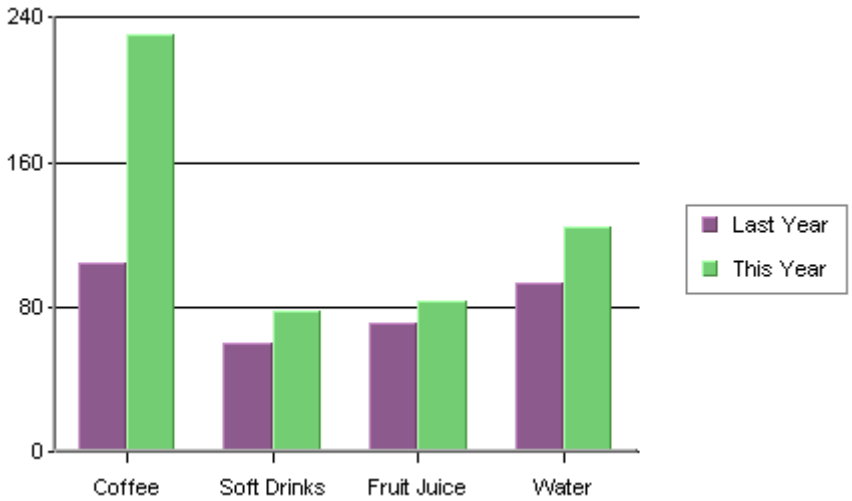


Column Chart (Without Data Series)

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.

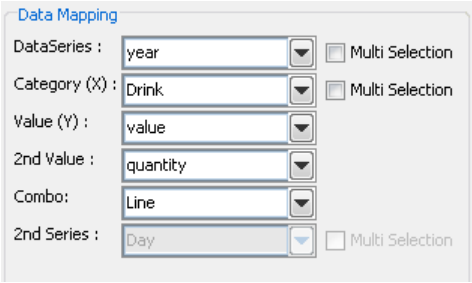
In the examples given, we have chosen Drink as the category variable and Value as the value variable (most examples use the `Sample.dat` file included in the EspressChart installation). Year has been selected as the data series variable for the chart below. Therefore, each name has two columns shown: 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

5.2.1. Data Mapping

Data mapping for column charts is fairly straightforward. It is similar to the examples first presented in Section 3.2 - Basic Data Mapping. For column charts, the following options are available:



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.

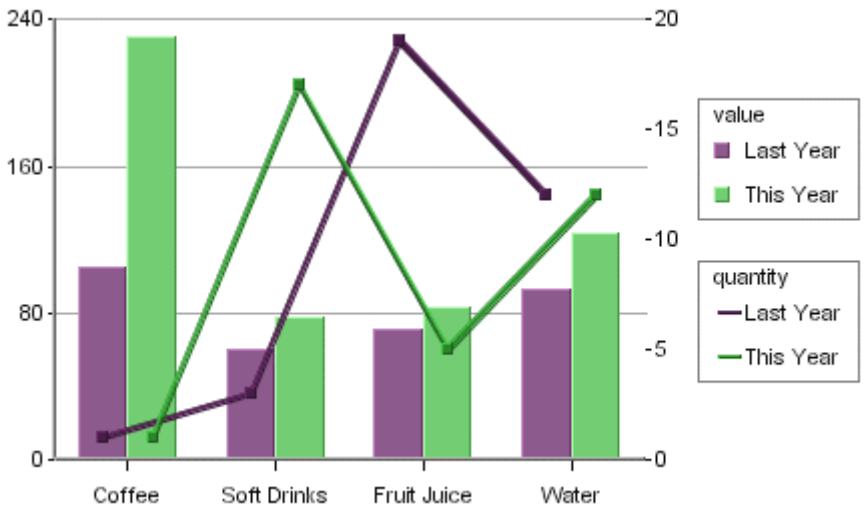
2nd value: Add a second value to create a combination chart.

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 5.1.1 - Data Transposition.

The last three data mapping options allow you to add a second value to the chart. EspressChart 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.

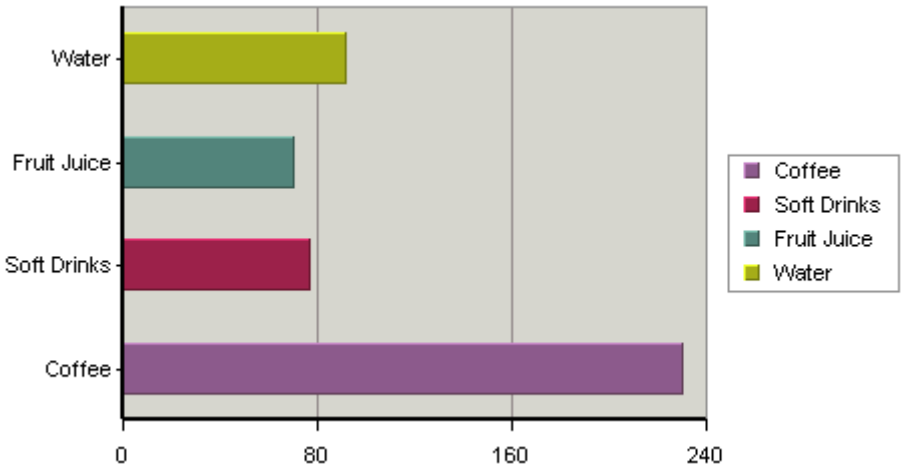


Column Chart With 2nd Value

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 5.17 - Overlay Charts.

5.3. Bar Charts

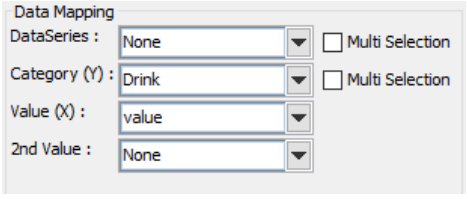


Bar Chart

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.

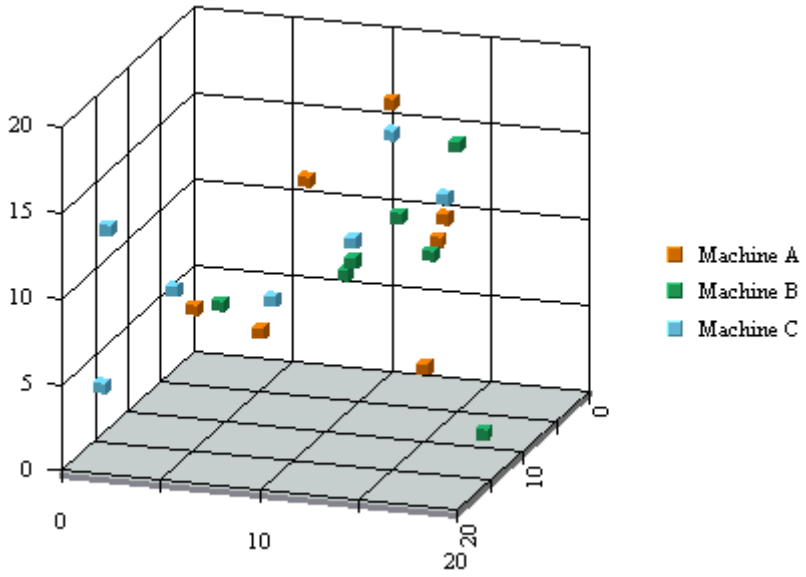
5.3.1. Data Mapping



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.

5.4. XY(Z) Scatter Charts

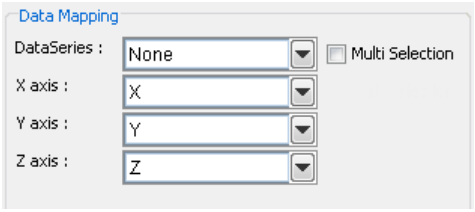


XY(Z) Scatter Chart

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.

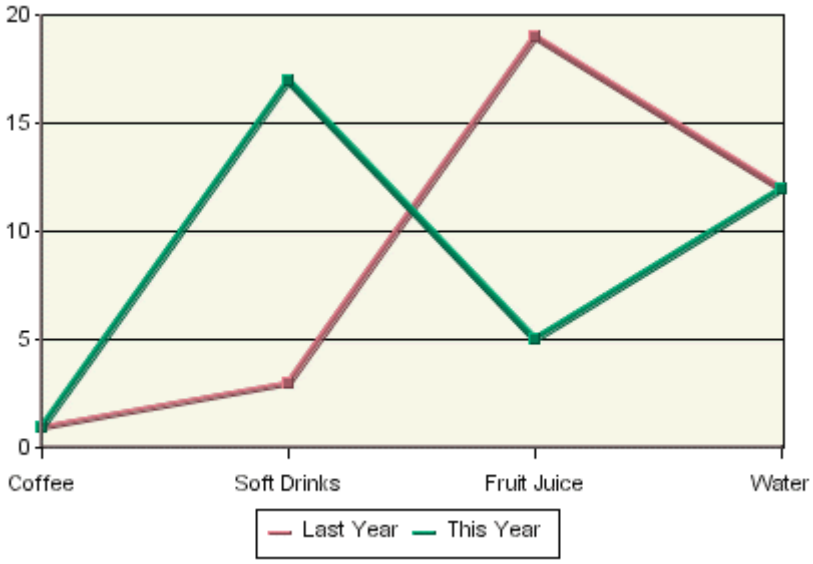
5.4.1. Data Mapping



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).

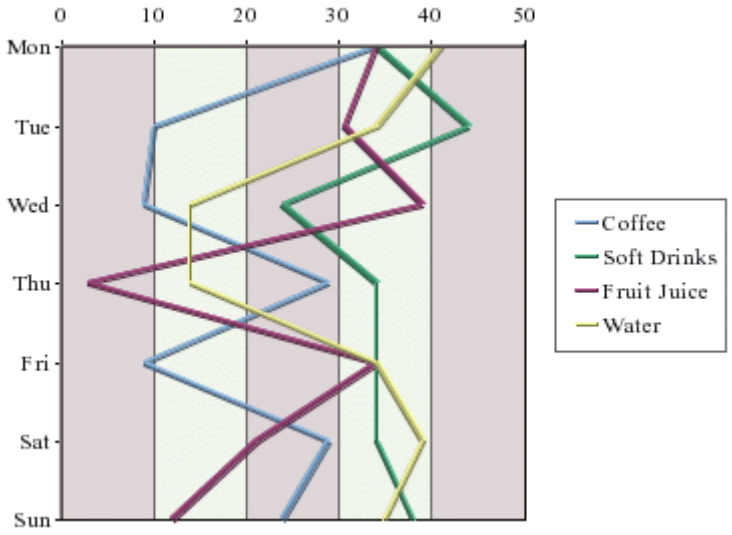
5.5. Line Charts



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.

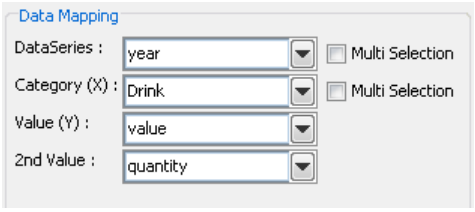
EspressChart allows 2D line charts to be displayed vertically in addition to the default horizontal setting. To use this feature in Chart Designer, create a line chart and then select Format → Chart Options. Next, select *Vertical*. The chart is rotated clockwise 90 degrees.



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.

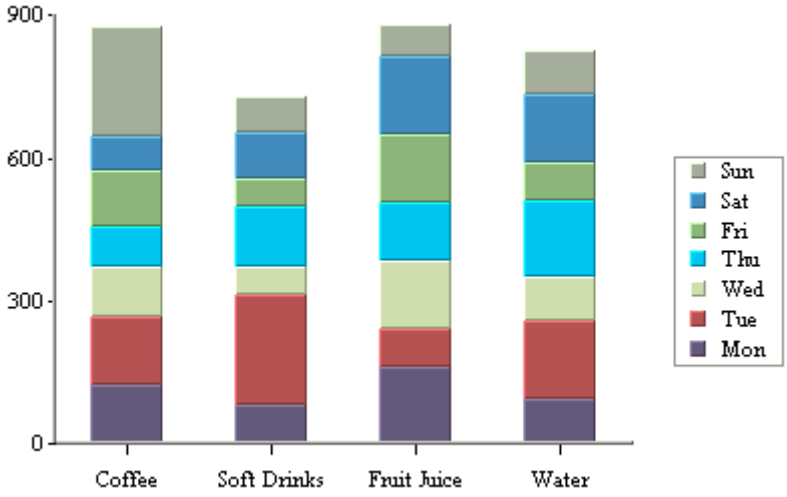
5.5.1. Data Mapping



Mapping Options for Line Charts

Data mapping for line charts is almost exactly the same as for column charts (discussed in Section 5.2.1 - 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.

5.6. Stack Column Charts



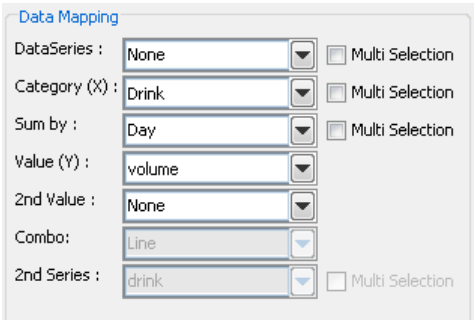
Stack Column Chart

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.

5.6.1. 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.



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.

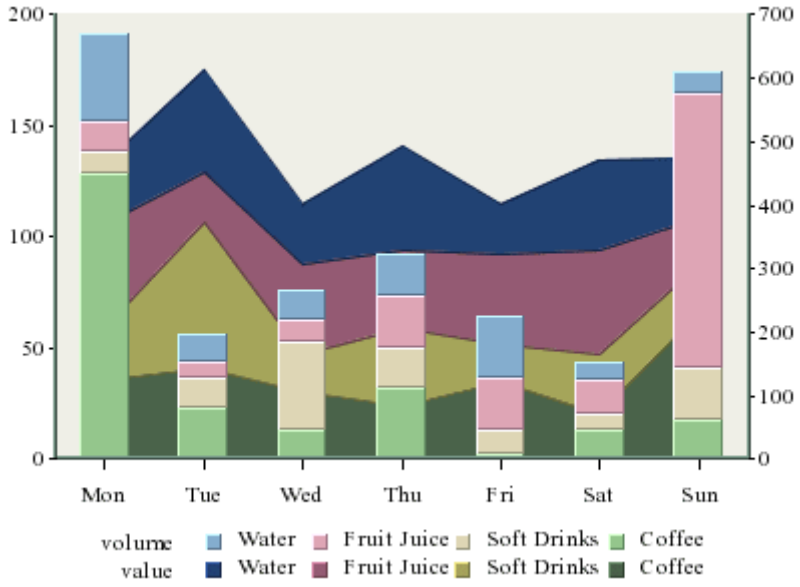
2nd value: Add a second value to create a combination chart.

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 5.1.1 - Data Transposition.

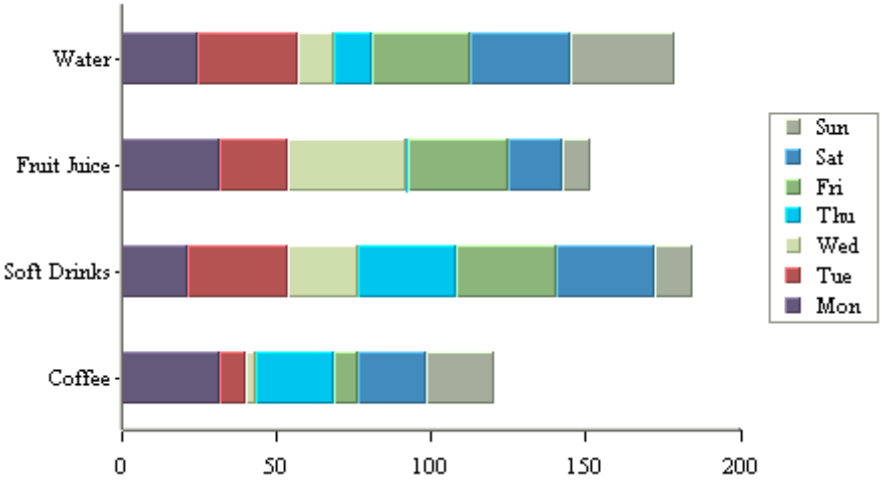
Stack column charts have a unique combination option which allows users to plot a second value as a stack area chart. Both values share the same category and sum-by values. This chart can compare component based categories for different values in the data.



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.

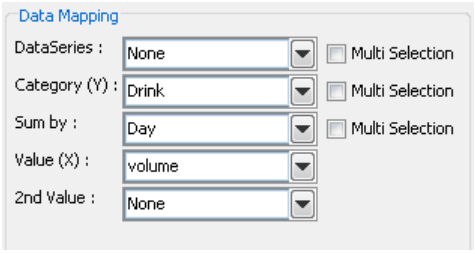
5.7. Stack Bar Charts



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.

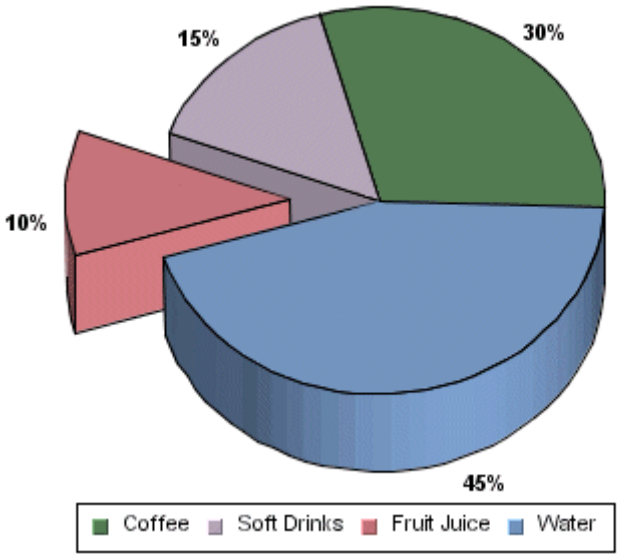
5.7.1. Data Mapping



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.

5.8. Pie Charts

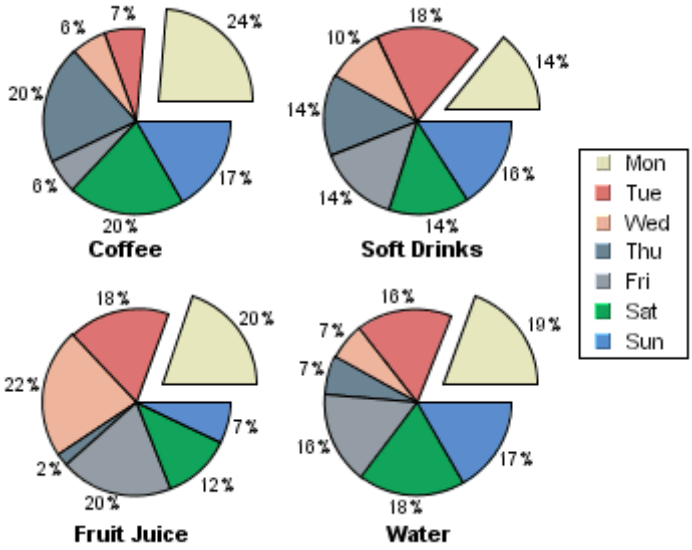


Pie Chart

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.

5.8.1. Data Mapping

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.

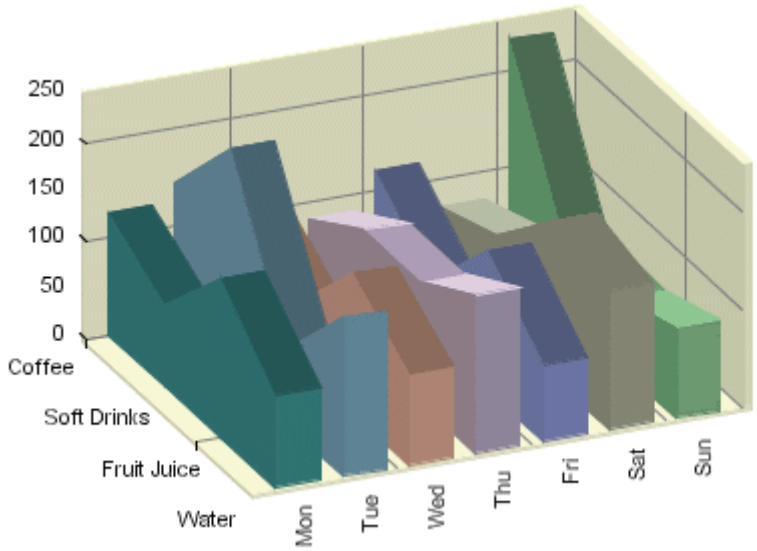
Category (X): Allows you to choose a data column whose distinct values determine the various categories.

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 5.1.1 - Data Transposition.

5.9. Area Charts



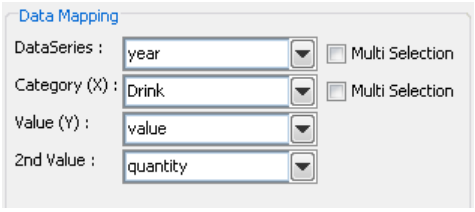
Three-Dimensional Area Chart

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 6.8.3 - Data Ordering) or set the area translucent (see Section 6.1.3 - Format Menu) as in the example above.

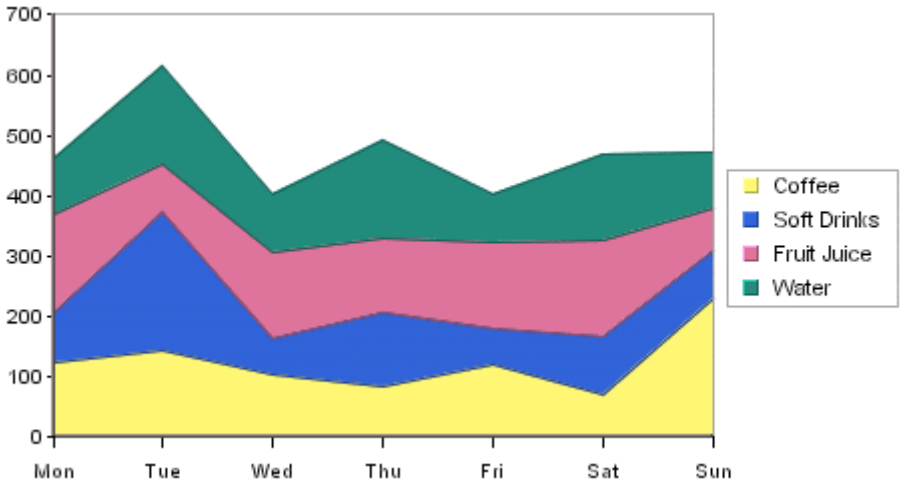
5.9.1. Data Mapping



Mapping Options for Area Charts

Data mapping for area charts is almost exactly the same as for column charts (discussed in Section 5.2.1 - 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.

5.10. Stack Area Charts



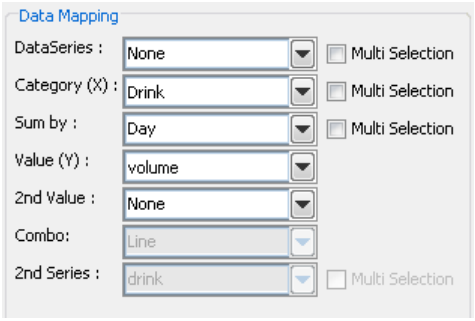
Stack Area Chart

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.

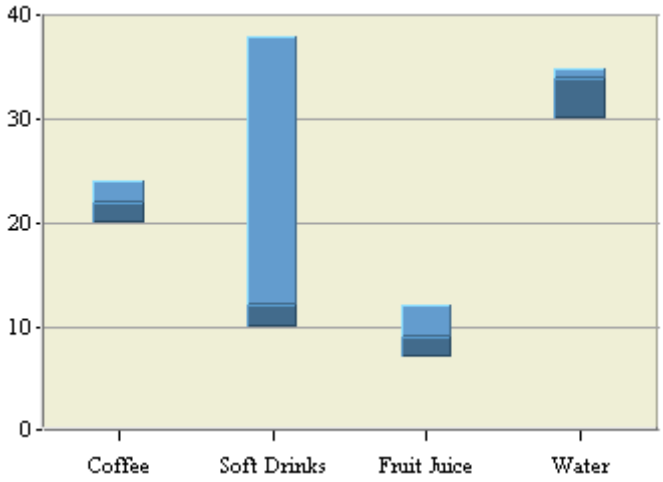
5.10.1. Data Mapping



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 5.6.1 - Data Mapping). However, two-dimensional stack charts cannot have a data series, and the combo options are *Line* and *Overlay*.

5.11. High-Low Charts

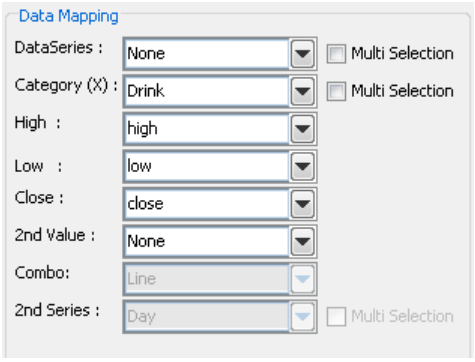


High-Low Chart

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.

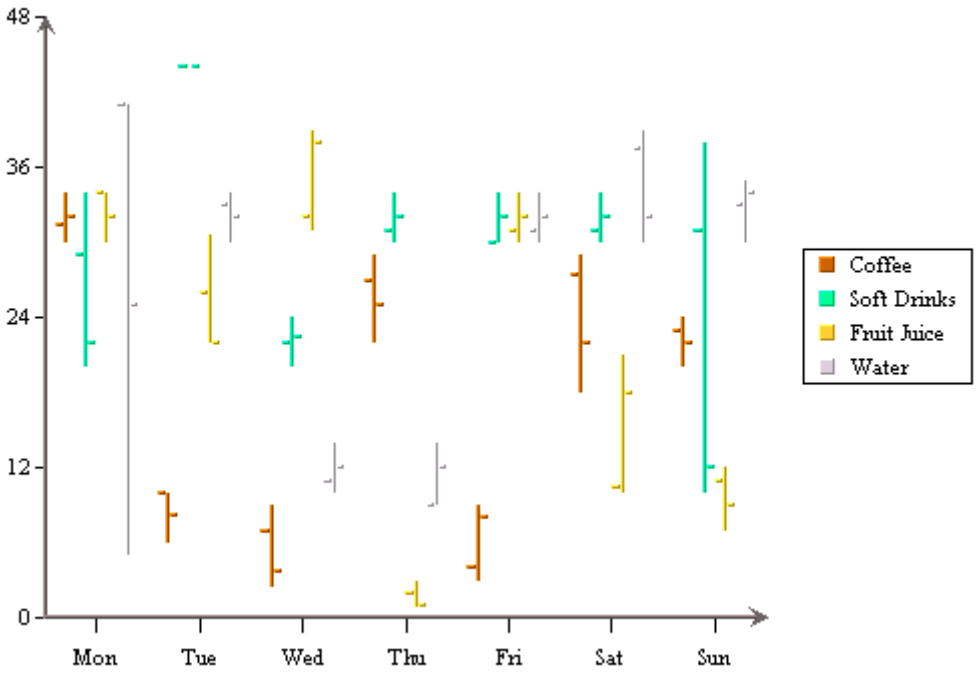
5.11.1. Data Mapping



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*.

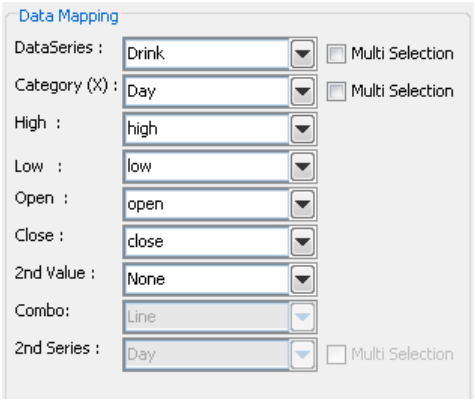
5.12. HLCO Charts



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 Chapter 6 - The Designer Interface.

5.12.1. Data Mapping



Mapping Options for HLCO Charts

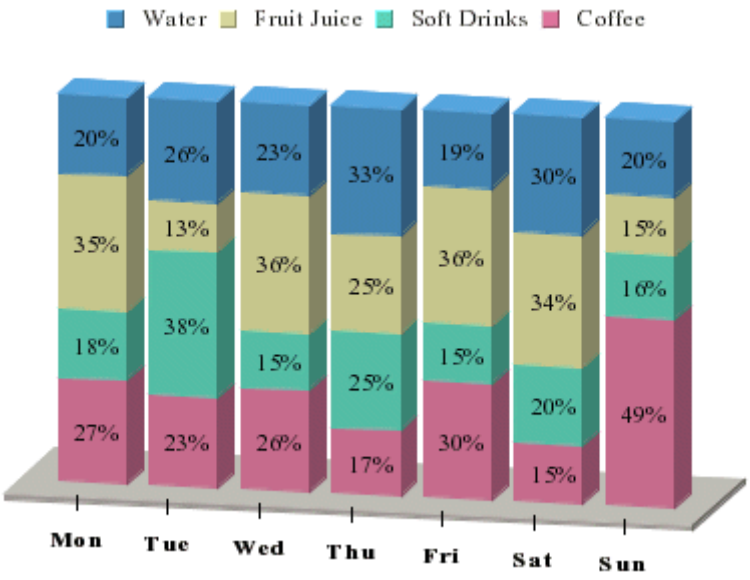
Data mapping for HLCO charts is similar to that for High-Low charts. The only difference is that instead of specifying two different value columns, you must specify four different columns - *High*, *Low*, *Open* and *Close*. Combo options are *Line*, *Column* and *Overlay*.

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 EspressChart.

HLCO-Column Combination Chart

This example plots stock price and volume data over a three month period.

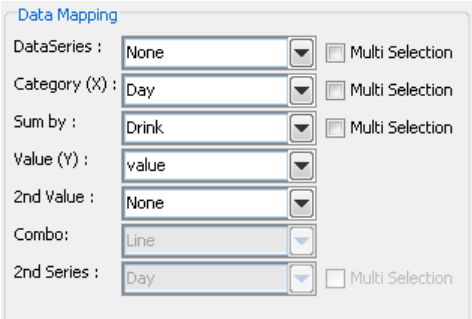
5.13. Percentage Column Charts



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.

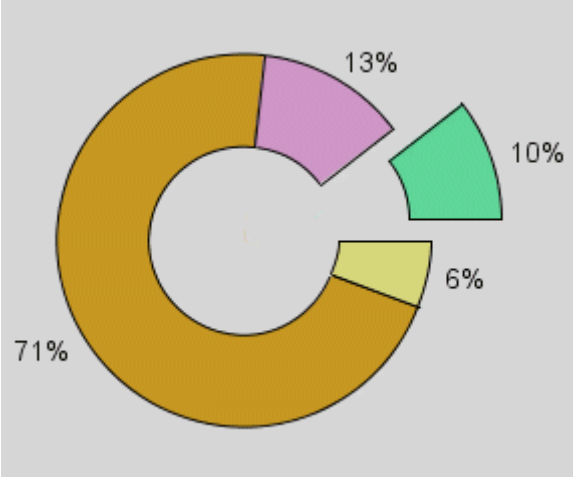
5.13.1. Data Mapping



Mapping Options for Percentage Column Charts

Data mapping for percentage column charts is the same as for stack column charts (covered in Section 5.6.1 - 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*.

5.14. Doughnut Charts



Doughnut Chart

A doughnut chart is the same as a pie chart, except for the “hole” in the center.

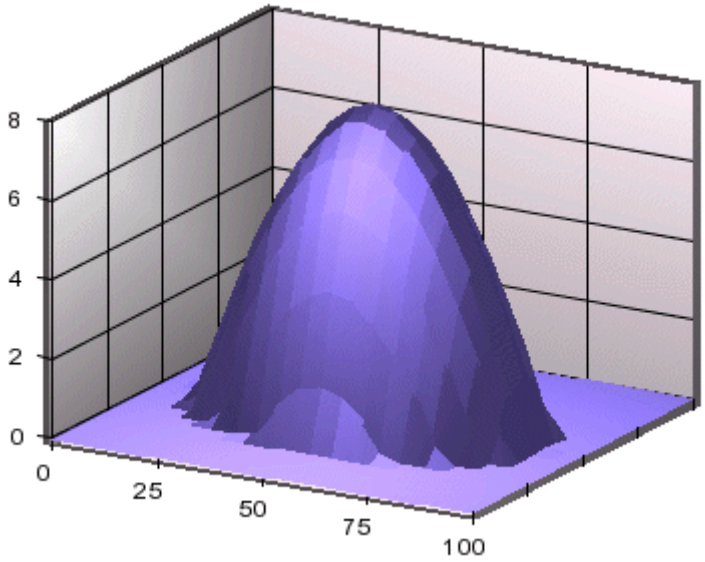
Just as for a pie chart, if a data series is selected, a separate doughnut will be drawn for each category element and each doughnut will be comprised of the series elements. When a series is present, all the separate doughnuts are drawn on a single plot area and resized in proportion with the plot. You have an option to either stack the different doughnuts or draw them in a line.

5.14.1. Data Mapping

Mapping Options for Doughnut Charts

Data mapping for doughnut charts is exactly the same as for pie charts (discussed in section Section 5.8.1 - Data Mapping).

5.15. 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.

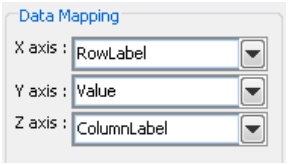
Sample surface chart data:

	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.

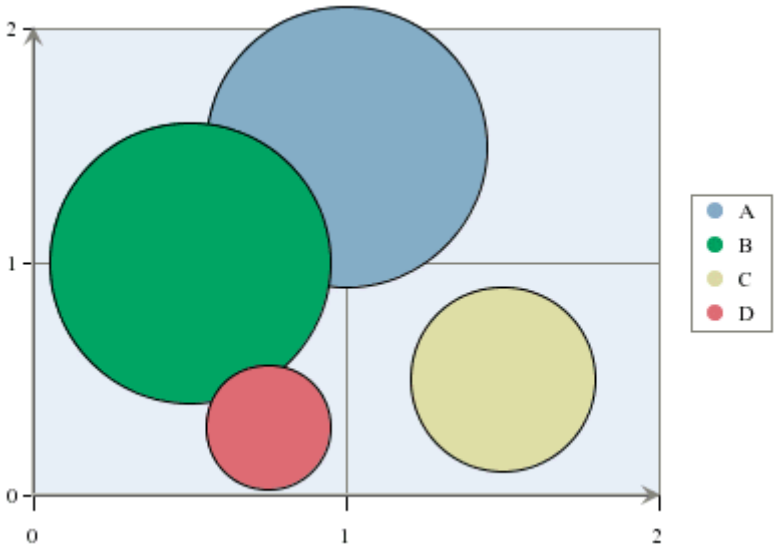
5.15.1. Data Mapping



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.

5.16. 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 6.9.1 - Bubble Charts.

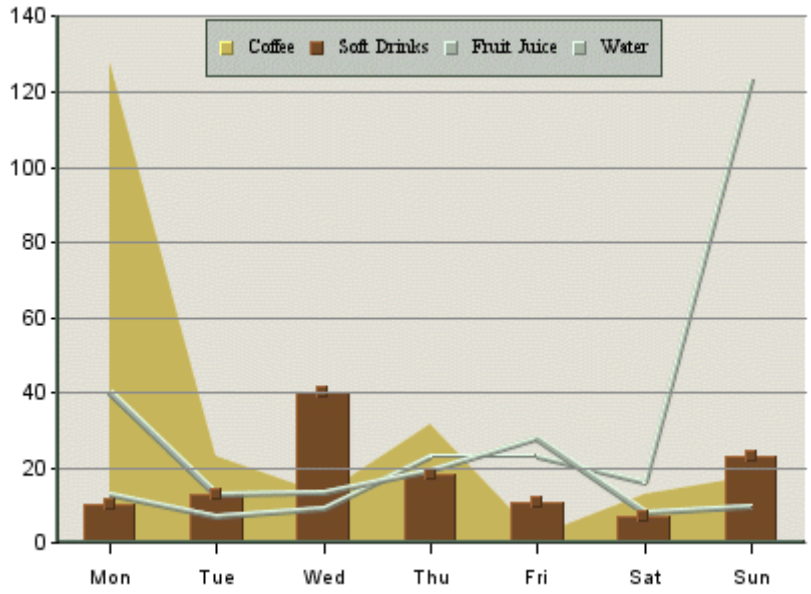
5.16.1. Data Mapping

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).

Mapping Options for Bubble Charts

Note that similarly to a scatter chart, the columns have to be numeric in order to be mapped successfully.

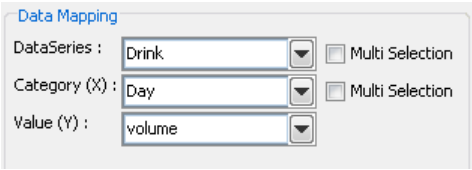
5.17. Overlay Charts



Overlay Chart

EspressChart 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.

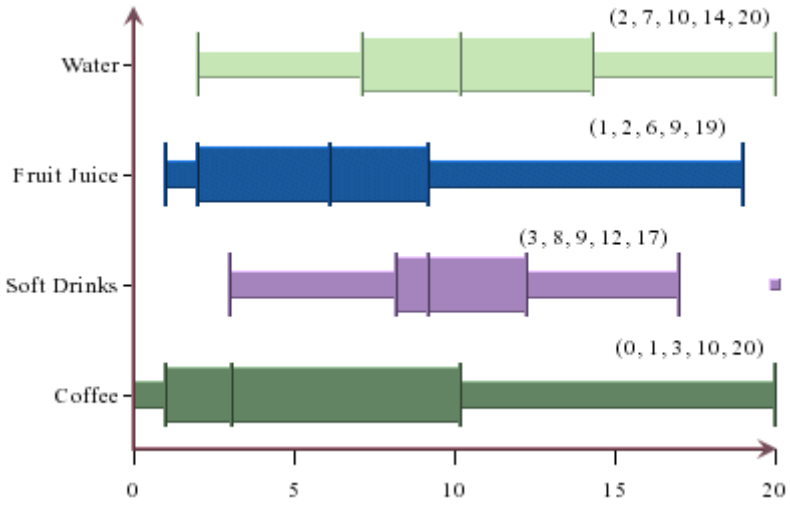
5.17.1. Data Mapping



Mapping Options for Overlay Charts

The data mapping for an overlay chart is similar to a column chart (covered in Section 5.2.1 - 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 6.9.3 - Overlay Charts.

5.18. Box Charts

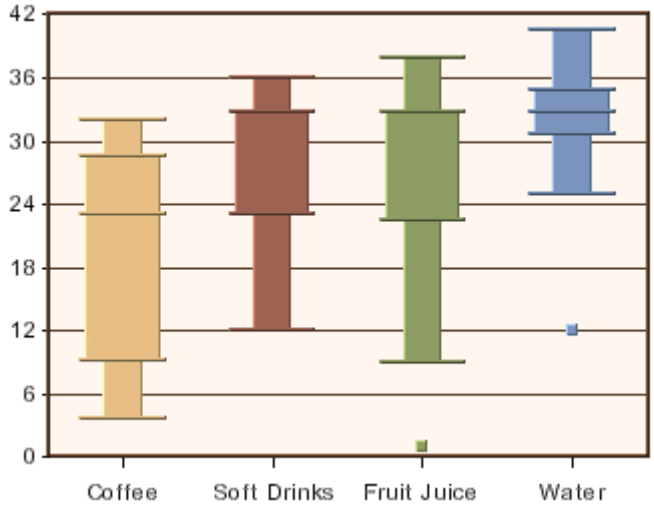


Box Chart

EspressChart 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.

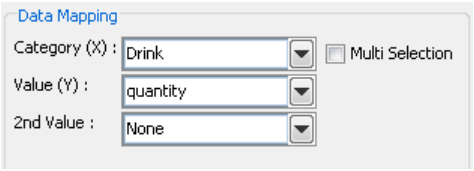
EspressChart allows box charts to be displayed vertically in addition to the default horizontal setting. To use this feature in Chart Designer, create a box chart and then select Format → Chart Options, and select *Vertical*.



Vertical Box Chart

This chart is available in two-dimensional form only.

5.18.1. Data Mapping



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.

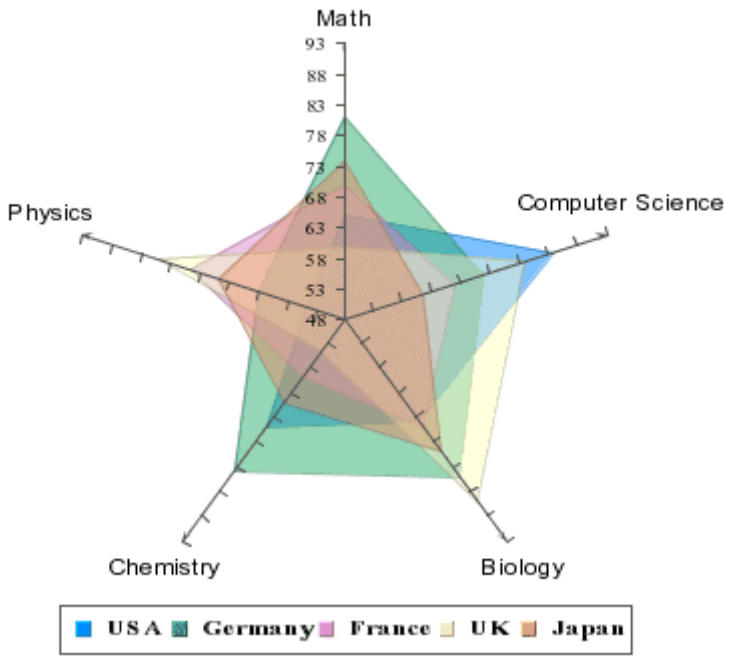
Value (Y): Allows you to choose a data column to provide values for each category. These values are used to calculate the minimum data point, the 25th percentile, the median, the 75th percentile, and the maximum data point.

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 5.1.1 - Data Transposition.

5.19. Radar Charts

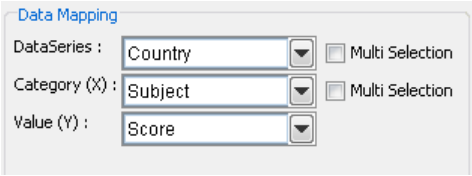


Radar Chart

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.

5.19.1. Data Mapping



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.

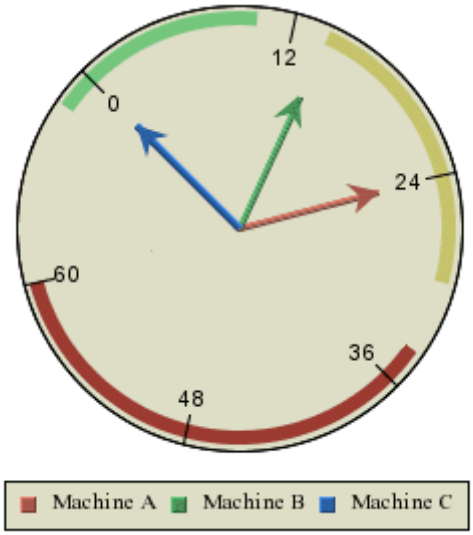
Category (X): Allows you to choose a data column whose distinct values determine the categories. The number of unique elements in this column determines the number of axes in the radar chart.

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 5.1.1 - Data Transposition.

5.20. 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.

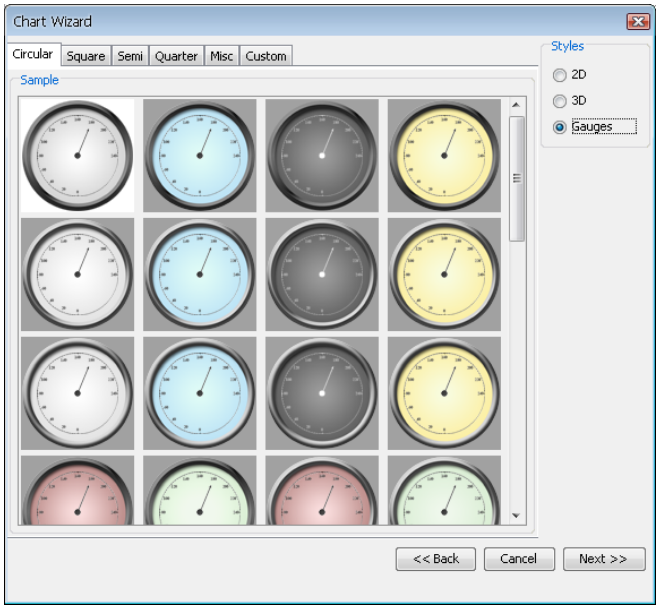
5.20.1. Data Mapping

Mapping Options for Dial Charts

The data mapping of a dial chart is similar to a pie chart (detailed in Section 5.8.1 - Data Mapping). However, instead of pie wedges, the categories become dial hands. Also, dial charts do not support data series or secondary values.

5.20.2. Gauges

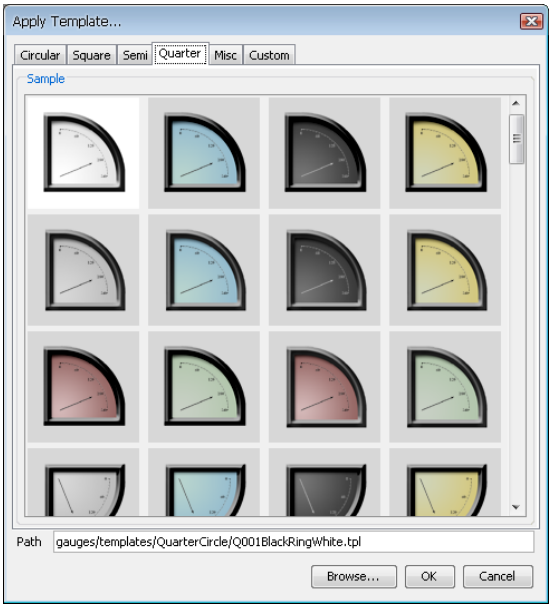
Gauges are special types of dial charts containing a plot background image and/or plot foreground image. To create a new gauge, select the *Gauges* radio button on the *Chart Type Selection* dialog.



Selecting a Gauge Template

You can also create your own gauge by creating a template, saving the tpl file in the `<EspressChart Install>/gauges/templates/Custom/` folder (please note that the `Custom` folder is not created automatically by the EspressChart 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 `<EspressChart Install>/gauges/screenshots/selected/Custom/`, and a dimmer version of the same template placed in `<EspressChart 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.

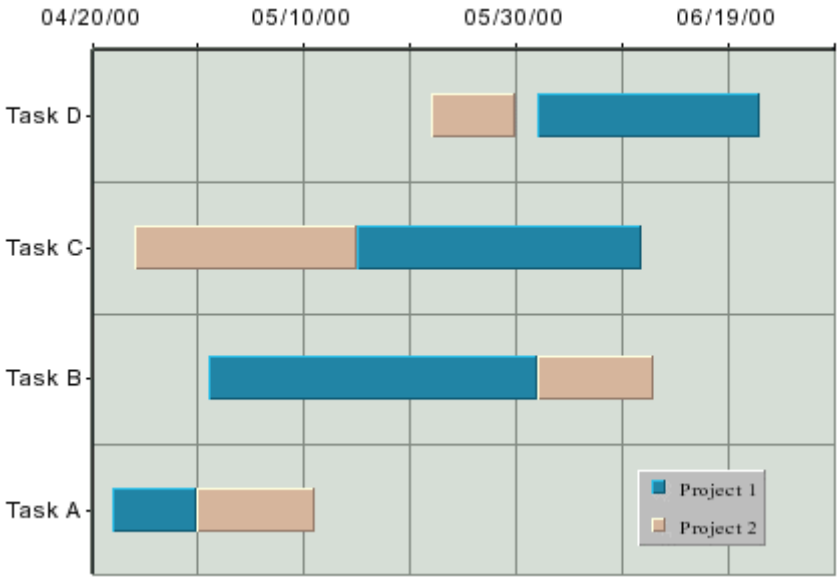
It is also possible to apply a gauge template onto an existing dial chart. Selecting apply template when the current chart is a dial chart will display the gauge tabs just like when creating a new chart.



Applying a Gauge Template

For more information regarding the dial background and foreground images, see Section 6.9.2 - Dial Charts.

5.21. Gantt Charts



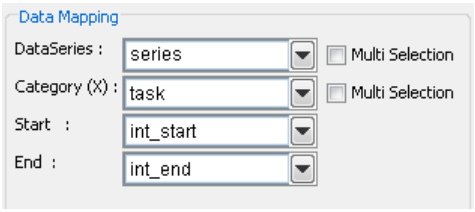
Gantt Chart

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.

5.21.1. 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.



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.

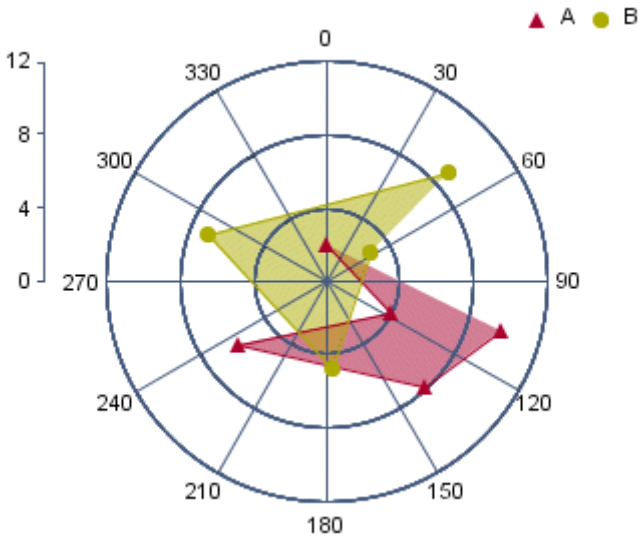
Category (X): Allows you to choose a data column whose distinct values will determine the categories.

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 5.1.1 - Data Transposition.

5.22. Polar Charts



Polar Chart

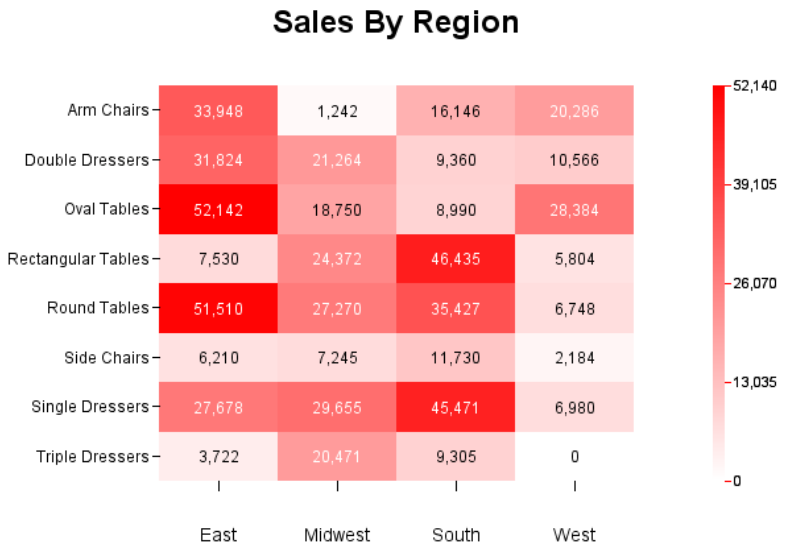
Like a two-dimensional scatter chart, a polar chart plots points on a plane. However, instead of rectangular coordinates, a polar chart plots points using polar coordinates (r, θ), where r represents the distance from the center of the circular plot and θ represents the angle of a ray emanating from 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.

5.22.1. Data Mapping

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.

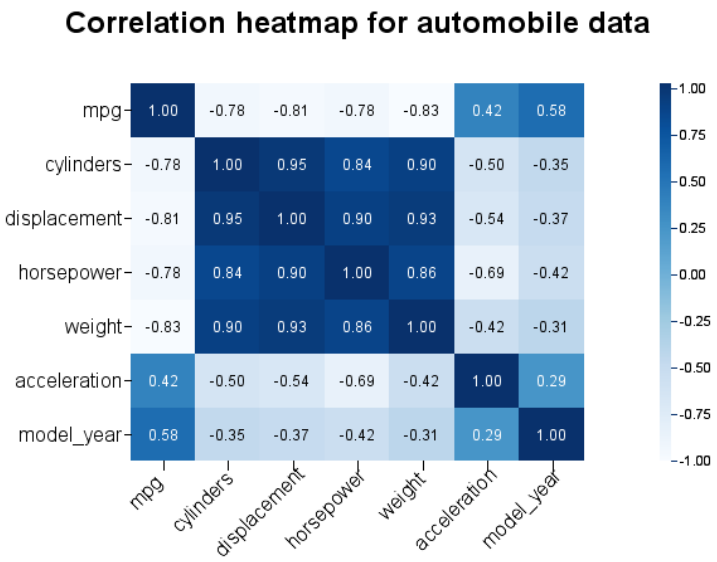
5.23. Heatmap Charts



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.



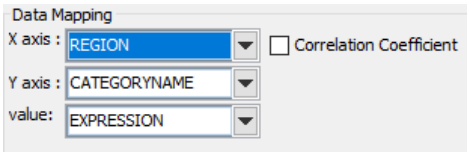
Correlation Heatmap Chart

Note

Heatmap chart is available in a two-dimensional form only.

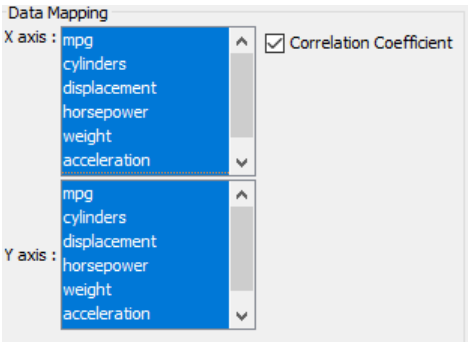
5.23.1. Data Mapping

Data mapping of heatmap without applying the correlation function is straightforward.



Heatmap data mapping

Data mapping of heatmap with correlation function applied.



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.

5.24. 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.

Chapter 6. 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.

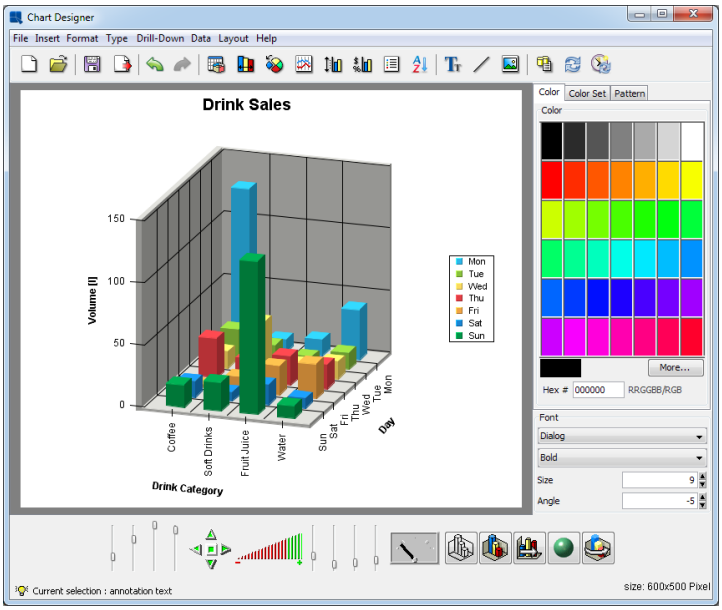


Chart Designer Interface

6.1. 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.

6.1.1. File Menu

This menu performs basic file operations such as opening, closing, and saving files. For more information about file options, please see Chapter 8 - 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 8.1.1 - 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.

6.1.2. 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 6.6.1 - 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. EspressChart 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.

6.1.3. 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.

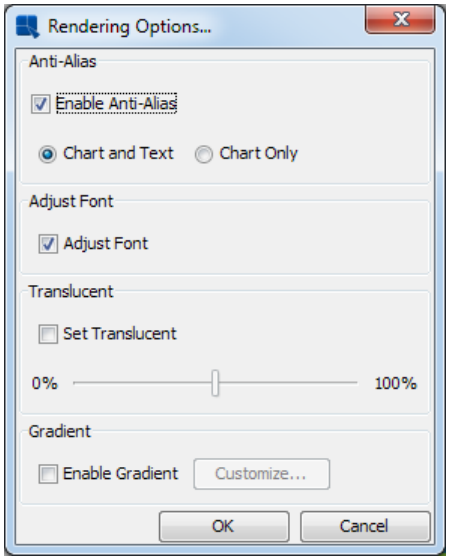
Histogram Options: This allows you to specify if you want to draw the chart as a histogram and display additional options to specify the frequency count.

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.



Rendering Options Dialog

Among the options available are:

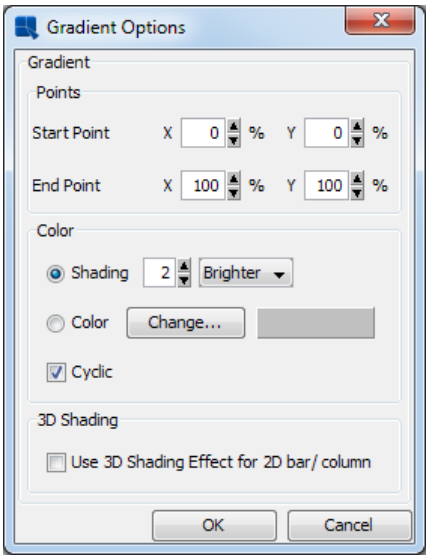
Enable Anti-Alias: This allows you to specify anti-aliasing for the chart. Anti-aliasing will smooth text and lines in the chart, creating a smoother appearance. This feature can be applied to either the entire file or just to the chart, which will leave the text unaffected.

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 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 8.2.1 - PDF Font Mapping

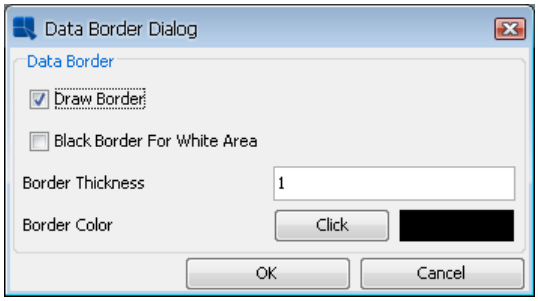
Chart Options: This brings up some specific options for the chart type that you are using. Options vary depending on chart type.

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.

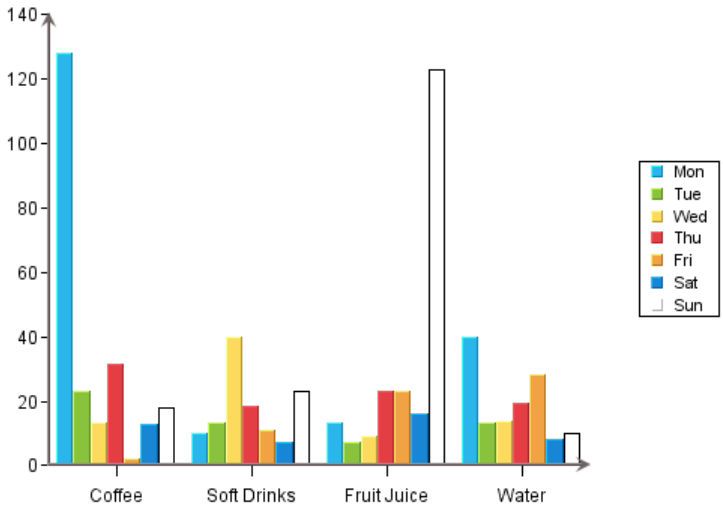
Data Border: Add a border to data elements (columns, bars, etc).



Data Border Dialog

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 2.6 - 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.

6.1.4. 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.

6.1.5. 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 Chapter 7 - Drill-Down

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.

6.1.6. 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 a database as the data source, this option allows you to modify the database connection that the chart uses. For more on this feature, please see Section 4.2.5 - Editing Database Connections.

Modify Query: If the chart uses a database as the data source for the chart, this option allows you to modify the query used to retrieve the chart data. For more on this feature, please see Section 4.2.4 - 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 2.6 - 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.

6.1.7. 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 EspressChart User's Guide.

6.1.8. 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.

6.2. 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:


Start a new chart


Open an existing chart

 Save the current chart

 Export the current chart

 Undo the last change

 Redo the last undone change

 Change data mapping for the current chart

 Modify data display properties

 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

 Refresh chart data

 Schedule periodic data refresh

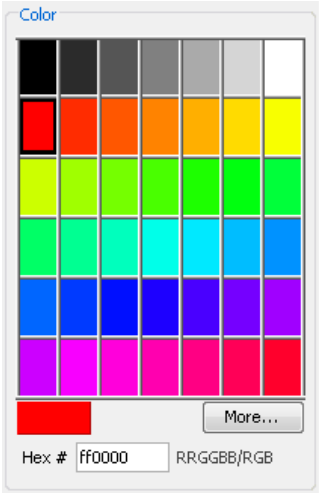
6.3. Color, Color set, Pattern, and Font Panels

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.

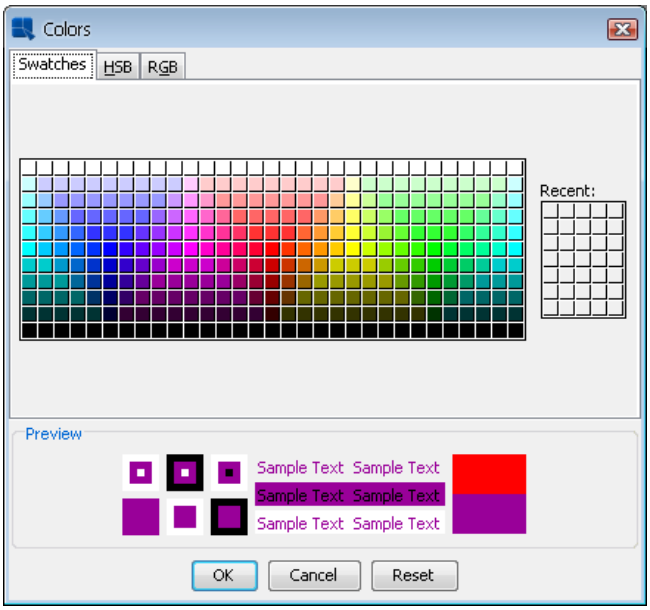
6.3.1. Color Panel

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.



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.

6.3.2. 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.

6.3.2.1. 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).

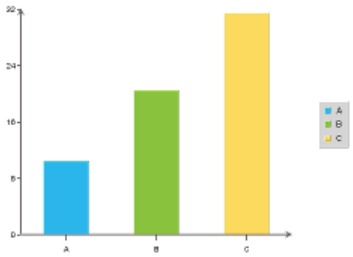


Image 1 - Example Chart

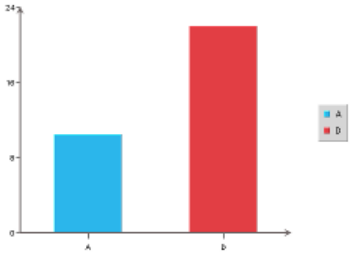


Image 2 - "Save Colors for Categories" Feature On

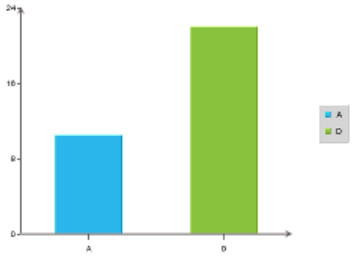


Image 3 - "Save Colors for Categories" Feature Off

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)

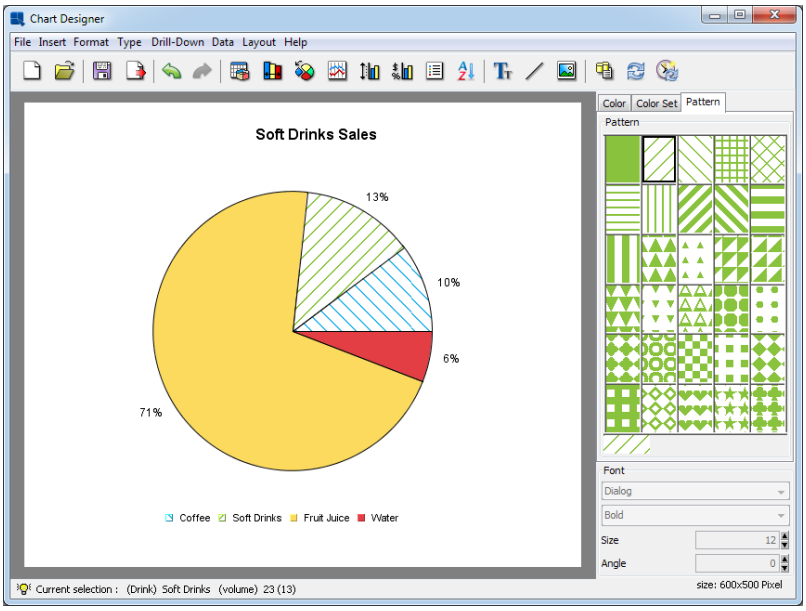
`category D` ... green color (category `D` has the second color from the color set, because the first is already assigned to the category `A`)

For this scenario, the color set will be selected under the *Color Set* tab because colors of data points correspond to the color set.

6.3.3. 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.

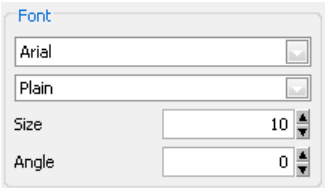
If a pattern has already been defined for a data point, the user can still change the color by selecting the color panel tab and picking a different color from the color palette. The color on the pattern will be changed to the new color immediately. The patterns shown on the pattern palette will change to the new color as well.



Pattern Example

6.3.4. 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.



Font Panel

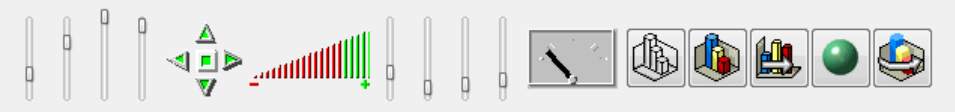
The first drop-down box allows you to select the font that you would like to use. The second drop-down box allows you to select the font style either plain, bold, italic, or bold and italic. The third box allows you to select the font size. The last box allows you to specify the angle of the text.

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.

EspressChart 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 21 or higher.

6.4. 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.

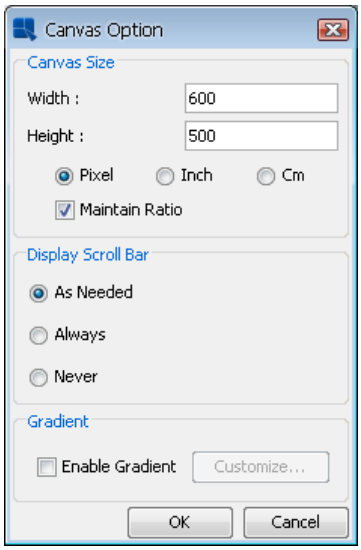
Animation On/Off: This toggle switch allows you to start/stop the animation of a three-dimensional chart. The speed of animation may be set using the Navigation Speed control. During animation, all panel controls except the animation speed control are disabled.

6.5. 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.

6.5.1. 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.



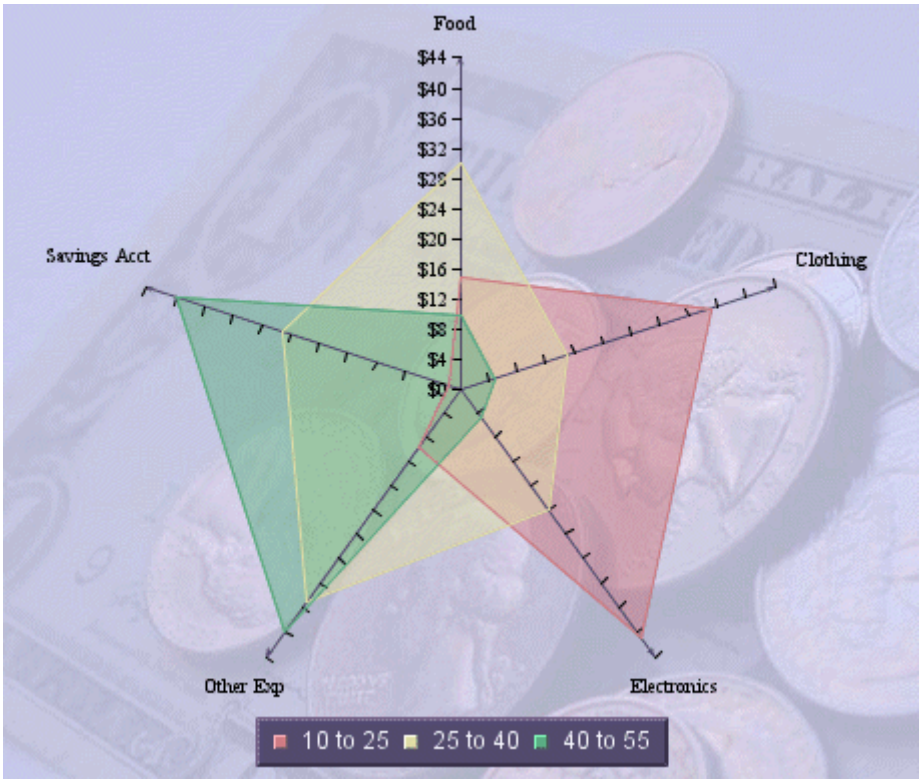
Canvas Formatting Dialog

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.

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 6.1.3 - Format Menu.

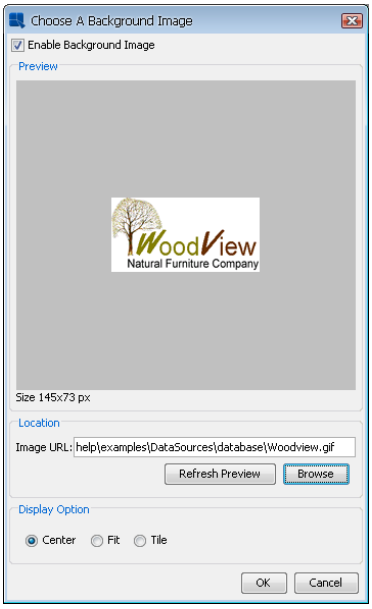
6.5.1.1. Background Images

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.



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.

6.5.2. 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 right click and drag within the plot area. This will cause the plot to enlarge or shrink. You can also size three-dimensional charts by using the zoom control in the navigation panel.

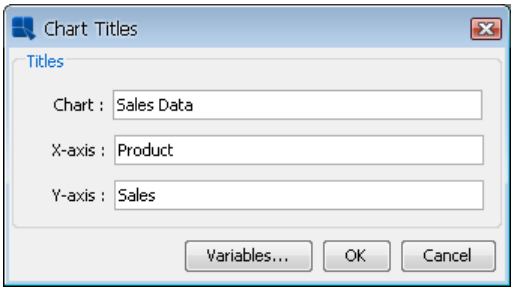
6.6. Adding Chart Elements

In addition to the default chart elements, EspressChart provides a number of additional elements that you can add to a chart.

6.6.1. 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.

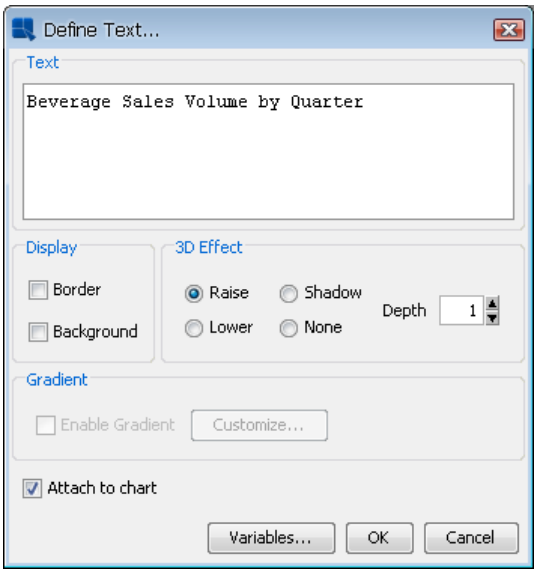


Add Titles Dialog

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.



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 6.1.3 - Format Menu.

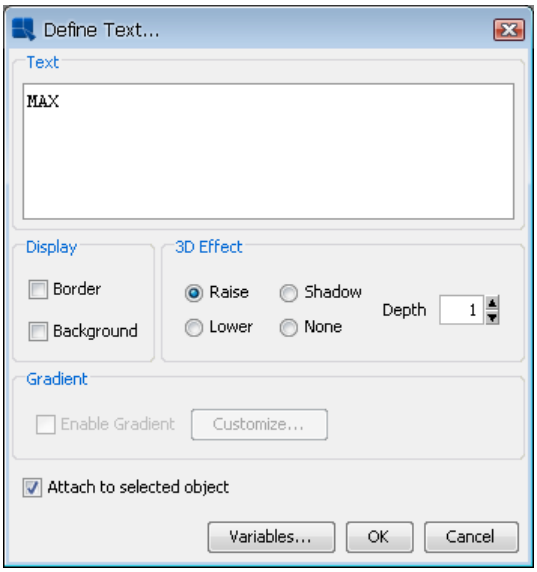
Once you have finished specifying the text, click *OK*. You will then be able to place the text in the chart. A small rectangle will follow your pointer around the chart canvas. Click the mouse where you would like to place the text.

Annotation Text:

EspressChart 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 6.6.2.3 - 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.

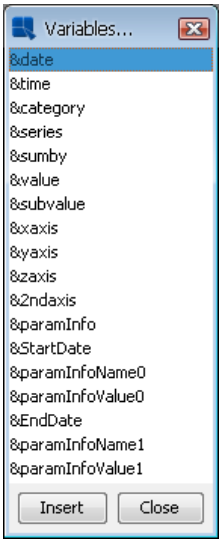


Add Text (Annotation) Dialog

6.6.1.1. TextVariables

EspressChart 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.



Variables Dialog

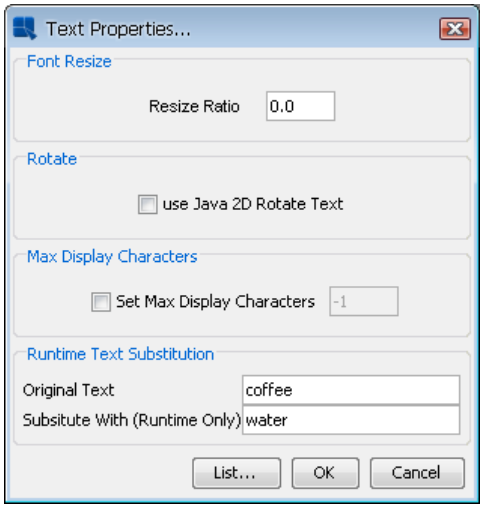
The following text variables are supported:

&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.

6.6.1.2. Text Replacement

EspressChart 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.



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”.

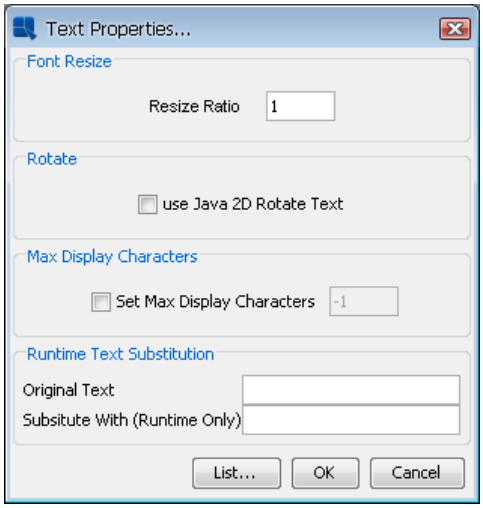
You can see a list of all the original text and the replacements by clicking on the *List* button. This will bring up a dialog listing all of the text replacement in the chart.

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.

6.6.1.3. Automatic Text Resizing

EspressChart 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.

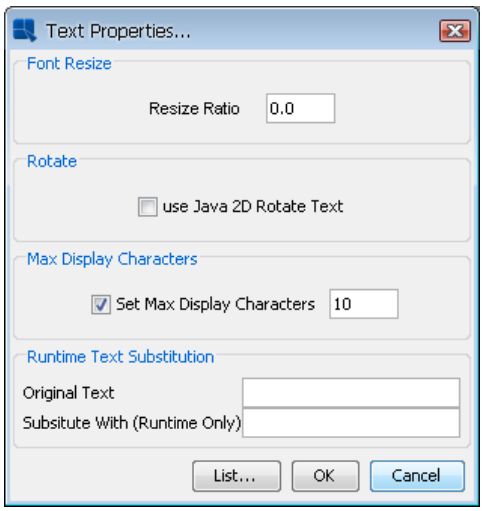


Text Properties Dialog

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.

6.6.1.4. 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, EspressChart 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.



Text Properties Dialog

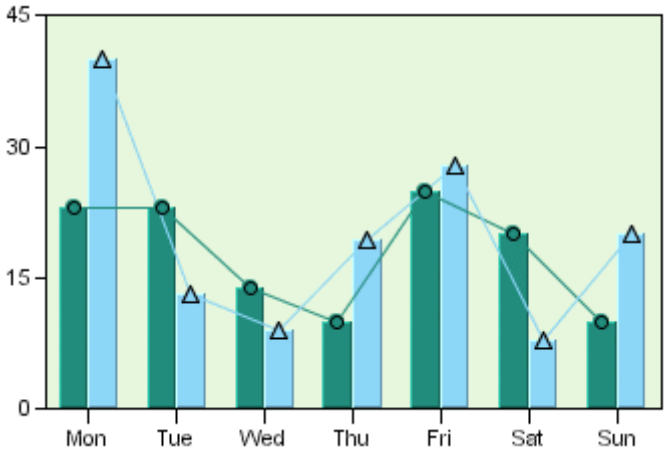
To enable text cropping, check the *Set Max Display Characters* option and specify the maximum character length in the dialog. Any text longer than the specified number of characters will be truncated.

6.6.2. Adding Lines

EspressChart allows you to add and format a number of different types of lines for charts.

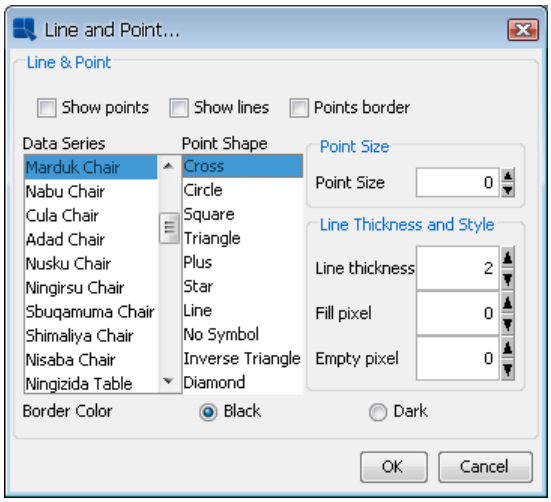
6.6.2.1. Line and Point Formatting

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 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.

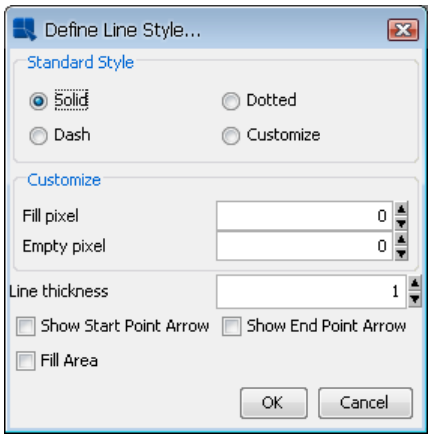
The last option allows you to change data point symbol border color to black or darker shade of symbol color.

6.6.2.2. 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.

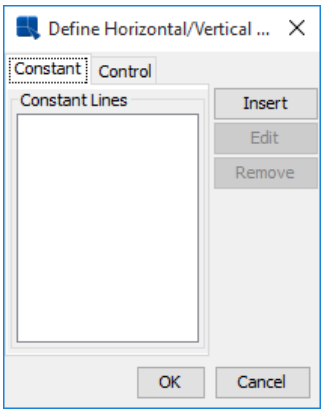


Line and Point Dialog for Floating Lines

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.

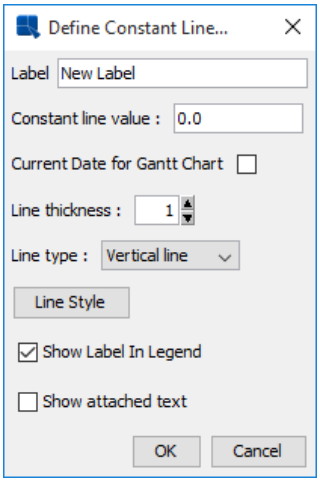
6.6.2.3. Fixed Horizontal/Vertical Lines

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.



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.

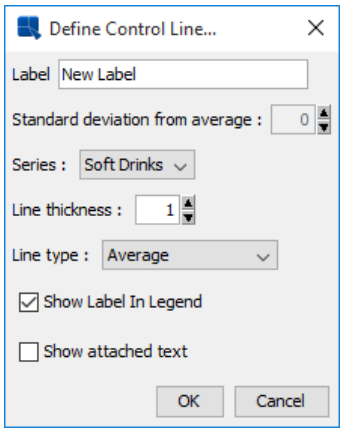
For Gantt charts, there is one extra option called *Current Date for Gantt Chart*. If you choose this option, the *Constant line value* field will deactivate and the current date will be used as the line value (the line position will be updated every time you run the chart).

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 6.6.1 - Adding Text.

To add a control line, click on the *Control Line* tab in the dialog.

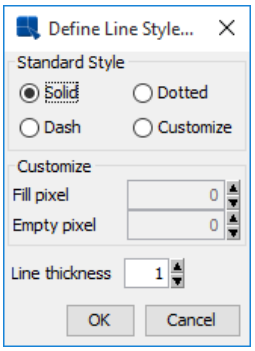


Control Line Dialog

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.



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.

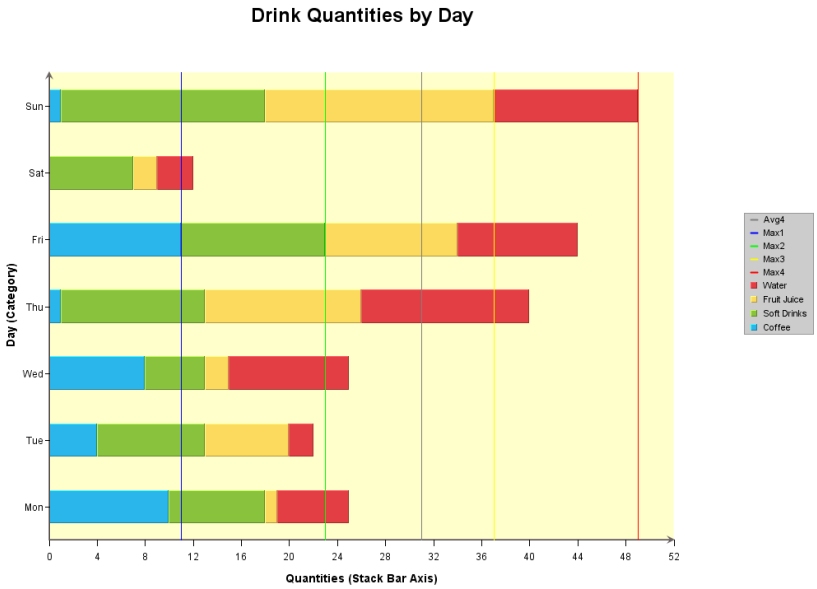
6.6.2.3.1. Multiple control lines for Stack Type Charts

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 10.6.2.5 - 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

6.6.2.4. Trend Lines

A powerful feature of EspressChart 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.

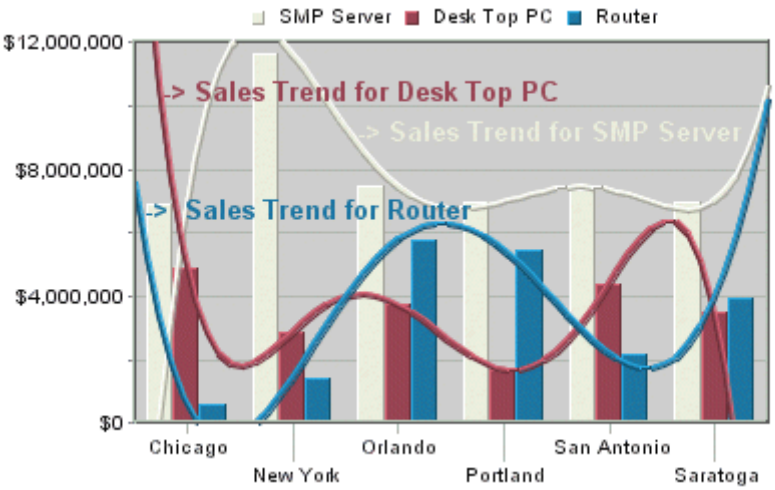
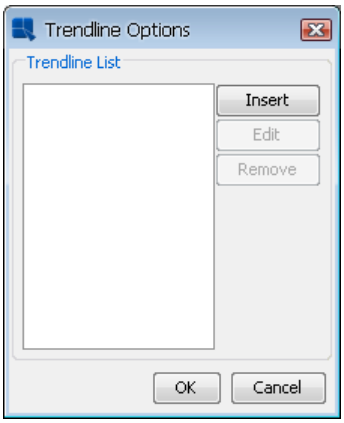


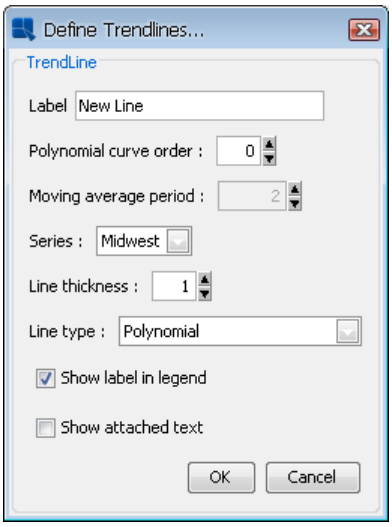
Chart with Trend Lines

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

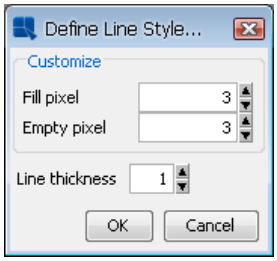
Clicking on *Insert* or clicking on *Edit* when an existing trend line is selected, respectively, will bring up a dialog allowing you to configure the selected trend line.



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.



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.

6.6.2.4.1. 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.

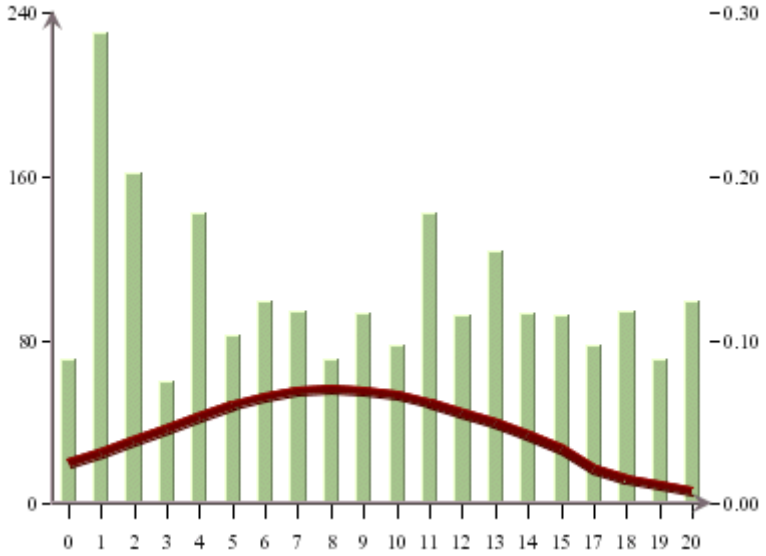
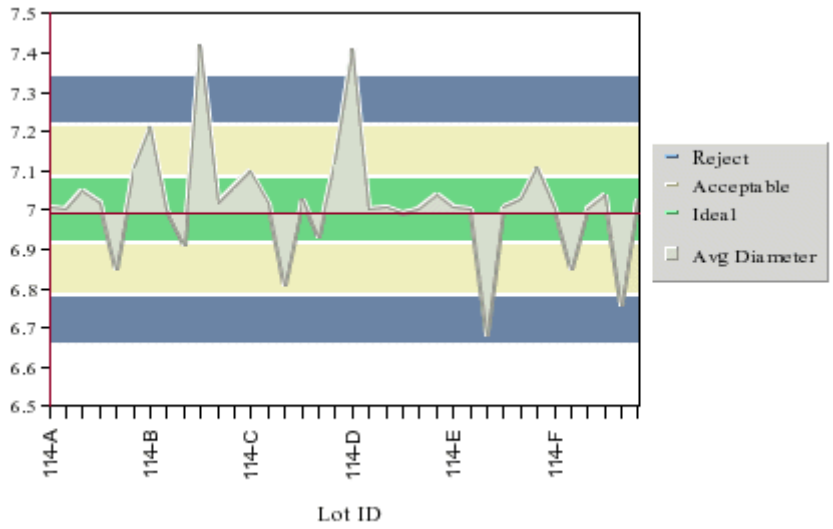


Chart with Normal Distribution Curve

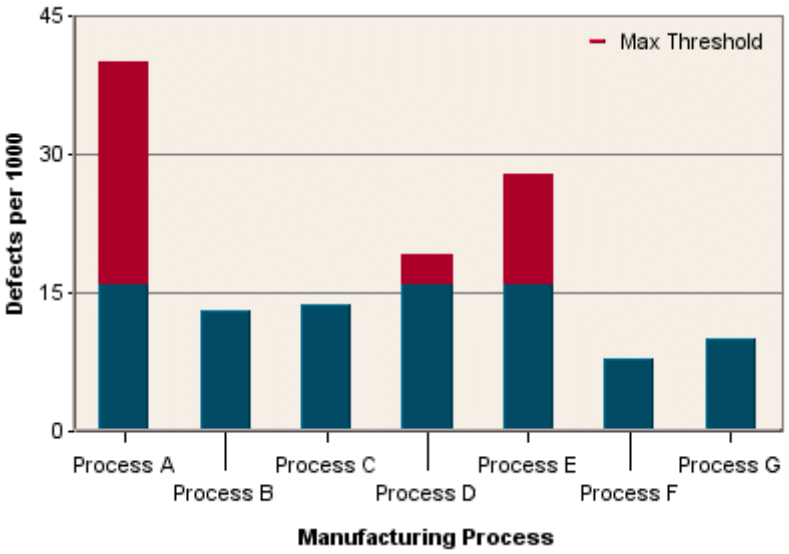
Since the scale for the curve is usually different than the scale for the value axis, the curve is shown on a secondary axis. You can modify the scale by changing the scale for the secondary axis.

6.6.3. 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.

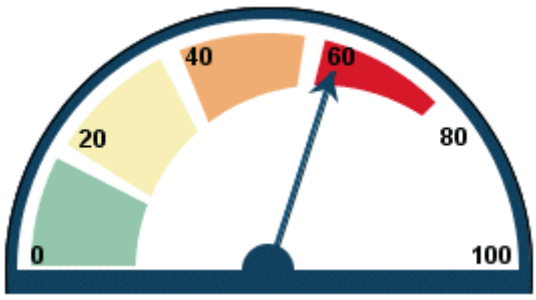


Two-Dimensional Area Chart with Control Areas



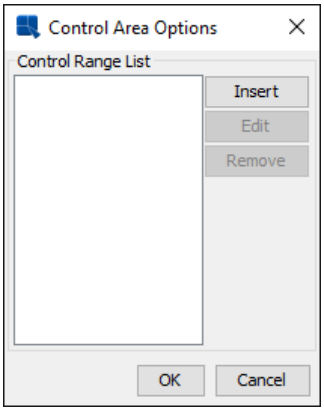
Column Chart with Control Area Drawn for Data Points

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.



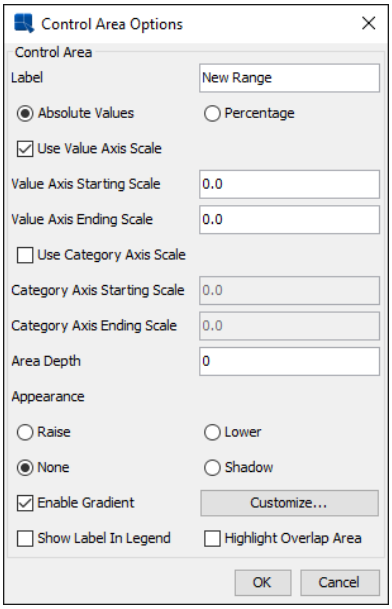
Dial Chart with Control Areas

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.



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.

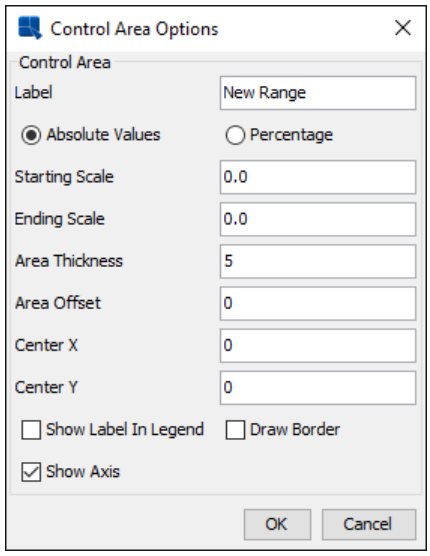
Appearance: This option allows you to specify a 3D or shadow effect for the control area. If the area depth is specified as zero, the appearance will not take effect.

Enable Gradient: Enable color gradient for the control. Gradient settings are described in Section 6.1.3 - 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.



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.

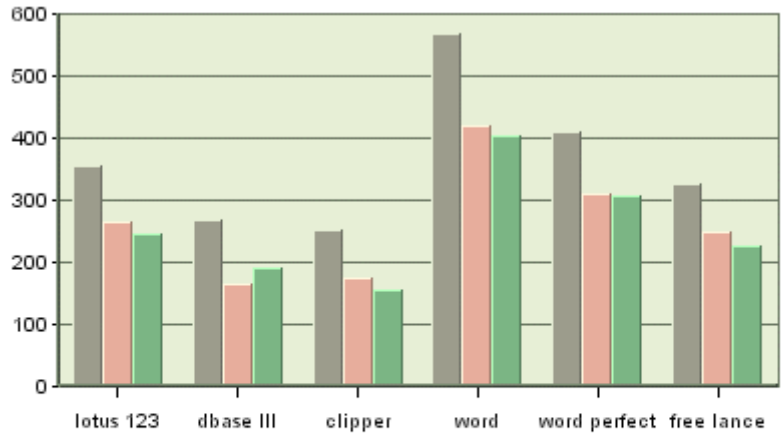
Draw Border: This option allows you to draw a border around the control area.

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.

6.6.4. Adding Tables

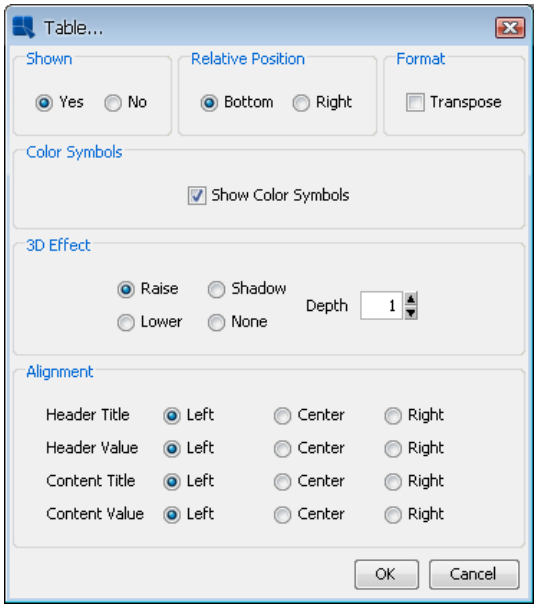
In addition to displaying charts, you can also display a table showing the data points displayed in the chart. The table can be placed below or to the right of the chart plot.



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.



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

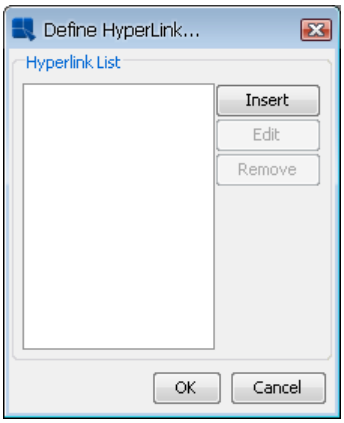
The *Alignment* options allows you to specify horizontal alignment of the text in the table cells, either left, center, or right. The alignment can be set for row headers, column headers, and inner table cells.

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.

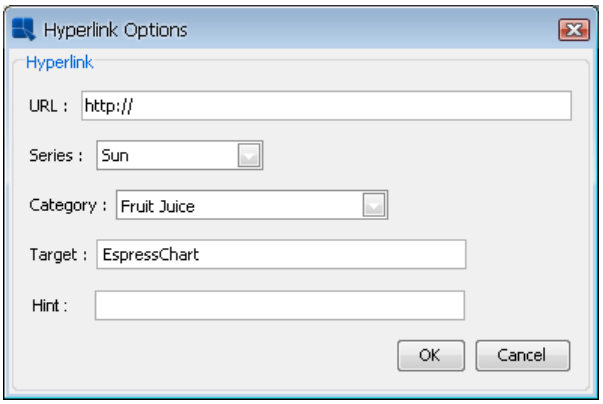
6.6.5. Adding Hyperlinks

EspressChart 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.



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.

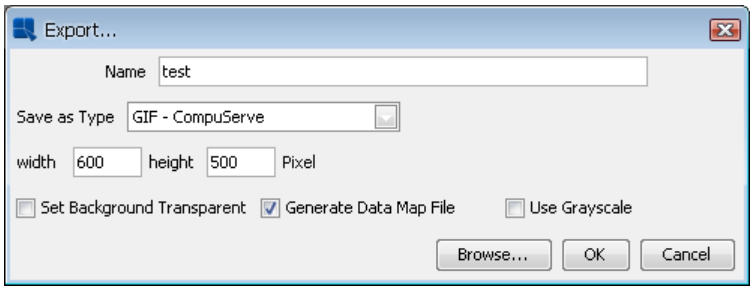
The *Hint* field allows you to enter text that will pop-up when you move your mouse cursor over a data point. If you want to create pop-up labels without hyperlinks, you can leave the URL field blank and only specify the hint.

6.6.5.1. 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.



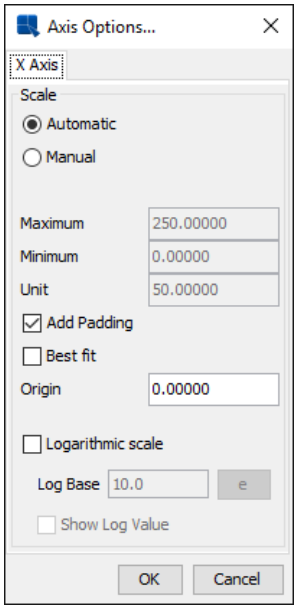
Export Dialog

6.7. Formatting Chart Axes

EspressChart 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.

6.7.1. 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.



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.

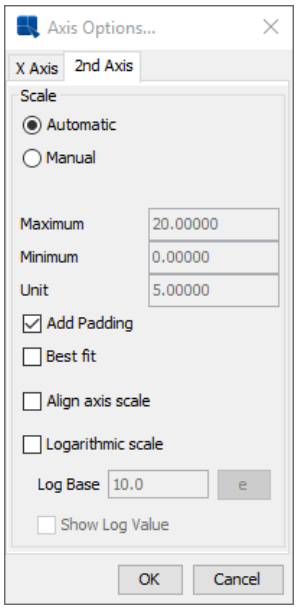
Best fit: This will automatically place the origin of the chart based on the minimum and maximum values of the data.

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.

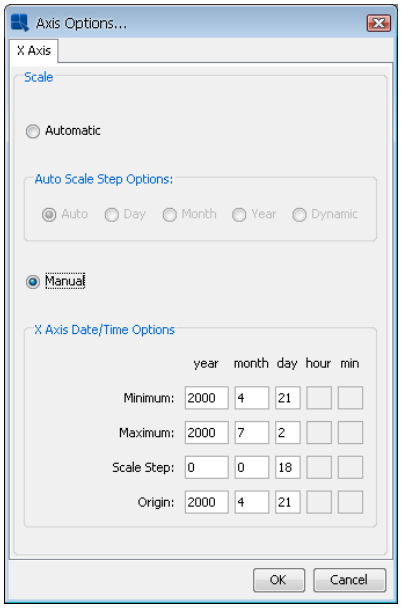
- **Log Base:** This allows you to specify the base for the log value.
- **Show Log Value:** This specifies whether to show log values in the axis labels or not.

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.



Axis Scale Dialog for Secondary Axis

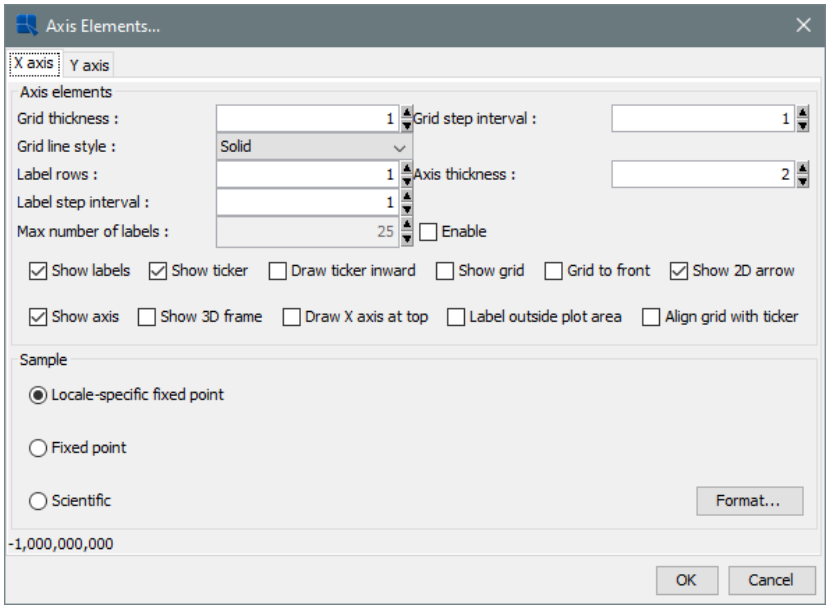
The axis scale dialog is different for a Gantt chart. You can still select the *Automatic* option to have the scale configured automatically or the *Manual* option to set the scale manually. The *Automatic* axis scale has a few Auto Scale Step Options: Auto, Day, Month, Year, and Dynamic. The *auto* 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).



Axis Scale Dialog for Gantt Chart

6.7.2. 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 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.

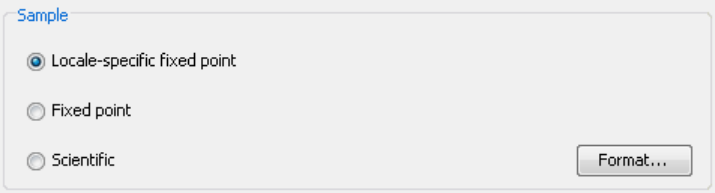
Grid line style:	This option allows you to select the grid line style (solid, dotted, dash).
Label rows:	This option allows labels to be displayed in alternating rows. This can prevent overlapping. This option is only available for X-axis.
Axis thickness:	This option allows you to set the thickness of 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.

6.7.2.1. Axis Label Formatting

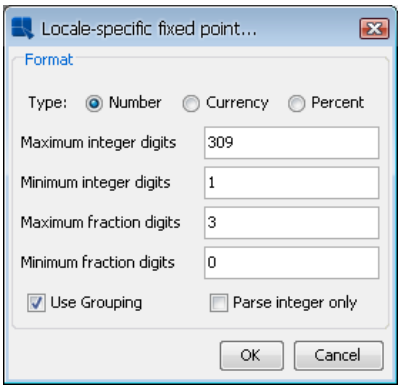
The axis elements dialog also allows you to format the appearance of the axis labels, depending on what type of data is plotted on the axis. The *Format Options* portion of the dialog contains the label formatting dialog.

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.



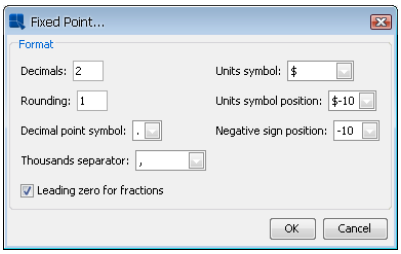
Numeric Data Format Options

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.



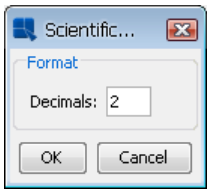
Locale-Specific Formatting Options

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



Fixed Point Formatting Options

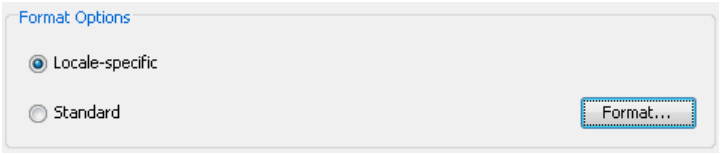
Scientific: This will display the data in scientific notation. Additional formatting for this option allows you to set the number of decimals.



Scientific Formatting Options

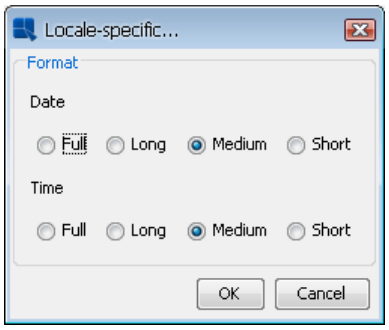
Formatting Date/Time Data

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.



Date/Time Data Format Options

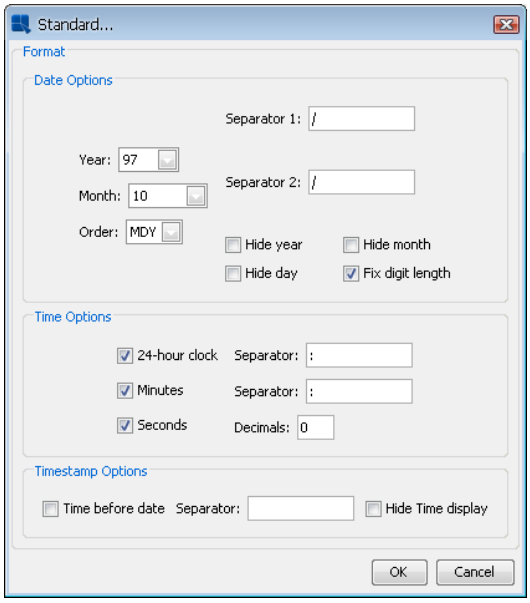
Locale-Specific: This will change the format of the data depending on the locale in which it is being viewed. Additional formatting for this option allows you to select full, long, medium, or short notations for date and time information. Other display attributes will vary depending on locale.



Locale-Specific Formatting Options

Standard:

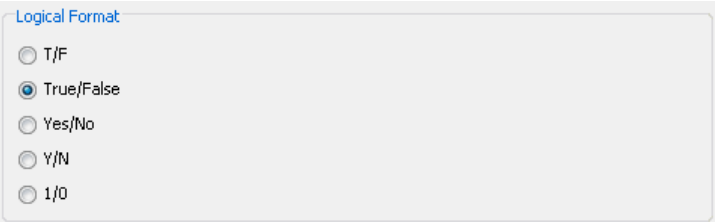
This will keep the data format consistent, regardless of locale. Additional formatting for this option allows you to select year and month displays, as well as the order in which month, day, and year information is presented. You can also select the characters that you want to be used as separators. 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.



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 Data Formatting Options

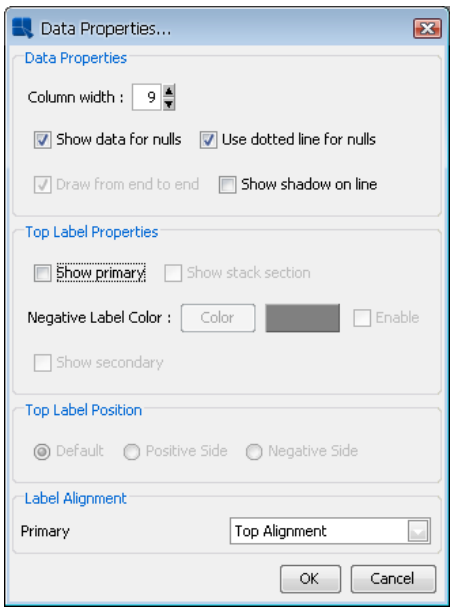
Any changes you make to the data formatting will take effect after you click on the *OK* button in the axis elements dialog. Note that there are no additional formatting options for string data.

6.8. Formatting Plot/Data Elements

EspressChart 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.

6.8.1. 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 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.

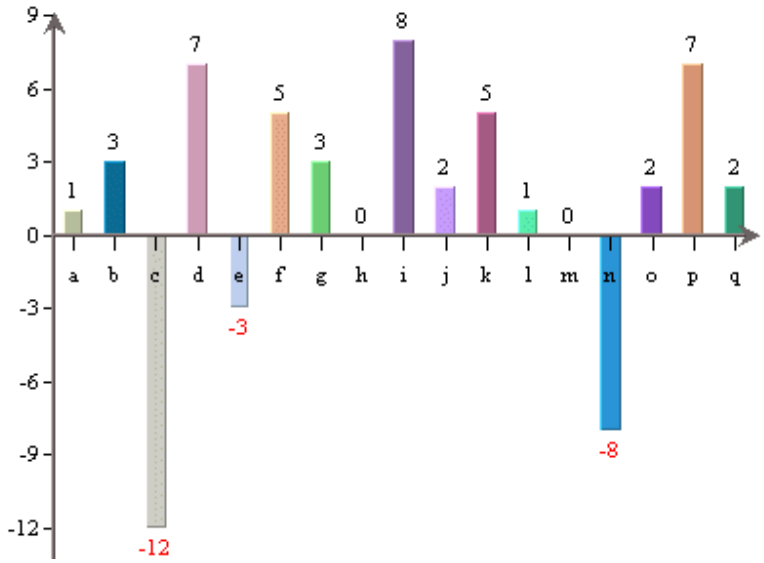


Chart with Colored Negative Top Labels

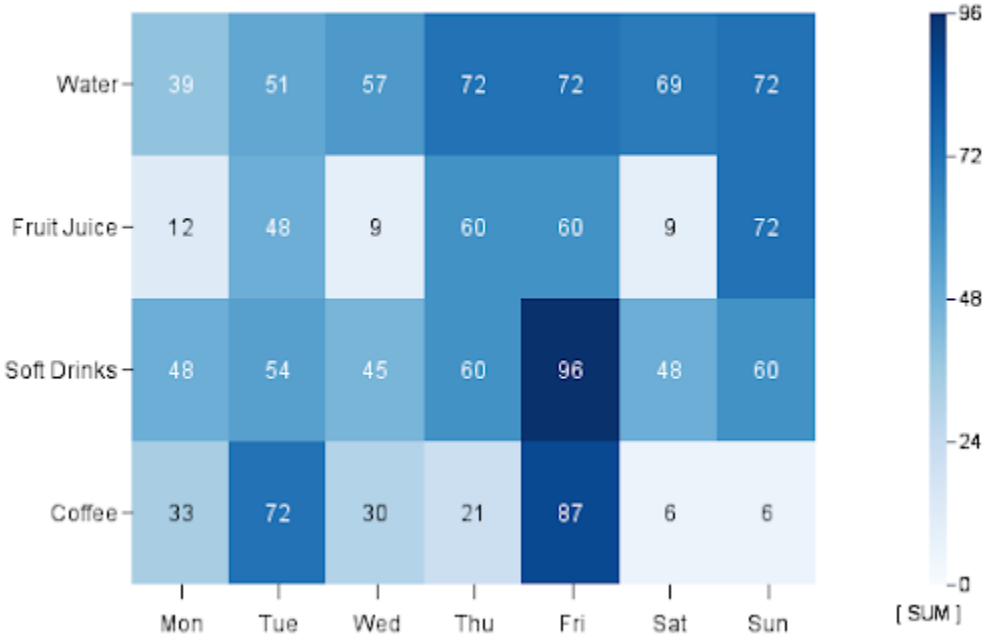
Show secondary: This option will display data top labels for the secondary values in the chart.

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.

6.8.1.1. 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

6.8.2. Date/Time Based Zooming

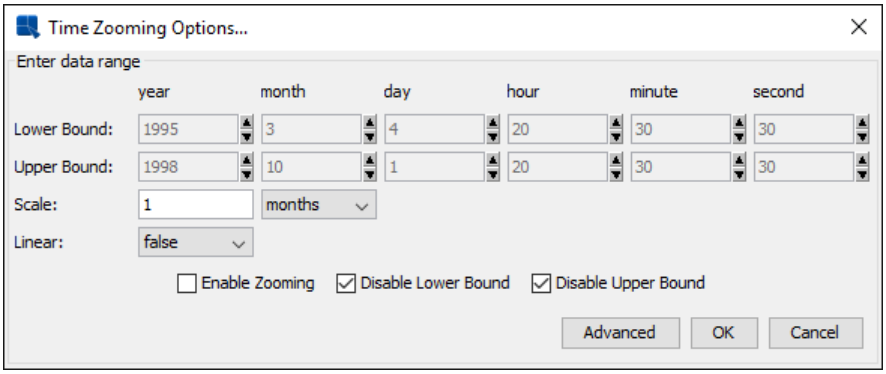
For charts displaying date or time data on the category axis, EspressChart provides a unique feature allowing users to perform date/time based zooming. Using this feature, you can group the category elements into user-defined intervals and aggregate the points in each group. You can also filter the data 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.

6.8.2.1. 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.



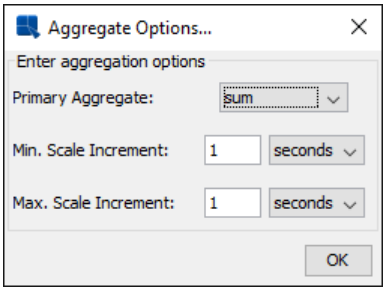
Zoom Options Dialog

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.



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.

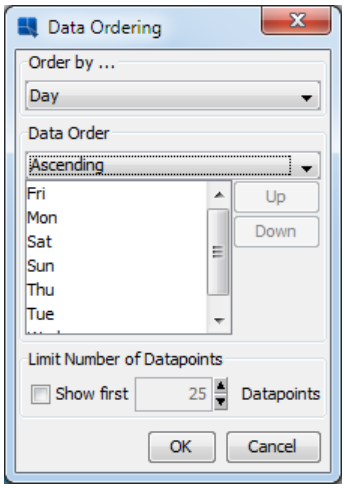
6.8.2.2. Zooming In Chart Viewer

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.

6.8.3. Data Ordering

EspressChart 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:



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:

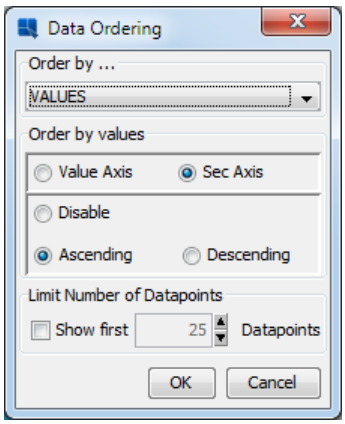
DataSource Order: This option turns the ordering off. The categories/series order will depend on the data source only and will not be altered by the EspressChart at all.

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.

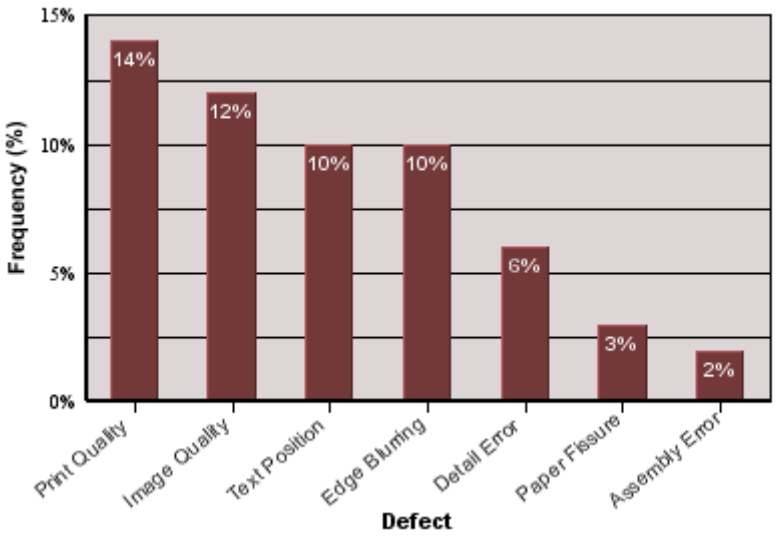
Customize: This option allows you to customize the categories/series order. To customize the order, select an item from the list of Categories/Series items and then move it upwards or downwards in the list by clicking on the *Up* or *Down* button (the buttons are inactive until you select the *Customize* 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.



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

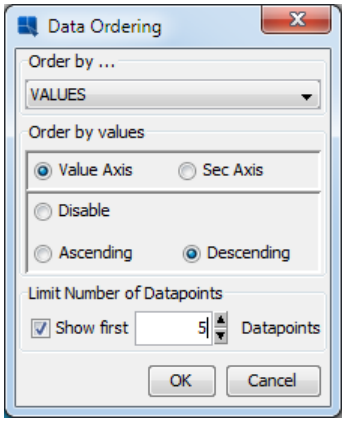
Please note that any sorting set will be re-applied if the chart is refreshed and/or if the data changes.

6.8.3.1. 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.



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.

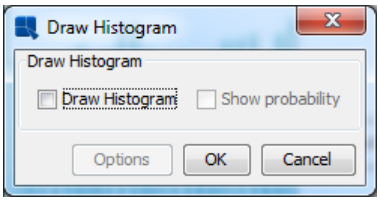
6.8.4. Histograms

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. EspressChart 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.

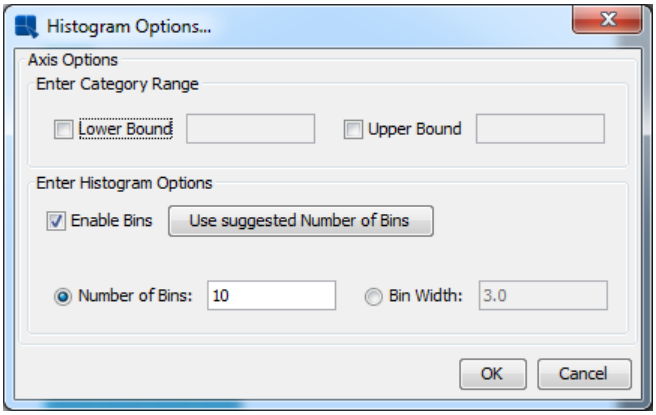
To create a histogram, you must start with a 2D column chart, bar chart, line chart or area chart. In the Data Mapping dialog, make sure DataSeries is set to *None* and the field you want to plot in the histogram is set in the Category axis.

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.



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.



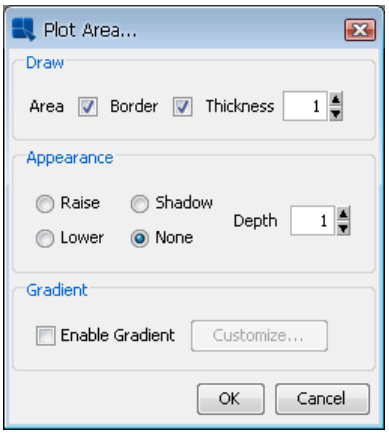
Histogram Options Dialog

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.

6.8.5. 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.



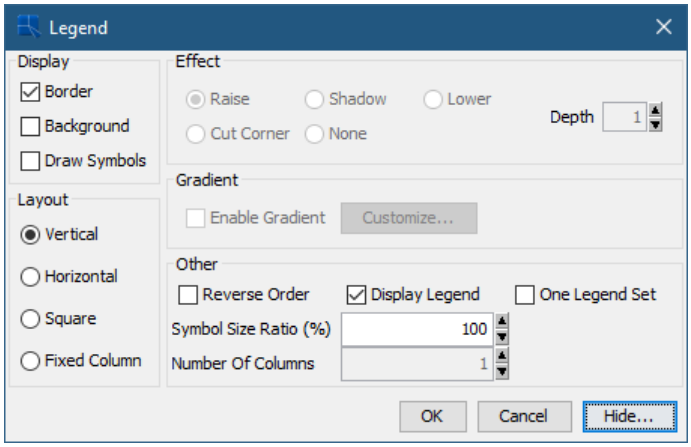
Plot Area Dialog

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 6.1.3 - Format Menu.

6.8.6. 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.



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.

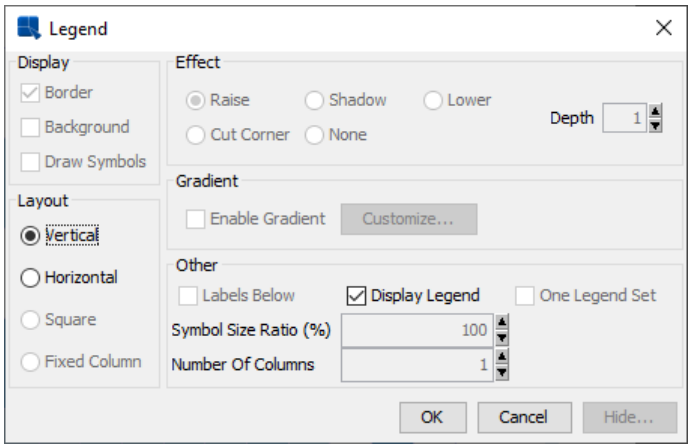
Layout: This allows you to change the legend from vertical, horizontal, square, or fixed column layout.

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 6.1.3 - 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.

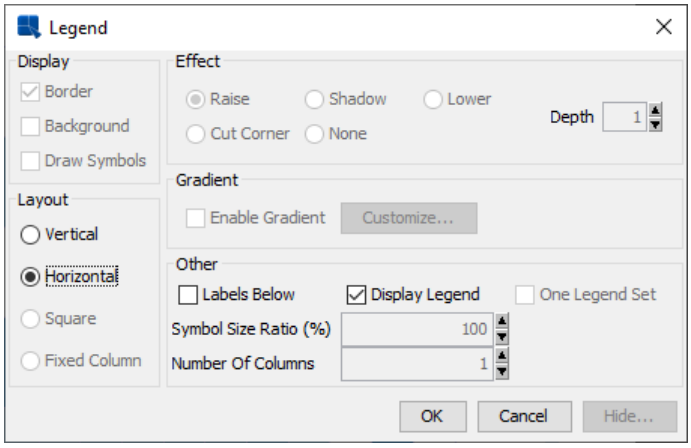
Additionally, you can remove specific category/series elements from the legend by clicking on the *Hide* button. This will bring up a list of the legend items, where you can select which elements you would like to hide.

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.



Heatmap Vertical Legend

In the horizontal layout, you can choose to display tickers and labels below the legend by selecting the *Labels Below* option.

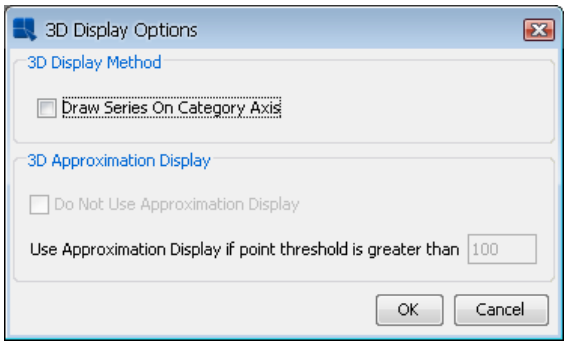


Heatmap Horizontal Legend

6.8.7. 3D Display Options

EspressChart 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.

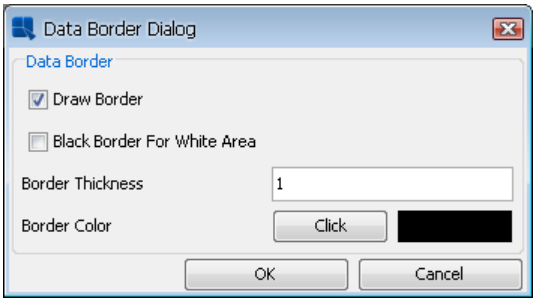


3D Display Options Dialog

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).

6.8.8. Data Border

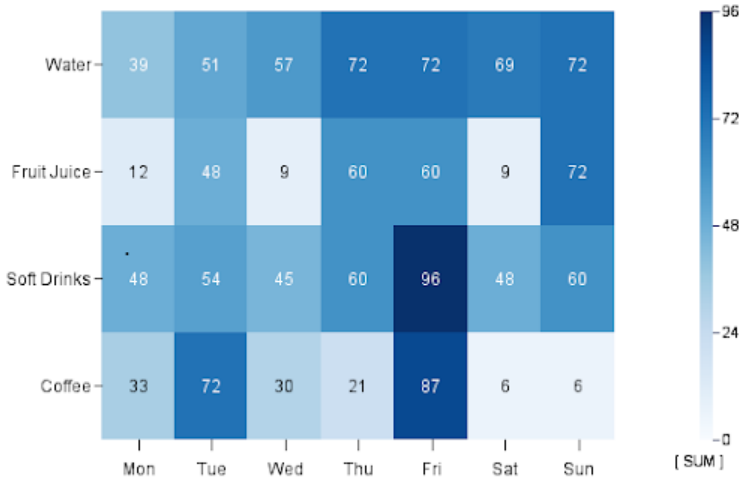
For column, bar, stack column, stack bar and HLCO charts, EspressChart allows you to configure a border around the columns. To set the border option, select Format → Data Border. This will bring up a dialog allowing you to set border options.



IData Border Dialog

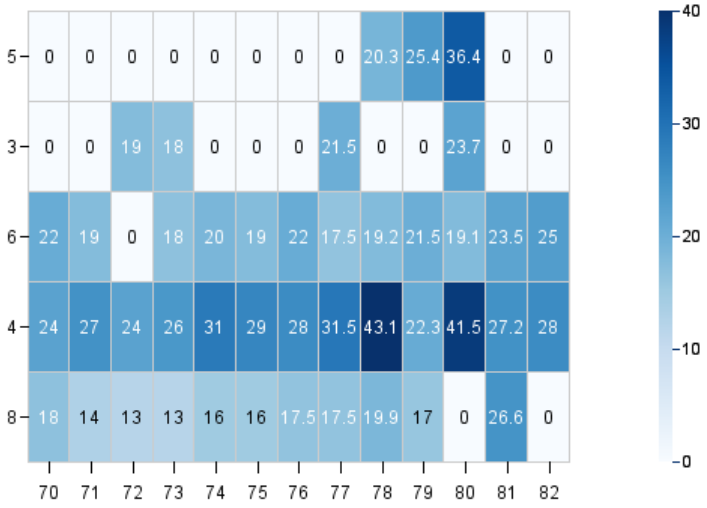
The first option allows you to turn on/off the data border. The second option allows you to set a black border for any white areas in the chart. Please note that the border is black only if the first option is unchecked and will only appear around white areas in the chart. The third and fourth options 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.

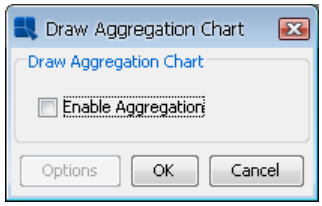


Heatmap chart with light gray colored border

6.8.9. Aggregation

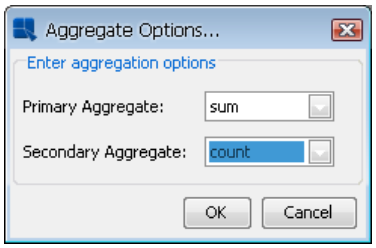
EspressChart allows you to aggregate data if there is more than one data point associated to a given category (and its series and/or stack, if a series and/or stack is present). This allows for a broader look at the data rather than just a single data point (out of many).

To aggregate the data, select Format → Aggregation Options. A dialog will appear allowing you to enable aggregation.



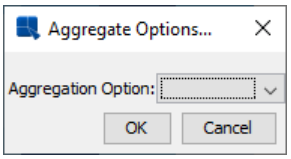
Select Aggregation Dialog

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 Dialog

For heatmap charts, the aggregation option can be used when there are multiple data values at a single (x, y) point.



Aggregate Options

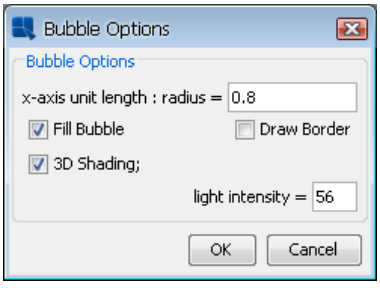
By default, no aggregation is applied to heatmap charts, and the first data value is always selected if there are multiple values at the same (x, y) point.

6.9. 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.

6.9.1. Bubble Charts

For bubble charts, the following dialog is displayed:



Bubble Options Dialog

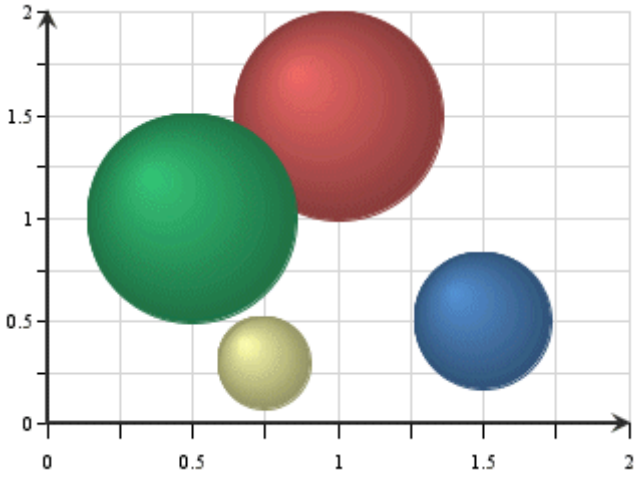
The following options are available for bubble charts:

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.

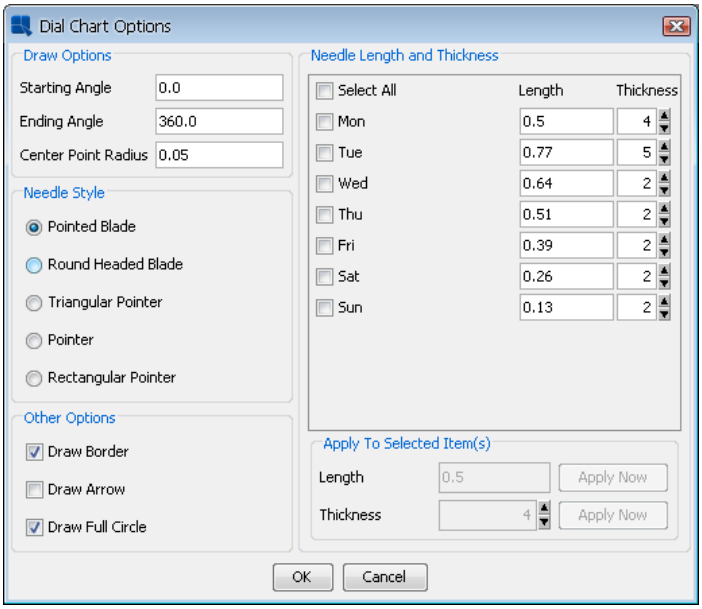
3D Shading: This option allows you to add 3D shading to the bubbles. You can also specify the light intensity for the shading.



Bubble Chart with 3D Shading

6.9.2. Dial Charts

For dial charts, the following dialog is displayed.



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.

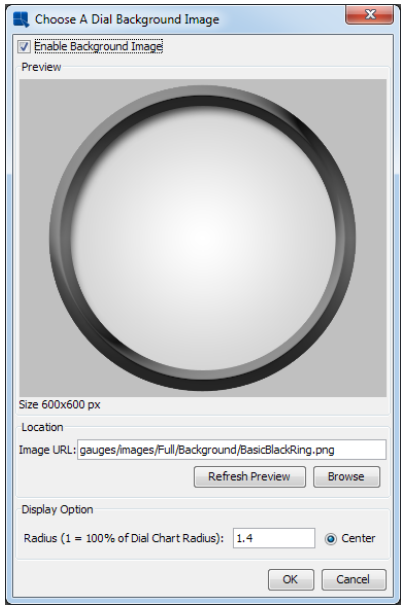
Draw Full Circle: This option specifies whether to draw the dial as a complete circle (360 degrees) or only draw the portion of the circle determined by the starting and ending angles.

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.

6.9.2.1. 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.

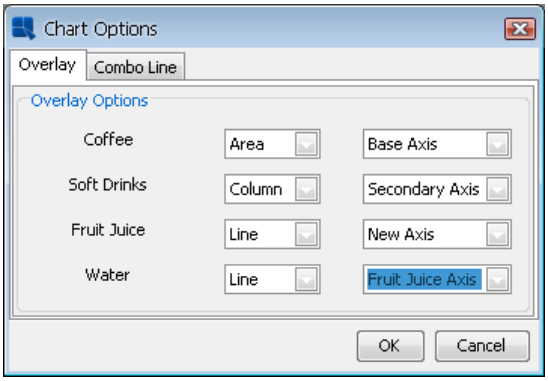


Dial Chart Background Image Dialog

Selecting an image works in the same way as the background image dialog, see Section 6.5.1.1 - 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.

6.9.3. Overlay Charts

For overlay charts, the following dialog is displayed:

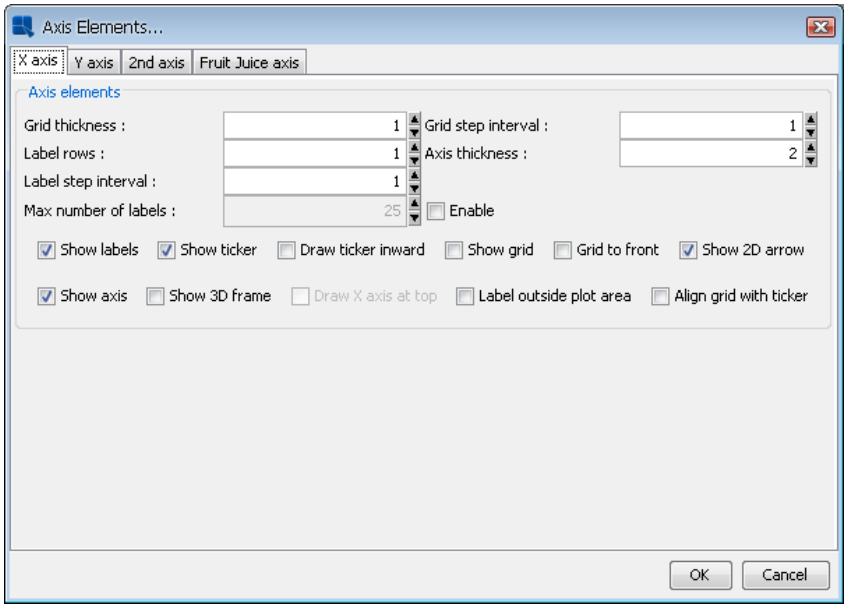


Overlay Options Dialog

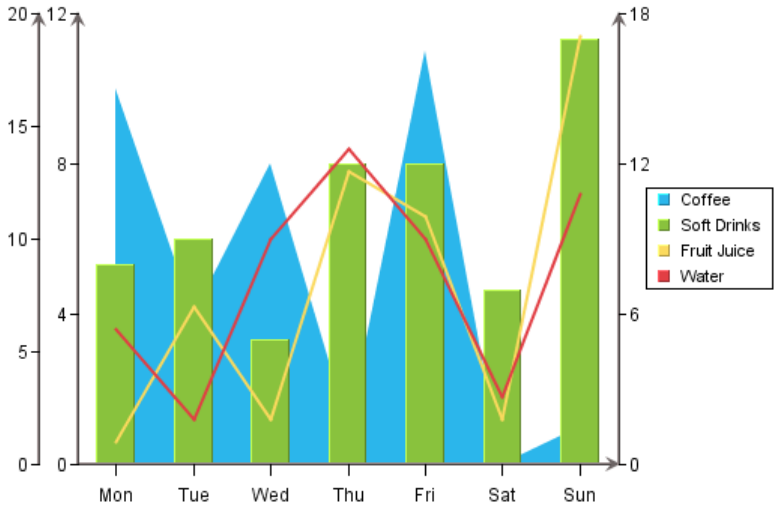
From this dialog, you can specify which chart type you would like to use for each element of the data series. Available chart types for the series elements in an overlay chart are column, area, and line. You can also choose not to display certain series elements.

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.

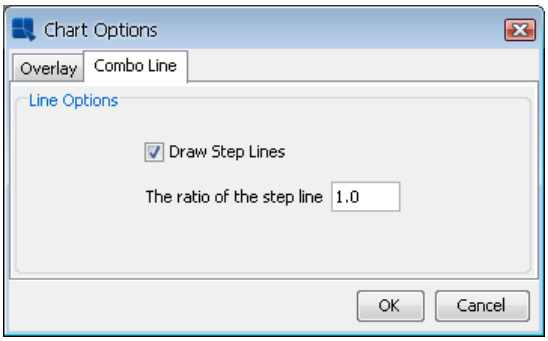


Axis Options Dialog for Multiple Value Axes



Overlay Chart with Multiple Value Axes

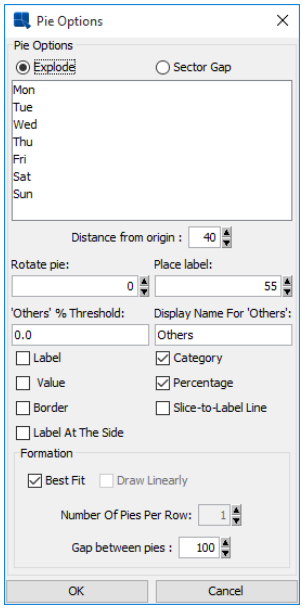
If one of the chart types used by the overlay chart is a line, you can specify whether or not to draw the line as a step line using the *Combo Line* tab. For more about step lines see Section 6.9.5 - Line Charts:



Combo Line Options for Overlay Charts

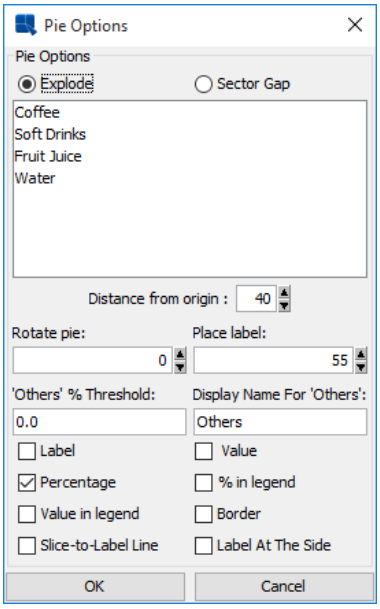
6.9.4. Pie Charts

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 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 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.

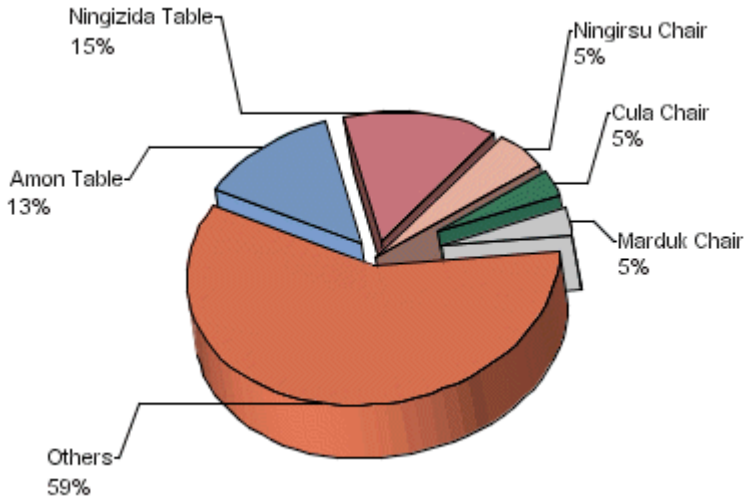
% in legend: This option allows you to display the percentage represented by each slice in the pie in the legend. This can be a preferable presentation if the pie slices become too thin. This option is only available if the pie chart does not have a data series.

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.

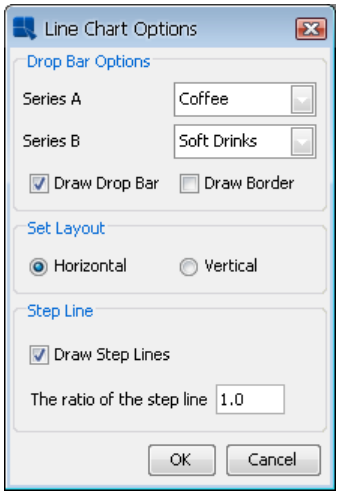
Draw Linearly: This option will arrange multiple pies in a straight horizontal line. It's only available for pies with data series.

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.

6.9.5. 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 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.

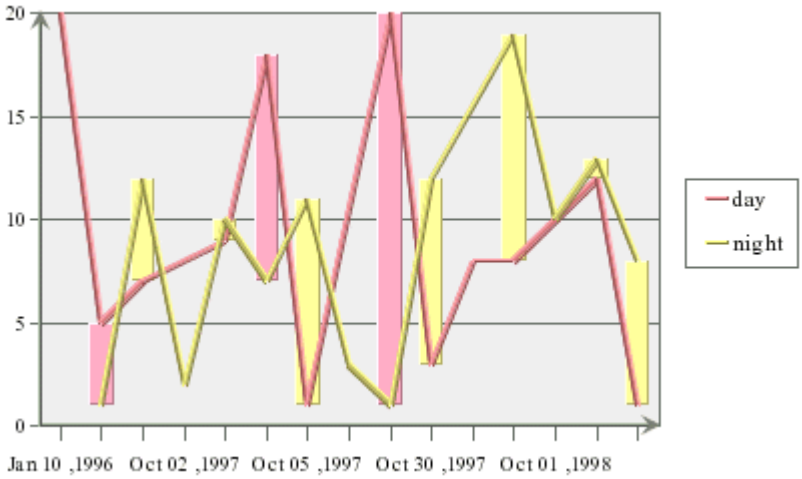
Series B: This option specifies the second series element for the drop bar.

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.

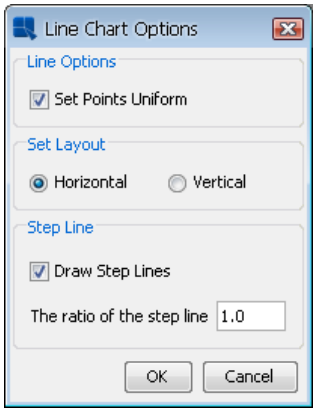
Set Layout: This option specifies whether to draw a line chart in vertical or horizontal orientation.

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.



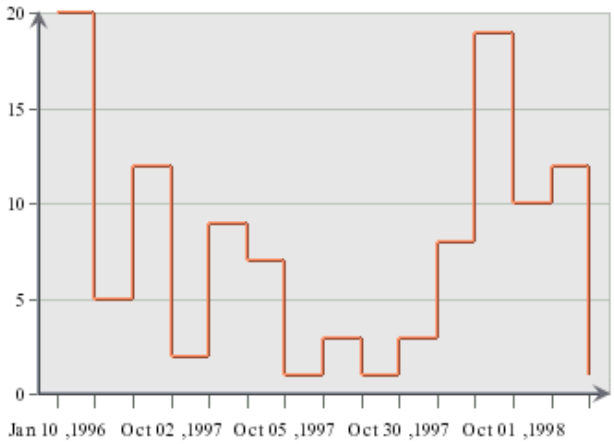
Line Options Dialog (without series)

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.



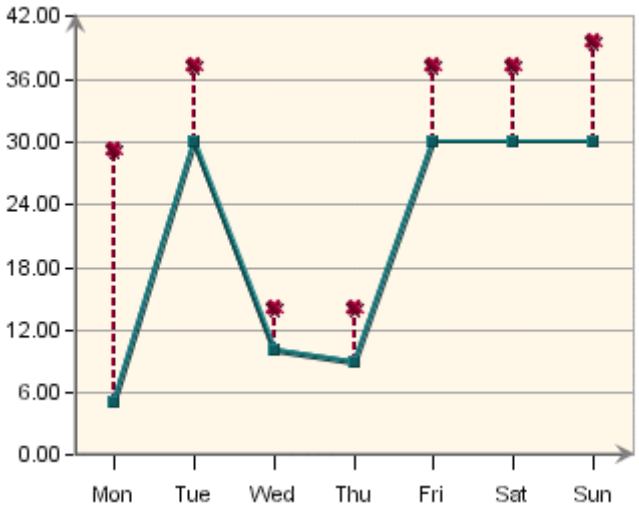
Line Chart with Step Lines

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.

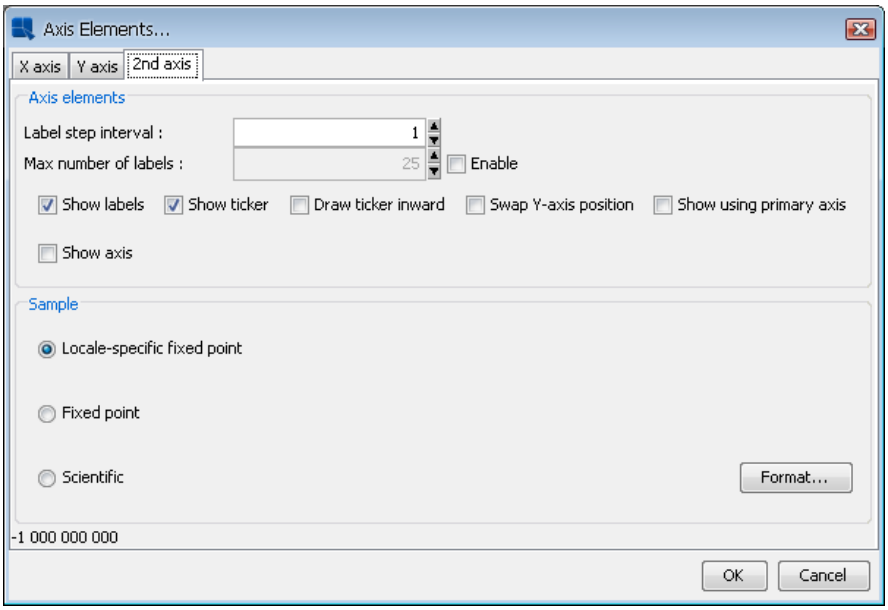
6.9.5.1. Double Value Line Charts

EspressChart 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.

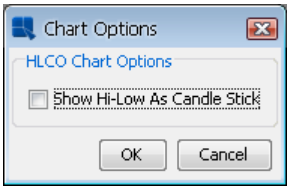


Axis Elements Dialog for Line-Line Combination Charts

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.

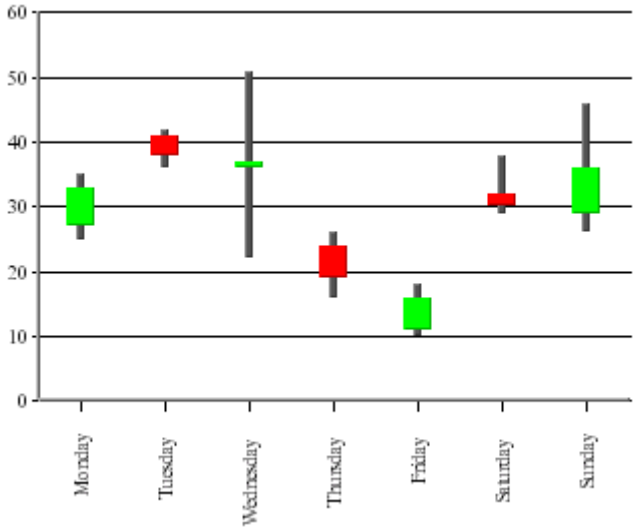
6.9.6. HLCO Charts

For HLCO charts, the following dialog is displayed:



HLCO Options Dialog

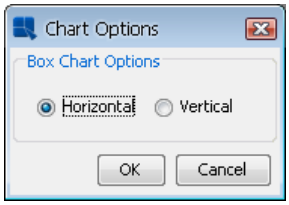
The *Show Hi-Low As Candle Stick* option will turn the HLCO chart into a candle representation. A candle HLCO chart blends high, low, close, and open data into a single object that resembles a candlestick.



HLCO Candlestick Chart

6.9.7. Box Charts

For box charts, the following dialog is displayed:

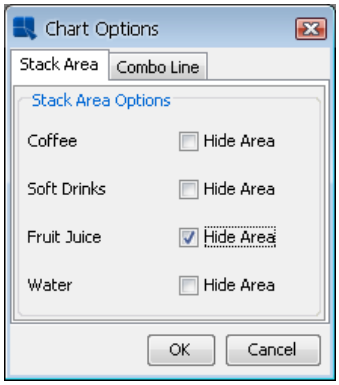


Box Options Dialog

This dialog allows you to specify whether to display the box chart in a horizontal or vertical orientation.

6.9.8. Stack Area Charts

For stack area charts, the following dialog is displayed:

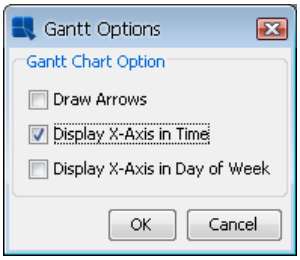


Stack Area Options Dialog

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.

6.9.9. Gantt Charts

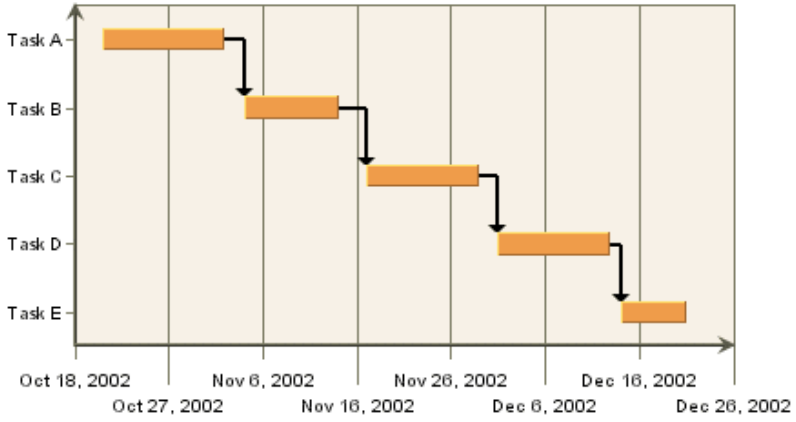
For Gantt charts, the following dialog is displayed:



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.



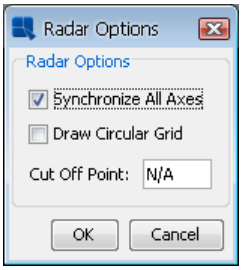
Gantt Chart with Arrows

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.

6.9.10. Radar Charts

For radar charts, the following dialog is displayed:



Radar Options Dialog

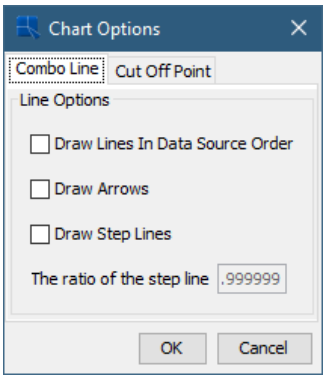
By default, the scale is same for all axes in the radar chart. Unchecking the *Synchronize All Axes* option will allow each axis in the radar chart to be scaled independently. You can select to use auto-scaling for each axis, or you can set the scales manually by invoking the axis scale dialog.

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.

6.9.11. Scatter Charts

For scatter charts, the following dialog is displayed:



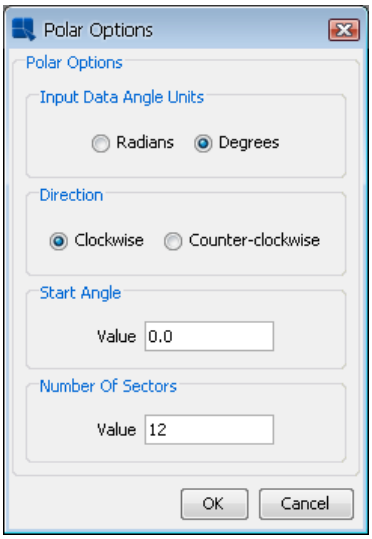
Scatter Options Dialog

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.

6.9.12. Polar Charts

For polar charts, the following dialog is displayed:



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.

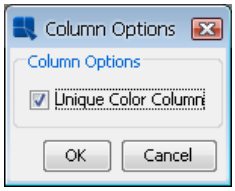
Direction This option allows you to specify whether the circular plot should be drawn clockwise or counter-clockwise.

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.

6.9.13. Column Charts with Series

For column charts with data series, the following dialog is displayed:

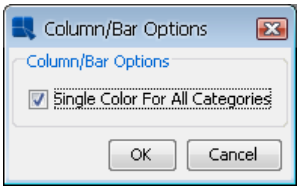


Column Options Dialog

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.

6.9.14. Column/Bar Charts without Series

For column/bar charts without data series, the following dialog is displayed:

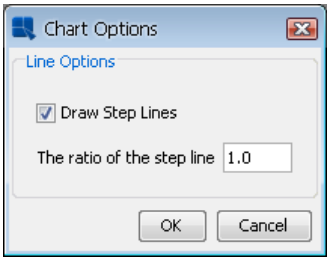


Column/Bar Options Dialog

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.

6.9.15. Two-Dimensional Line Combination Charts

For any other two-dimensional line combination chart, the following dialog is displayed:

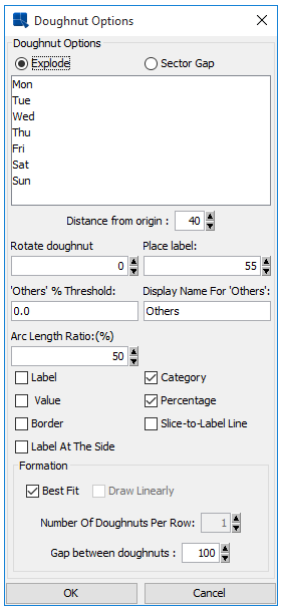


Line Combination Chart Options

This dialog allows you to specify whether to draw the combo line as a step line, as well as specify the step line ratio.

6.9.16. 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 6.9.4 - Pie Charts for more details.



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.

6.10. Chart Designer in Mac OS X

When running Chart Designer in OS X most of the controls are the same as for Windows or Unix/Linux, with one exception:

Right-Click: To invoke pop-up menus, and resize chart plot area, use **Ctrl+Click** instead of right-click when running in OS X.

Chapter 7. Drill-Down

Charts provide a great means to quickly analyze and comprehend information in a set of data. However, sometimes the data being viewed is either too voluminous or too complex to be easily rendered in a single chart. One way to display this type of data is to create a top-level chart that displays only summarized data. This enables users to click through specific data points or their corresponding labels in the chart legend to see underlying data. This is the concept of drill-down. It is possible to create a drill-down effect by creating the different charts individually and using hyperlinks to link the appropriate chart to the appropriate data point or to the corresponding label in the legend. However, this approach only works well if there are a small number of data points. For example, assume you have a top-level chart with 20 data points. For one level of drill-down you would have to create 20 charts, one for each data point. Suppose that the sub-level chart has 15 data points. To add another level of drill-down you would have to create 20 x 15 charts, or 300 separate charts. To avoid this problem, EspressChart contains several different built-in drill-down mechanisms that allow you to create only one chart for each level of drill-down. All of the drill-down options are available from the Drill-Down menu in Chart Designer.

7.1. 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.

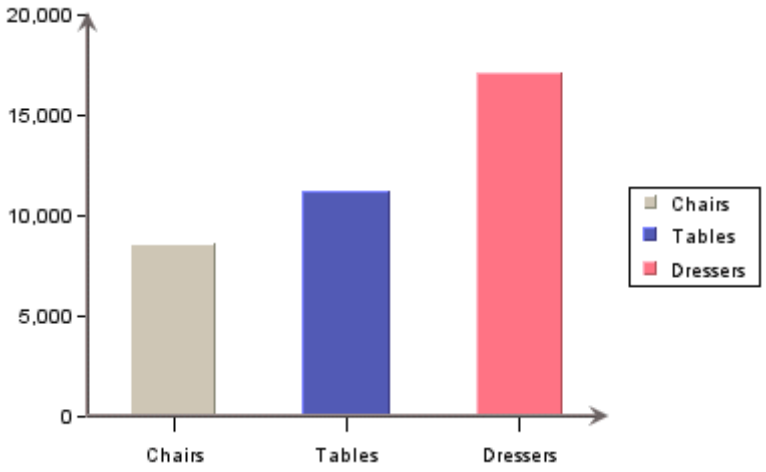
For example, assume you have the following table as the chart data:

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.

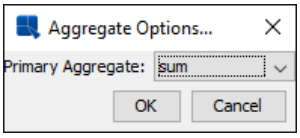
7.1.1. Adding Data Drill-Down

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:



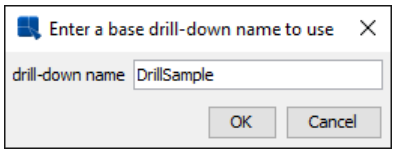
Top-Level Chart

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.



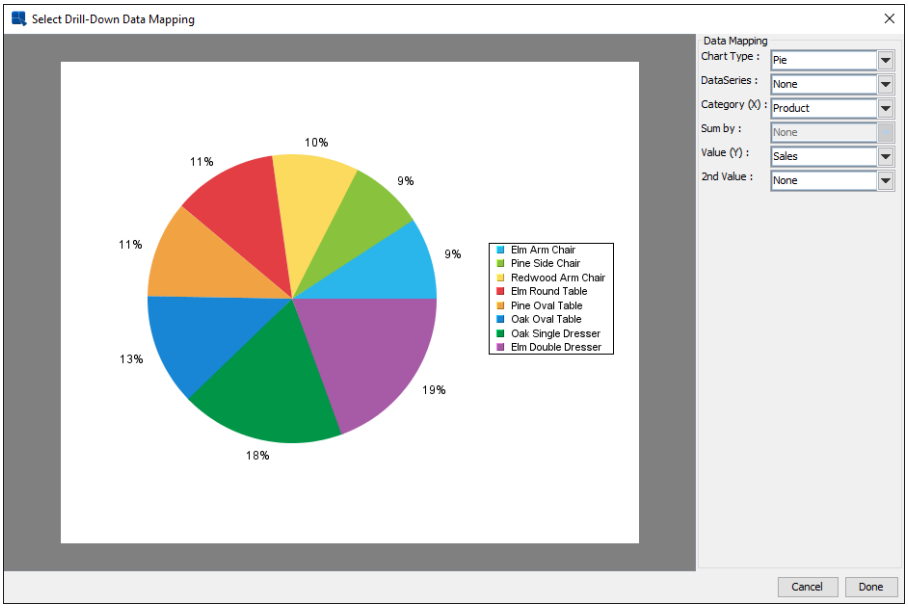
Aggregation Dialog

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.



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 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.

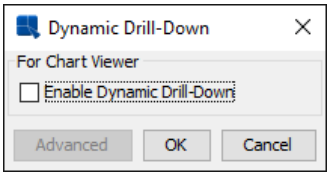
You can remove a level of the drill-down by navigating to that level and selecting Drill-Down → Remove This. The entire drill-down chart can also be deleted by selecting Drill-Down → Remove All. Selecting Drill-Down → Previous brings the drill-down chart to a higher level, which is the same as right 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.

7.2. Dynamic Data Drill-Down

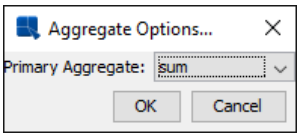
Usually, mapping and drill options for data drill-drown are fixed during design time. Dynamic drill-down is an additional option that allows you to select the mapping. The only thing specified during design time is the top-level chart and aggregation.

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.



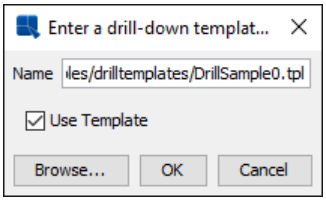
Enable Dynamic Drill-Down Dialog

Check the *Enable Dynamic Drill-Down* box and a new dialog will pop-up prompting you to select the aggregation.



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.



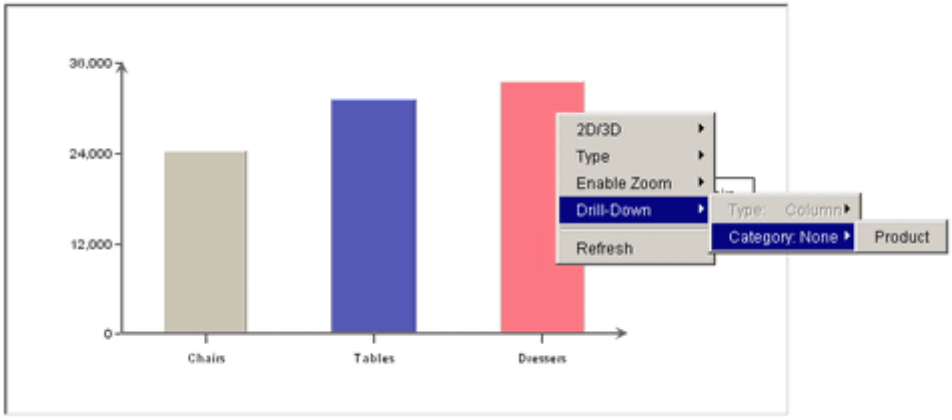
Select Dynamic Drill-Down Template Dialog

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 2.6 - 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

7.3. Parameter Drill-Down

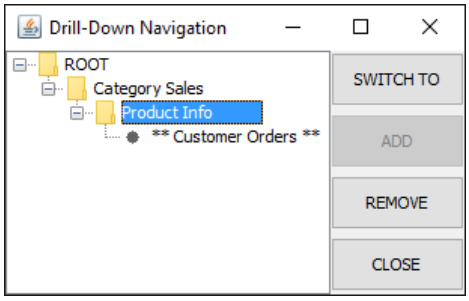
The third type of a drill-down you can use for charts is a parameter drill-down. Parameter drill-down is the most flexible implementation because you can relate the data between drill-down levels in any way you like. Specifically, you do not need to use the same value element for each level. Instead, 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.

7.3.1. 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.



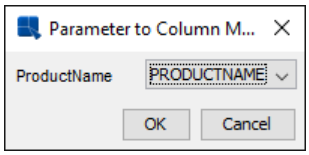
Parameter Drill-Down Navigation Window

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.



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.

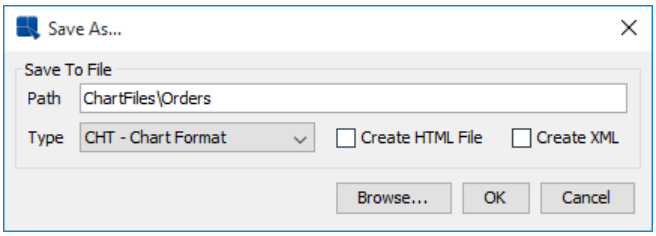
Chapter 8. Saving & Exporting Charts

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 2.6 - Run Applets in WebStart with JNLP file).

8.1. Saving Charts

To save the current chart, select the 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:



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 Chapter 3 - 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.

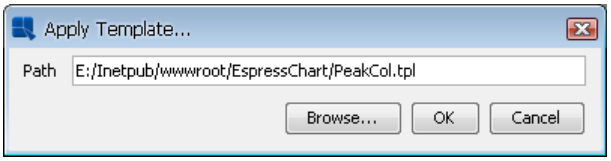
8.1.1. Working with Templates

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 10.5.2 - 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.



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.

8.1.2. Saving XML Templates

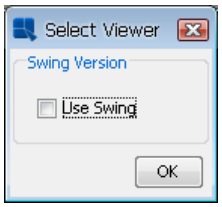
In addition to the two binary chart representations, you can also save the chart definitions in XML format. This option allows you to have a text-based chart template that can be used to modify chart properties outside of Chart Designer or the API.

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.

8.1.3. 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 Chapter 9 - 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 EspressChart installation. Before the file is written, you will be prompted to select which version of the Viewer that you would like to use.



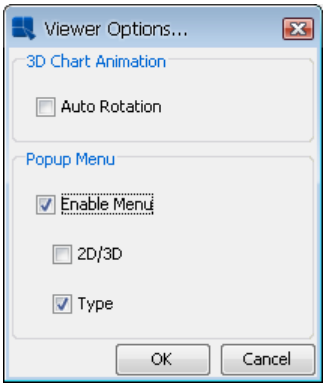
Select Viewer Dialog

EspressChart 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 EspressChart must be installed in a Web server, and you must access the page via http (i.e. `http://machinename:port/EspressChart/yourfile.html`) in order for it to load correctly.

8.1.3.1. 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.



Viewer Options Dialog

The following options are available:

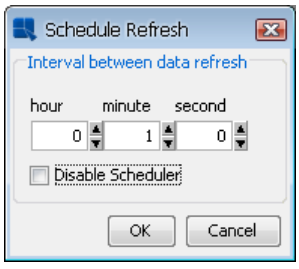
Auto Rotation: This will enable automatic rotation for three-dimensional charts. The charts will begin to rotate when the applet is loaded. The rotation can be stopped using the appropriate button on the navigation panel.

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.

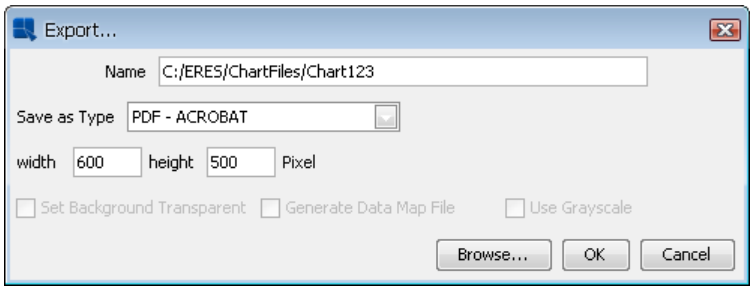


Schedule Refresh Dialog

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.

8.2. 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.



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 EspressChart 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.

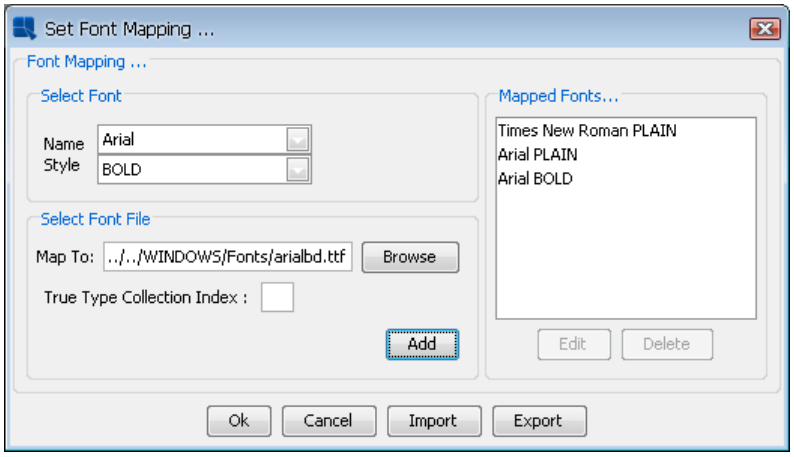
TXT This will generate a text data file containing the chart's data.

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.

8.2.1. PDF Font Mapping

EspressChart allows you to use any font on the system for charts. For most image exports (GIF/JPEG/PNG) fonts will written in the generated image. 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 chart.

To set font mapping for PDF export, select Format → Font Mapping. This will bring up a dialog allowing you to specify font files.



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.

8.2.1.1. 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.

Chapter 9. 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 2.6 - 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 2.6 - 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).

9.1. 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 2.6 - 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 2.6 - 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.

9.2. 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.

9.2.1. 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

9.2.2. 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>
```

9.2.3. 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.

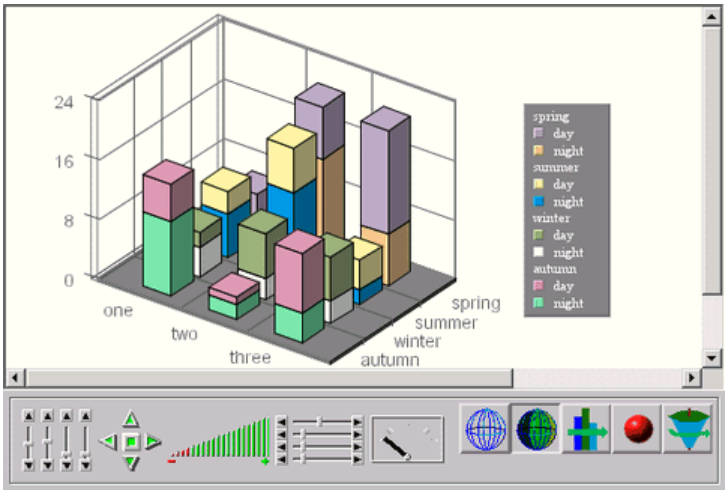
```
<applet-desc code = "quadbase.chartviewer.Viewer.class" width=640
 height=480>
```

```
      <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 "|".

9.3. 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

9.4. 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/Docs72/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.

9.5. 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"`**).

9.6. 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.

9.6.1. 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.

9.6.2. 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.

9.6.3. 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 6.8.2 - 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.

9.6.4. 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.

9.6.5. 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.

9.7. 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 `EspressChart/lib` directory. Call the following class when using the Swing viewer: `quadbase.chartviewer.swing.Viewer.class`.

Chapter 10. EspressChart Chart API

10.1. Introduction and Setup

In addition to designing and creating chart templates in the EspressChart Designer, EspressChart 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 class `QbChart` is used for creating a chart. 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 `EspressChart/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 EspressChart root directory>\lib
(For Solaris) export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<path to EspressChart
 root directory>/lib
```

10.2. Recommended Approach for Using Chart API

EspressChart 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 EspressChart 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 EspressChart 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.

10.3. Interaction with EspressManager

EspressChart is generally used in conjunction with EspressManager. The chart component connects to Espress-Manager 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 2.6 - 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 2.6 - Run Applets in WebStart with JNLP file)/application can connect to EspressManager 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 EspressChart (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 2.6 - 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 2.6 - 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:

10.4. Connecting to EspressManager

10.4.1. 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 `QbChart` and `QbChartDesigner`.

10.4.2. 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("EspressChart/servlet");
```

will connect to EspressManager running at `http://someMachine:somePortNumber/EspressChart/servlet`.

Please note that these methods exist in `QbChart` and `QbChartDesigner`.

10.5. 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 `<EspressChartInstall>/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 `<EspressChartInstall>/lib` directory.

Also, all the API examples will show the core code in the manual. To compile the examples, make sure the CLASSPATH includes `EspressAPI.jar` and `qblicense.jar`.

For more information on the API methods, please refer the API documentation [https://data.quadbase.com/Docs72/ec/help/apidocs/index.html].

10.5.1. 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
                      "OpeningChartTemplate.cht"); // template

      // Show chart in Viewer

      return chart;

}
```

Full Source Code [https://data.quadbase.com/Docs72/help/manual/code/src/OpeningChartTemplate.zip]

Exported Results [https://data.quadbase.com/Docs72/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.

10.5.1.1. 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.

10.5.1.1.1. 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.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;

 // Open the template
 QbChart chart = new QbChart(parent, // container
   "ObjectArray.cht", // template
   queryParams); // Query Parameters

 // Show chart in Viewer
 return chart;
}
```

Full Source Code [https://data.quadbase.com/Docs72/help/manual/code/src/ObjectArray.zip]

Exported Results [https://data.quadbase.com/Docs72/help/manual/code/export/ObjectArray.png]

10.5.2. Applying a Chart Template

When a QbChart object is created from scratch (see Appendix 10.B - 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 2.6 - 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/Docs72/help/manual/code/src/ApplyingChartTemplate.zip]

Exported Results [https://data.quadbase.com/Docs72/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.

10.5.3. 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.

10.5.3.1. 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 10.A.1 - Data from a Database).

The following example, which can be run as a applet(More in: Section 2.6 - 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/Docs72/help/manual/code/src/ChartSwitchToDatabase.zip]

Exported Results [https://data.quadbase.com/Docs72/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.

10.5.3.1.1. 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 2.6 - 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/Docs72/help/manual/code/src/ChartSwitchToDatabaseJNDI.zip]

Exported Results [https://data.quadbase.com/Docs72/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.

10.5.3.2. Data from a Data File (TXT/DAT/XML/CSV)

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 10.A.2 - Data from a Data file (TXT/DAT/XML/CSV)).

The following example, which can be run as a applet(More in: Section 2.6 - 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/Docs72/help/manual/code/src/ChartSwitchToDataFile.zip]

Exported Results [https://data.quadbase.com/Docs72/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.

10.5.3.3. 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 10.A.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 2.6 - 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/Docs72/help/manual/code/src/ChartSwitchToXMLData.zip]

Exported Results [https://data.quadbase.com/Docs72/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.

10.5.3.4. 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 10.A.5 - Data Passed in your Custom Implementation).

The following example, which can be run as a applet(More in: Section 2.6 - 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/Docs72/help/manual/code/src/ChartSwitchToCustomData.zip]

Exported Results [https://data.quadbase.com/Docs72/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.

10.5.3.5. 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 10.A.4 - Data Passed in an Array in Memory).

The following example, which can be run as a applet(More in: Section 2.6 - 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/Docs72/help/manual/code/src/ChartSwitchToArrayData.zip]

Exported Results [https://data.quadbase.com/Docs72/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.

10.5.4. Modifying Chart Attributes

EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/package-summary.html] and quadbase.ChartAPI [https://data.quadbase.com/Docs72/ec/help/apidocs/quadbase/ChartAPI/package-summary.html] packages.

10.5.4.1. 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.

10.5.4.2. 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), EspressChart 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 2.6 - 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/Docs72/help/manual/code/src/ChartSetPredefinedPattern.zip]

Exported Results [https://data.quadbase.com/Docs72/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/Docs72/ec/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`.

10.5.4.3. 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 2.6 - 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/Docs72/help/manual/code/src/ChartSetCustomPattern.zip]

Exported Results [https://data.quadbase.com/Docs72/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.

10.5.4.4. 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.

10.5.4.5. 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 2.6 - 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/Docs72/help/manual/code/src/ZoomChart.zip]

Exported Results [https://data.quadbase.com/Docs72/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.

10.5.4.6. 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
```

10.5.5. Exporting the Chart

Any charts included in the chart can be exported into JPEG, GIF and PNG formats as well; although by default, the charts in the chart are exported as JPEG's. The format for the chart can be changed, when creating the `ChartChartObject` object, by specifying the image type use the method `setImageType(int)`.

The EspressChart 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 EspressChart 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 EspressChart Viewer or EspressChart 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/Docs72/help/manual/code/src/ExportChart.zip]

Exported Results [https://data.quadbase.com/Docs72/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.

10.5.5.1. Record File Exporting

EspressChart 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>);
```

10.5.5.2. Streaming Charts

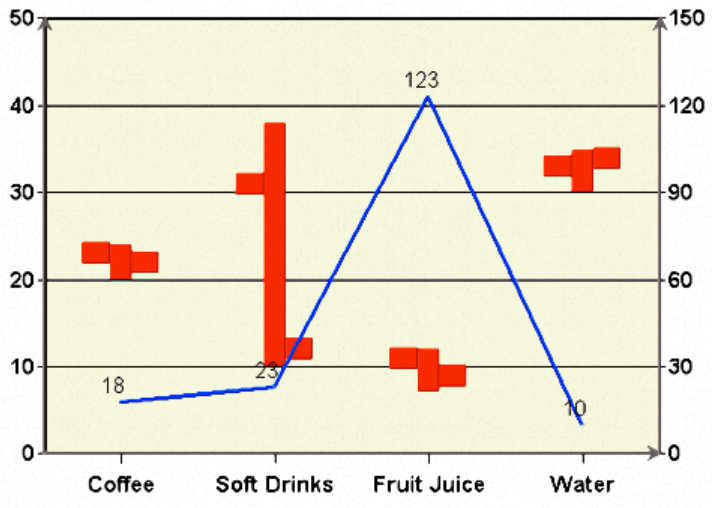
In addition to exporting to local drives, charts can also be exported as a byte stream and streamed directly to a web browser. Note that server side code typically exports the image only in a binary format. You are responsible for creating a wrapper (i.e., DHTML/HTML content) around the exported image.

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/Docs72/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

10.5.6. Calling Chart Designer from Chart API

Chart Designer can be called from Chart API in either an application or an applet. Depending on the code, you can pass in parameters to open up Chart Designer with a specified template file or a specified data source or other parameters.

To call Chart Designer from Chart API, you must:

- make sure that EspressManager is up and running;
- add `EspressAPI.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 and backgroundimages directories to the working directory or pass the location (of those directories) in the prper constructor in your code;

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
 QbCharttDesigner 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 QbChartDesigner class
```

Given below are the different ways Chart Designer can be called via Chart API

10.5.6.1. Specify a Chart Template

You can open Chart Designer with a specified chart template file. This allows the end users to create their own custom charts in Chart Designer GUI and then view the finished chart.

The following constructor is used:

```
QbChartDesigner(Object parent, String chtFile, String[] imagesPath);
```

Given below is an example of calling Chart Designer with a specified .cht file:

```
public void doChartDesignerApplet(Object parent)  throws Exception {

 // Connect to EspressManager
 QbChartDesigner.setServerAddress("127.0.0.1");
 QbChartDesigner.setServerPortNumber(22071);

 // these folders are located in the <InstallDir> directory. Change the
 paths to absolute paths, or paths relative to this application.
 String[] imagesPath = {"images", "backgroundImages"};

 // Create a new QbReportDesigner instance
 designer = new QbChartDesigner(parent, "CDWithTemplateFile.cht",
 imagesPath);

 // Start Designer
```

```
 designer.setVisible(true);
}
```

Full Source Code [https://data.quadbase.com/Docs72/help/manual/code/src/CDWithTemplateFile.zip]

Note that the above code can be used as both an application and an applet.

When the user runs this code, Chart Designer is launched with the chart template you specified.

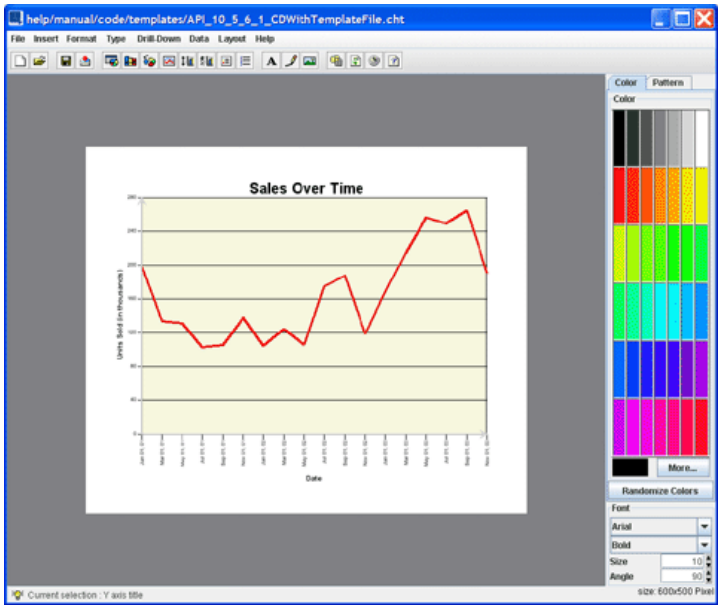
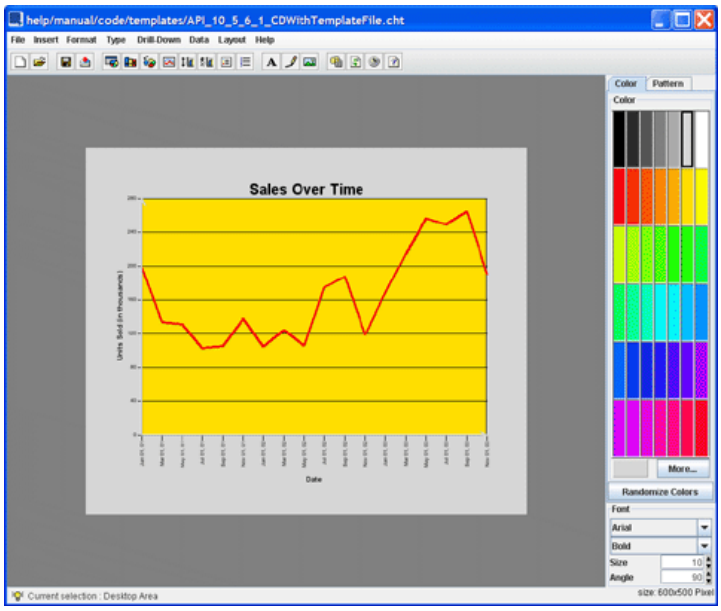


Chart Designer with Template

The user can then customize the chart and save the results.



End User Customization

10.5.6.2. Specify a Data Registry

You can open Chart 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 charts in Chart Designer GUI and then view the finished chart.

The following constructor is used:

```
QbChartDesigner(Object parent, String nameOfXMLFile, boolean newChart,
 String[] imagesPath);
```

Given below is an example of calling Chart Designer with a specified .xml file:

```
public void doChartDesignerWithDataRegApplet(Object parent) throws Exception
 {
 // Connect to EspressManager
 QbChartDesigner.setServerAddress("127.0.0.1");
 QbChartDesigner.setServerPortNumber(22071);

 // these folders are located in the <InstallDir> directory. Change the
 paths to absolute paths, or paths relative to this application.
 String[] imagesPath = {"images", "backgroundImages"};

 // QbChartDesigner (Object, name of XML file, start from Data Registry?,
 images directory)
 designer = new QbChartDesigner(parent, "DataRegistry/sample.xml", true,
 imagesPath);

 designer.setVisible(true);

}
```

Full Source Code [https://data.quadbase.com/Docs72/help/manual/code/src/CDWithDataRegistry.zip]

Note that the above code can be used as both an application and an applet.

10.5.6.3. Specify a DBInfo Object (Database Information)

You can open Chart Designer and have it starting at the *Select Chart Type* wizard window (with a specified `DBInfo` object). This allows the end users to create their own custom charts in Chart Designer GUI and then view the finished chart.

The following constructor is used:

```
QbChartDesigner(Object parent, DBInfo databaseInformation, QueryInParamSet
 inset boolean newChart, String[] imagesPath);
```

Given below is an example of calling Report Designer with a specified `DBInfo` object:

```
public QbChartDesigner designer;
String url = "jdbc:hsqldb:help/examples/DataSources/database/woodview";
String driver = "org.hsqldb.jdbcDriver";
String username = "sa";
String password = "";
String query = "SELECT * FROM Products";

public void doChartDesignerWithDBInfoApplet(Object parent) throws Exception
 {
 // Connect to EspressManager
 QbChartDesigner.setServerAddress("127.0.0.1");
 QbChartDesigner.setServerPortNumber(22071);

 // these folders are located in the <InstallDir> directory. Change the
 paths to absolute paths, or paths relative to this application.
 String[] imagesPath = {"images", "backgroundImages"};
```

```
 // QbReportDesigner (component, Database Information, start from Select
 Chart wizard, Parameter Information, Images Path
 DBInfo dbInfo = new DBInfo(url, driver, username, password, query);
 designer = new QbChartDesigner(parent, dbInfo, null, true, imagesPath);

 designer.setVisible(true);

}
```

Full Source Code [https://data.quadbase.com/Docs72/help/manual/code/src/CDWithDBInfo.zip]

Note that the above code can be used as both an application and an applet.

10.5.6.4. 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 their own custom SQL queries in Query Builder GUI and save them.

Given below is an example of calling Query Builder with a specified `DBInfo` object:

```
public void doQueryBuilderApplet(Object parent, String sqlName) {

      // Connect to EspressManager
      QbChartDesigner.setServerAddress("127.0.0.1");
      QbChartDesigner.setServerPortNumber(22071);

      queryMain = new QbQueryBuilder(parent, 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 QbChartDesigner(applet, dbInfo, queryMain.getInSet(),
 true, imagesPath);
```

```
      designer.setVisible(true);

}
```

Full Source Code [https://data.quadbase.com/Docs72/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
      QbChartDesigner.setServerAddress("127.0.0.1");
      QbChartDesigner.setServerPortNumber(22071);

      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() {

      queryMain.getBuilder().setVisible(false);
      DBInfo dbInfo = new DBInfo(url, driver, username, password,
 queryMain.getSQL());
      designer = new QbChartDesigner(applet, dbInfo, queryMain.getInSet(),
 true, imagesPath);
      designer.setVisible(true);

}

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/Docs72/help/manual/code/src/QBWithDBInfo2.zip]

Note that the above code can be used as both an application and an applet.

10.5.6.5. Open Chart Designer with pre-set Directories

You can also set the directories that charts 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 Chart Designer by specifying the root directory for browsing:

```
public void doChartDesignerApplet(Object parent) throws Exception {
 QbChartDesigner.setServerAddress("127.0.0.1");
 QbChartDesigner.setServerPortNumber(22071);

 // these folders are located in the <InstallDir> directory. Change the
 paths to absolute paths, or paths relative to this application.
 String[] imagesPath = {"images", "backgroundImages"};

 // IMPORTANT: put the CDWithPreSetDirectories.cht file to the <InstallDir>/
 directory (for example: C:/EspressChart/CDWithPreSetDirectories.cht)
 designer = new QbChartDesigner(parent, "CDWithPreSetDirectories.cht",
 imagesPath);
```

```
 designer.setRootDirectoryForBrowse(".");

 designer.setVisible(true);
}
```

Full Source Code [https://data.quadbase.com/Docs72/help/manual/code/src/CDWithPreSetDirectories.zip]

Note that the above code can be used as both an application and an applet.

In addition to the above approach, you can also use the following method to get the default directories used by Chart Designer:

```
public BrowseDirectories getBrowseDirectories();
```

You can then use the methods in BrowseDirectories [https://data.quadbase.com/Docs72/ec/help/apidocs/quadbase/common/util/BrowseDirectories.html] to get the default location of the directories for the different browse dialogs.

10.5.6.6. Open Chart 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(Object parent) throws Exception {
 QbChartDesigner.setServerAddress("127.0.0.1");
 QbChartDesigner.setServerPortNumber(22071);

 // these folders are located in the <InstallDir> directory. Change the
 paths to absolute paths, or paths relative to this application.
 String[] imagesPath = {"images", "backgroundImages"};

 designer = new QbChartDesigner(parent, "CDWithModifiedBars.cht",
 imagesPath);

 // Add a new menu item
 JMenuBar menuBar = designer.getChartMenuBar();
 JMenu fileMenu = menuBar.getMenu(0);
 newItem = new JMenuItem("Hello World");
 newItem.addActionListener(this);
 fileMenu.insert(newItem, 7);

 // Remove toolbar buttons
 JToolBar designBar = designer.getChartToolBar();
 designBar.remove(5);    // Remove Separator
 designBar.remove(4);    // Remove Export
 designer.setSaveOnExitEnabled(false);  // Do not prompt to save the chart
 if unsaved on exiting Designer
 designer.setVisible(true);
}

public void actionPerformed(ActionEvent e) {

 /*** save report **********/
 designer.save("CDWithModifiedBars_Temp.cht");
 /**** 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/Docs72/help/manual/code/src/CDWithModifiedBars.zip]

Note that the above code can be used as both an application and an applet.

10.6. API Only Features

While EspressChart 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`.

10.6.1. 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.

10.6.1.1. 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.

10.6.1.1.1. Customizable Message for No-Data-In-Plot

EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/INoDataToPlotMessage.html]).

10.6.1.1.2. Set Chart to Fit Canvas

EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/ICanvas.html]).

10.6.1.1.3. 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/Docs72/ec/help/apidocs/quadbase/util/ICanvas.html]).

10.6.1.1.4. Applying Different Graphics Rendering

EspressChart 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.quad-base.com/Docs72/ec/help/apidocs/quadbase/ChartAPI/QbChart.html]).

This functionality is only available for stand-alone charts.

10.6.1.1.5. Chart Plot Position

EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/IPlot.html]).

10.6.1.1.6. Drawing Multiple Charts in same Plot

EspressChart 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
  "AddOnChart1.cht"); // template

// Open other two templates
QbChart chart2 = new QbChart(parent, "AddOnChart2.cht");
QbChart chart3 = new QbChart(parent, "AddOnChart3.cht");
```

```
primaryChart.setAddOnChart(new QbChart[] { chart2, chart3 });
```

Full Source Code [https://data.quadbase.com/Docs72/help/manual/code/src/AddOnChart.zip]

Exported Results [https://data.quadbase.com/Docs72/help/manual/code/export/AddOnChart.png]

Please refer to the online API documentation for more information (quadbase.ChartAPI.QbChart [https://data.quadbase.com/Docs72/ec/help/apidocs/quadbase/ChartAPI/QbChart.html]).

10.6.1.2. 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.

10.6.1.2.1. Modify Hint Box

EspressChart 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
  "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/Docs72/help/manual/code/src/ModifyHintBox.zip]

Exported Results [https://data.quadbase.com/Docs72/help/manual/code/export/ModifyHintBox.png]

Please refer to the online API documentation for more information (quadbase.util.IHintBoxInfo [https://data.quadbase.com/Docs72/ec/help/apidocs/quadbase/util/IHintBoxInfo.html]).

This functionality is only available for stand-alone charts.

10.6.1.2.2. Data and Hyperlink Hint Box Offset

EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/IHint.html]).

10.6.1.2.3. Hint Box Border Color

EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/IHint.html]).

10.6.1.2.4. Customize Image Map Hint Box

EspressChart 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
  "CustomizeImageMapHintBox.cht"); // template

chart.setImageMapDataHintBoxHandle(new customizeImageMap());

try {
 // Export chart to image and map file
 chart.export(QbChart.PNG, "CustomizeImageMapHintBox", "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/Docs72/help/manual/code/src/CustomizeImageMapHintBox.zip]

Exported Map [https://data.quadbase.com/Docs72/help/manual/code/export/CustomizeImageMapHintBox.map]

Exported Image [https://data.quadbase.com/Docs72/help/manual/code/export/CustomizeImageMapHintBox.html]

Please refer to the online API documentation for more information (quadbase.util.ICustomizeImageMapDataHintBox [https://data.quadbase.com/Docs72/ec/help/apidocs/quadbase/util/ICustomizeImageMapDataHintBox.html]).

10.6.1.3. 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.

10.6.1.3.1. Annotation with Symbol

EspressChart 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 EspressChart.

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
  "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(.65f, .7f);
anno.setRelativePosition(newPosition);
set.addAnnotation(anno);
```

Full Source Code [https://data.quadbase.com/Docs72/help/manual/code/src/AnnotationWithSymbol.zip]

Exported Image [https://data.quadbase.com/Docs72/help/manual/code/export/AnnotationWithSymbol.png]

Please refer to the online API documentation for more information (quadbase.util.IAnnotationSet [https://data.quadbase.com/Docs72/ec/help/apidocs/quadbase/util/IAnnotationSet.html] and quadbase.util.IAnnotation [https://data.quadbase.com/Docs72/ec/help/apidocs/quadbase/util/IAnnotation.html]).

10.6.1.3.2. Set Relative Shift of Annotation Border

EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/IAnnotation.html]).

10.6.1.3.3. Set Legend Title

EspressChart 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/Docs72/help/manual/code/src/LegendTitle.zip]

Exported Image [https://data.quadbase.com/Docs72/help/manual/code/export/LegendTitle_exported.png]

Please refer to the online API documentation for more information(quadbase.util.ILegend [https://data.quadbase.-com/Docs72/ec/help/apidocs/quadbase/util/ILegend.html]).

10.6.1.3.4. Set LegendLineSpacing

EspressChart 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/Docs72/help/manual/code/src/LegendSpacing.zip]

Exported Image [https://data.quadbase.com/Docs72/help/manual/code/export/LegendSpacing_exported.png]

Please refer to the online API documentation for more information(quadbase.util.ILegend [https://data.quadbase.-com/Docs72/ec/help/apidocs/quadbase/util/ILegend.html]).

10.6.1.4. 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.

10.6.1.4.1. Ticker Label Replacement

EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/IAxis.html]).

10.6.1.4.2. Customizable Data Top Label

EspressChart 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
  "CustomizeDataTopLabel.cht"); // template
```

```
chart.gethDataPoints().setDataLabelInfo(new customizeDataTopLabel());

...

class customizeDataTopLabel implements IDataLabelInfo {

 static final long serialVersionUID = 1;

 public String getDataLabel(PickData pickData, String originalDataLabel) {
  return (pickData.toString());
 }
}
```

Full Source Code [https://data.quadbase.com/Docs72/help/manual/code/src/CustomizeDataTopLabel.zip]

Exported Image [https://data.quadbase.com/Docs72/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/Docs72/ec/help/apidocs/quadbase/util/IDataLabelInfo.html]).

10.6.1.4.3. Show Axis as Date/Time/Timestamp

EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/IAxis.html]).

10.6.1.4.4. 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. EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/IDataPointSet.html]).

10.6.1.4.5. Drawing Data Points above Horizonval/Vertical/Trend Lines

EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/ILinePropertySet.html]).

10.6.2. 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.

10.6.2.1. Getting the Coordinates

EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/IHint.html]).

10.6.2.2. Set Data Limit at Axis Scale

When a manual axis is applied, EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/IDataPointSet.html]).

10.6.2.3. Set Null Data as Zero

EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/IDataPointSet.html]).

10.6.2.4. Additional Trend Line Options

EspressChart 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/Docs72/help/manual/code/src/AdditionalTrendLine.zip]

Exported Image [https://data.quadbase.com/Docs72/help/manual/code/export/AdditionalTrendLine.png]

Please refer to the online API documentation for more information (quadbase.util.ITrendLine [https://data.quadbase.com/Docs72/ec/help/apidocs/quadbase/util/ITrendLine.html]).

10.6.2.5. Adding Multiple Control Lines to Stack Type Chart

EspressChart 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/Docs72/help/manual/code/src/MultipleControlLines.zip]

Exported Image [https://data.quadbase.com/Docs72/help/manual/code/export/MultipleControlLines.png]

Please refer to the online API documentation for more information (quadbase.util.IControlLine [https://data.quadbase.com/Docs72/ec/help/apidocs/quadbase/util/IControlLine.html]).

10.6.2.6. Additional Data Line Options

EspressChart 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/Docs72/help/manual/code/src/DataLineExtraOptions.zip]

Exported Image [https://data.quadbase.com/Docs72/help/manual/code/export/SalesByCustomerID_exported.png]

Please refer to the online API documentation for more information (quadbase.util.IDataLineSet [https://data.quadbase.com/Docs72/ec/help/apidocs/quadbase/util/IDataLineSet.html]) and (quadbase.util.IHorzVertLine [https://data.quadbase.com/Docs72/ec/help/apidocs/quadbase/util/IAnnotation.html]).

10.6.3. 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.

10.6.3.1. 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.

10.6.3.1.1. Color Separator

EspressChart 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/Docs72/help/manual/code/src/ColorSeparator.zip]

Exported Image [https://data.quadbase.com/Docs72/help/manual/code/export/ColorSeparator.png]

Please refer to the online API documentation for more information (quadbase.util.IDataPointSet [https://data.quadbase.com/Docs72/ec/help/apidocs/quadbase/util/IDataPointSet.html]).

10.6.3.1.2. Disabling Shadow

EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/IDataPointSet.html]).

10.6.3.2. 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.

10.6.3.2.1. Drawing Pie Slices Clockwise/Counter Clockwise

EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/IDataPointSet.html]).

10.6.3.2.2. Pie Border for 0% and 100% Slices

EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/IPiePropertySet.html]).

10.6.3.2.3. Customize Separator between Category and Percent Value Strings in Pie Legend

EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/IPiePropertySet.html]).

10.6.3.2.4. Pie Border Color Customizable

EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/IPiePropertySet.html]).

10.6.3.3. 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.

10.6.3.3.1. Line Area

EspressChart 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/Docs72/help/manual/code/src/LineArea.zip]

Exported Image [https://data.quadbase.com/Docs72/help/manual/code/export/LineArea.png]

Please refer to the online API documentation for more information (quadbase.util.ILinePropertySet [https://data.quadbase.com/Docs72/ec/help/apidocs/quadbase/util/ILinePropertySet.html]).

10.6.3.4. 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.

10.6.3.4.1. Show Series in Top Label

EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/IDataPointSet.html]).

10.6.3.4.2. Drawing Order

EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/IDataPointSet.html]).

10.6.3.4.3. Scatter Chart Cube Width

EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/IDataPointSet.html]).

10.6.3.5. 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.

10.6.3.5.1. Multiple Axes Titles

EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/IAxis.html]).

10.6.3.6. 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.

10.6.3.6.1. Control Area Scale Labels

EspressChart 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.quad-base.com/Docs72/ec/help/apidocs/quadbase/util/ControlRange.html]).

10.6.3.7. 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.

10.6.3.7.1. Changing Candle Stick Color

EspressChart 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.quad-base.com/Docs72/ec/help/apidocs/quadbase/util/IDataPointSet.html]).

10.6.3.7.2. Changing Candle Stick Wicker Width

EspressChart 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.quad-base.com/Docs72/ec/help/apidocs/quadbase/util/IDataPointSet.html]).

10.6.3.8. 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.

10.6.3.8.1. Heat Map

EspressChart 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/Docs72/help/manual/code/src/HeatMap.zip]

Exported Image [https://data.quadbase.com/Docs72/help/manual/code/export/HeatMap.png]

Please refer to the online API documentation for more information (quadbase.util.ColorSpectrum [https://data.quadbase.com/Docs72/ec/help/apidocs/quadbase/util/ColorSpectrum.html] and quadbase.util.I3DPropertySet [https://data.quadbase.com/Docs72/ec/help/apidocs/quadbase/util/I3DPropertySet.html]).

10.6.4. 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.

10.6.4.1. BufferedImage or Frame

EspressChart allows the choice of using either `java.awt.image.BufferedImage` or `java.awt.Frame` during export to improve performance. By default, EspressChart 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/Docs72/ec/help/apidocs/quadbase/ChartAPI/QbChart.html]).

This functionality is only available for stand-alone charts.

10.6.4.2. Chart Generation Order

EspressChart 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
  "ChartGeneration.cht"); // template

chart.setChartGraphics(new chartGenerationGraphics());

...
```

```
class chartGenerationGraphics implements IChartGraphics {

 static final long serialVersionUID = 1;

 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/Docs72/help/manual/code/src/ChartGeneration.zip]

Exported Image [https://data.quadbase.com/Docs72/help/manual/code/export/ChartGeneration.png]

Please refer to the online API documentation for more information (quadbase.util.IChartGraphics [https://data.quadbase.com/Docs72/ec/help/apidocs/quadbase/util/IChartGraphics.html]).

This functionality is only available for stand-alone charts.

10.6.5. Viewer

The features, shown below, deal with the Viewer when viewing the chart in either a java application or java applet(More in: Section 2.6 - Run Applets in WebStart with JNLP file). Each feature below shows additional functionality available to Chart Viewer.

10.6.5.1. Call Back Mechanism

EspressChart 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
  "CallBack.cht"); // template

chart.addActionListener(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/Docs72/help/manual/code/src/CallBack.zip]

Exported Image [https://data.quadbase.com/Docs72/help/manual/code/export/CallBack.png]

This functionality is only available for stand-alone charts.

10.6.5.2. Disable/Enable Tool Tips Text

EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/I3DControlPanel.html]).

10.6.5.3. Canvas Area

EspressChart 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/Docs72/ec/help/apidocs/quadbase/util/ICanvas.html]).

10.7. 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/Docs72/ec/help/apidocs/quadbase/util/IPopupMenu.html]).

10.8. Javadoc

Javadoc for the entire API is provided along with EspressChart. The API covers both the Chart and the Charting API. It is located at Quadbase website [https://data.quadbase.com/Docs72/ec/help/apidocs/index.html].

10.9. Swing Version

1.1 JFC/Swing versions of the EspressChart charting API is also available. For more details, please refer to quadbase.ChartAPI.swing package [https://data.quadbase.com/Docs72/ec/help/apidocs/quadbase/ChartAPI/swing/package-summary.html].

10.10. Summary

The EspressChart 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 2.6 - 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 EspressChart Designer.

10.A. Getting the Chart Data

The two appendices in this Chapter (Appendix 10.A - Getting the Chart Data and Appendix 10.B - 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. 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 10.B - Creating the Chart discusses the dimensions, the chart type, the column mapping, and the actual creation of the chart. Appendix 10.B - Creating the Chart also contains working examples for you to try.

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.

10.A.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 EspressChart. 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.

10.A.1.1. JNDI

EspressChart 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.

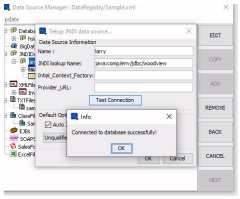
To connect to a JNDI data source, you must have a data source deployed in your application server. You must also provide the **`INITIAL_CONTEXT_FACTORY`** and **`PROVIDER_URL`** to successfully make the connection. Please note that when connecting to a JNDI data source deployed in Tomcat, the **`INITIAL_CONTEXT_FACTORY`** 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);
```

10.A.1.2. JNDI example using Tomcat and EspressManager running from servlet



Woodview database connection

10.A.2. Data from a Data file (TXT/DAT/XML/CSV)

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 Chart API, can read it.

A `.csv` file is similar to TXT or DAT files, but with one key difference: the first line contains the field names, and all subsequent lines represent the records.

```
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
```

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 4.3 - 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 EspressManager, relative paths should be in respect to the EspressManager working 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`**).

10.A.3. Data from a XML Data Source

In addition to the above, EspressChart 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/Docs72/ec/help/apidocs/quadbase/common/util/internal/XMLFileQueryInfo.html] constructor:

```
XMLFileQueryInfo xmlInfo = new XMLFileQueryInfo(xmlByteArray, fields,
 xmlCondition, fields);
```

10.A.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/Docs72/ec/help/apidocs/quadbase/util/IResultSet.html] (defined in quadbase.util [https://data.quadbase.com/Docs72/ec/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 EspressChart. The simplest implementation is provided by the class `DbData` (Other classes that provide an `IResultSet` implementation are QueryResultSet [https://data.quadbase.com/Docs72/ec/help/apidocs/quadbase/util/QueryResultSet.html] and StreamResultSet [https://data.quadbase.com/Docs72/ec/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);
```

10.A.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/Docs72/ec/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 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/Docs72/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/Docs72/ec/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/Docs72/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/Docs72/ec/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/Docs72/help/manual/code/src/CustomParamClassFile.zip]

10.A.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/Docs72/ec/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/Docs72/ec/help/apidocs/quadbase/util/SimpleSpreadSheet.html] (defined in the quadbase.util [https://data.quadbase.com/Docs72/ec/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

EspressChart 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);
```

10.A.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).

Provide the connection information 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/Docs72/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 2.6 - 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);
```

10.A.8. Data from a SOAP Data Source

EspressChart allows to retrieve data from SOAP services. SOAP data sources are similar to XML data sources. The main difference is that the XML files are transfered in a SOAP message. To use the SOAP data source, you need a SOAP service, which will be sending the XML files. This service can be written in any programming language.

The request SOAP message has no parameters and expects a SOAP response in this form:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance">
```

```
      <soapenv:Body>
            <XMLTYPE>QUADBASE</XMLTYPE> </soapenv:Body>

</soapenv:Envelope>
```

The message has only one body element – `XMLTYPE`. Its value specifies the type of the XML file. Possible values are:

QUADBASE XML is in Quadbase format

DTD XML with DTD schema

XSD XML with XML Schema

The XML file and schema (if used) are sent as MIME attachment of the SOAP message. If the QUADBASE XMLTYPE is used, the SOAP message has to have one attachment – the XML file in Quadbase format. If DTD or XSD XMLTYPE is used, the message has to have two attachments – the first is the XML file and the second is the DTD or XSD file. The MIME type of these attachments is **`text/xml`**.

This is the constructor for creating a chart from a SOAP data source. The XML has to be in the Quadbase format:

```
QbChart(java.applet.Applet applet, int dimension, int chartType,
 java.lang.String SOAPURL, java.lang.String serviceName, java.lang.String
 methodName, IColumnMap cmap, java.lang.String template, boolean
 doTransposeData, boolean[] transposeCol)
```

10.A.8.1. Example SOAP Service

EspressChart provides an example SOAP service. Its source code can be found in `<EC-installation-directory>/help/examples/soap/SOAPService.java`. To use the example, please edit the source code and replace all the occurrences of `<ECInstall>` with your EspressChart installation directory. Then compile the file (you need to include all the jar files from the `<EC-installation-directory>/lib` directory in your classpath to do that).

Next, you have to deploy this SOAP service. The instructions below shows how to deploy it on the Apache Tomcat. The <EC> refers to the EspressChart installation directory.

1. Download Tomcat 11 or higher from the Apache web sites and install it. The Tomcat installation directory is further referenced as <TOMCAT>.

2. Copy the following files from `<EC>/lib` to `<TOMCAT>/common/lib`: `activation.jar`, `axis.jar`, `commons-discovery-0.2.jar`, `commons-logging-1.0.4.jar`, `jaxrpc.jar`, `log4j-1.2.8.jar`, `mail.jar`, `saaj.jar`, `wsdl4j-1.5.1.jar`.

3. Copy the following files from `<EC>/help/examples/soap/` to `<TOMCAT>/webapps/axis/WEB-INF/classes/quadbase/soap` directory (you have to create this directory structure): `deployDS.wsdd`, `SOAPService.class`, `undeployDS.wsdd`.

4. Copy the `<EC>/help/examples/soap/web.xml` file to the `<TOMCAT>/webapps/axis/WEB-INF/` directory.

5. Start Tomcat (using `<TOMCAT>/bin/startup.bat`)

6. Run the following command in the `<TOMCAT>/webapps/axis/WEB-INF/classes/quadbase/soap/`, having all the JAR files from the step 2 in your classpath. It will deploy the SOAPService.

```
java org.apache.axis.client.AdminClient deployDS.wsdd
```

If your Tomcat is not running at default port (8080), you can use **-p** argument to specify different port.

Three web services are deployed in this example: “SOAPService-fetchData” returns XML in Quadbase format, “SOAPService-fetchData2” returns XML and a DTD file and “SOAPService-fetchData3” returns XML and a schema file (XSD). The necessary connection informations for the example are below.

Server URL is **http://machine:port/axis/services/SOAPService** for all three services. Please replace machine and port with machine name and port Tomcat is running on (the default port is 8080). If Tomcat is not running on the local machine, please remember to modify the URL links within the XML files.

Service lookup name and method lookup name for all the examples are listed below.

- service lookup name: **SOAPService**
  method lookup name: **fetchData**

  check *XML is in Quadbase format*

  This service sends XML in Quadbase Format. You can use the data source in the same way as XML File data source.

- service lookup name: **SOAPService**
  method lookup name: **fetchData2**

  uncheck *XML is in Quadbase format*

  This one uses a DTD file. You can add XML query using this data source and use the query in the same way as XML File Query.

- service lookup name: **SOAPService**
  method lookup name: **fetchData3**

  uncheck *XML is in Quadbase format*

  This one uses a Schema file. You can add XML query using this data source and use the query in the same way as XML File Query.

10.A.9. Data from multiple Data Sources

EspressChart 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`.

10.A.10. 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.

EspressChart 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.

10.A.11. Transposing Data

EspressChart 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:

10.A.11.1. 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

Product	January	February	March
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/Docs72/help/manual/code/src/NonTranspose.zip]

Exported Results [https://data.quadbase.com/Docs72/help/manual/code/export/NonTranspose.png]

Please note that the column index passed in `IColumnMap` would refer to the new dataset.

10.A.11.2. 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

Category	Product	ColumnLabel	Value
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/Docs72/help/manual/code/src/Transpose.zip]

Exported Results [https://data.quadbase.com/Docs72/help/manual/code/export/Transpose.png]

Please note that the column index passed in `IColumnMap` would refer to the new dataset.

10.B. 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 `<EspressChartInstall>/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 `<EspressChartInstall>/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)
Heatmap Chart	`QbChart.HEATMAP`(Two-Dimensional Only)

For more information on the different chart types, please refer to the Chapter 5 - 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 Chapter 5 - Chart Types and Data Mapping.

To define the Column Mapping for the chart template, the `ColInfo` class (located in the `quadbase.ChartAPI` 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.

10.B.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.

10.B.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

Order #	Product	Units Ordered	Units Shipped
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;
```

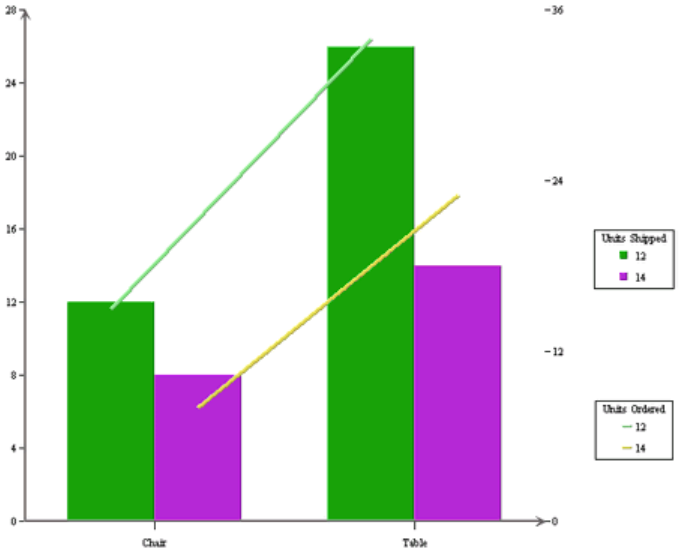
10.B.1.2. Creating the Chart

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/Docs72/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.

10.B.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.

10.B.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;
```

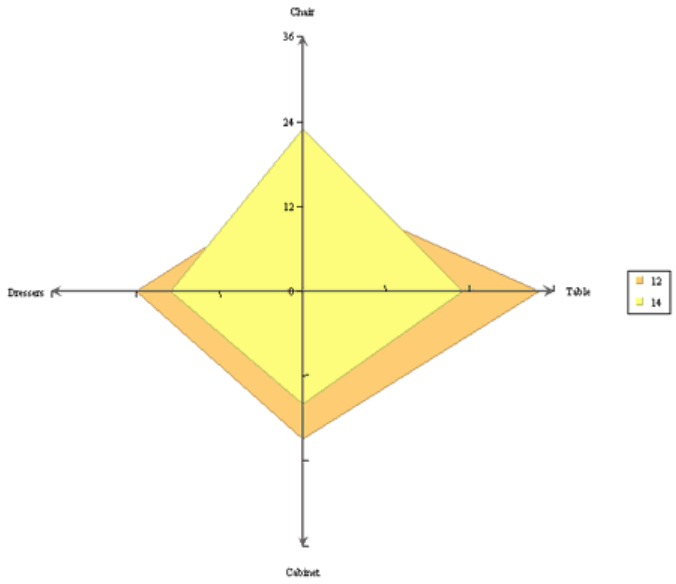
10.B.2.2. Creating the Chart

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/Docs72/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.

10.B.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.

10.B.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;
```

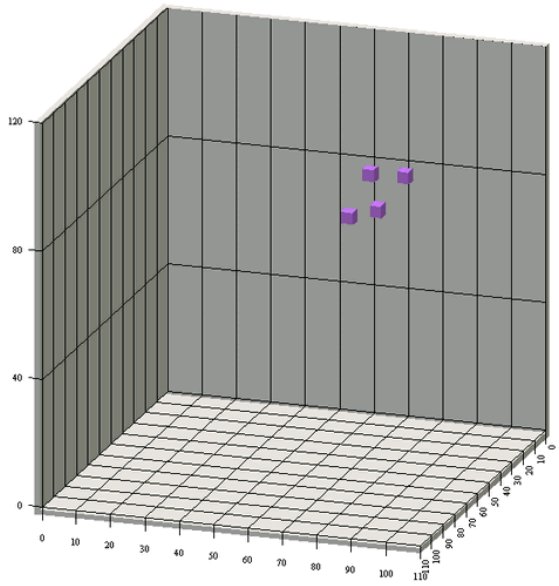
10.B.3.2. Creating the Chart

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/Docs72/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.

10.B.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.

10.B.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;
```

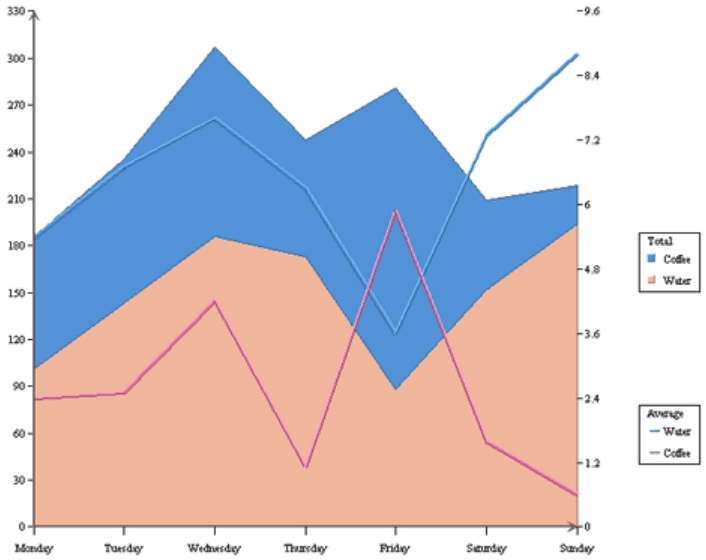
10.B.4.2. Creating the Chart

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/Docs72/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.

10.B.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.

10.B.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;
```

10.B.5.2. Creating the Chart

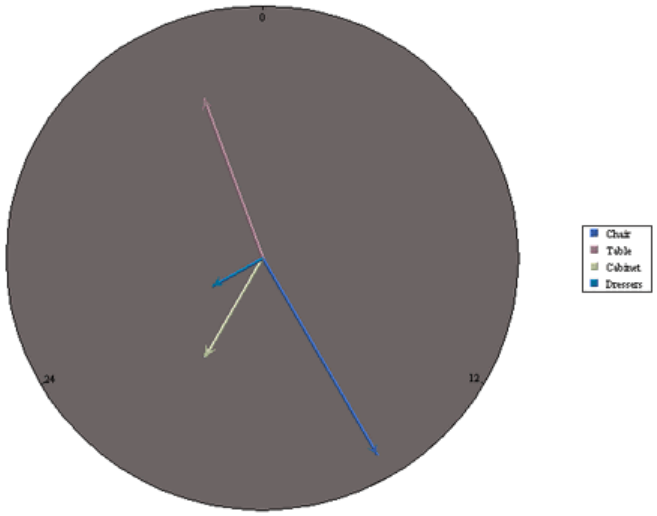
The following code demonstrates how to create the aforementioned Dial chart.

```
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/Docs72/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.

10.B.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.

10.B.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;
```

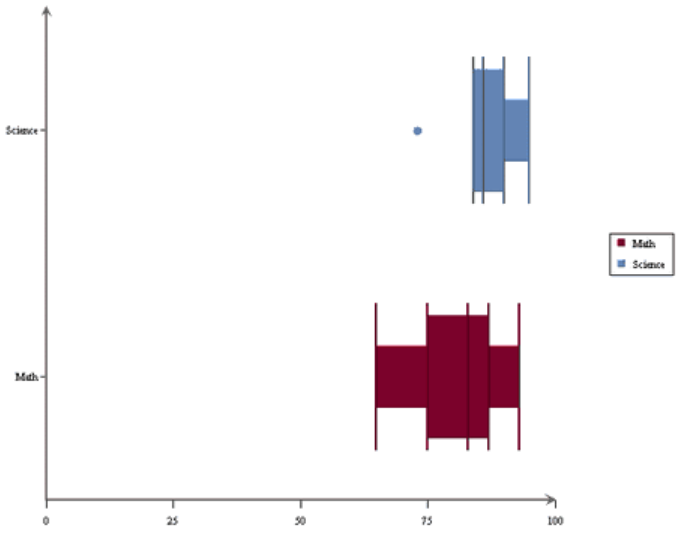
10.B.6.2. Creating the Chart

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/Docs72/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.

10.B.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.

10.B.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;
```

10.B.7.2. Creating the Chart

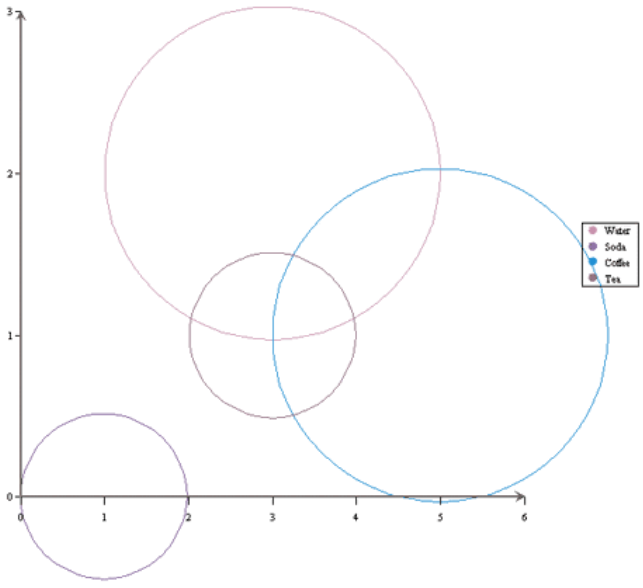
The following code demonstrates how to create the aforementioned Bubble chart.

```
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/Docs72/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.

10.B.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.

10.B.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;
```

10.B.8.2. Creating the Chart

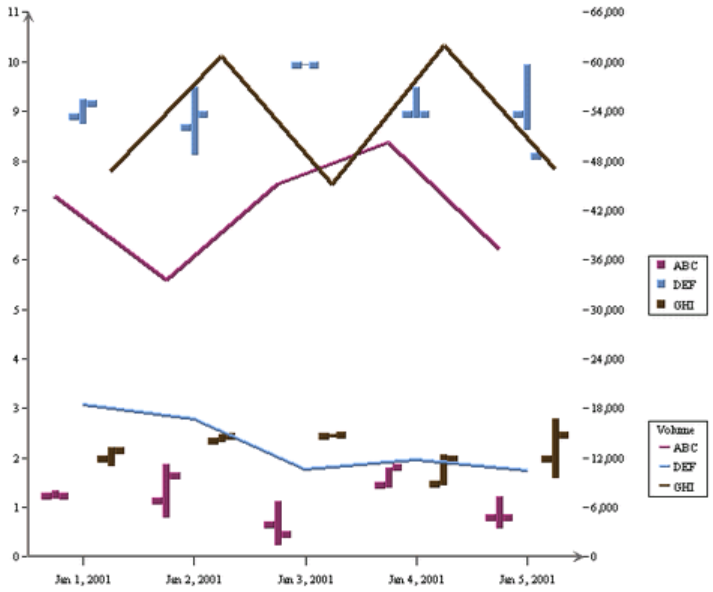
The following code demonstrates how to create the aforementioned HLCO chart.

```
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/Docs72/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.

10.B.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.

10.B.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);
```

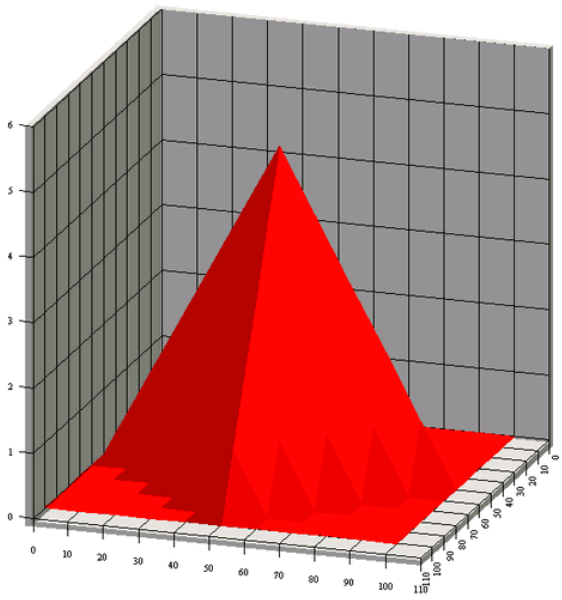
10.B.9.2. Creating the Chart

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/Docs72/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.

10.B.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.

10.B.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;
```

10.B.10.2. Creating the Chart

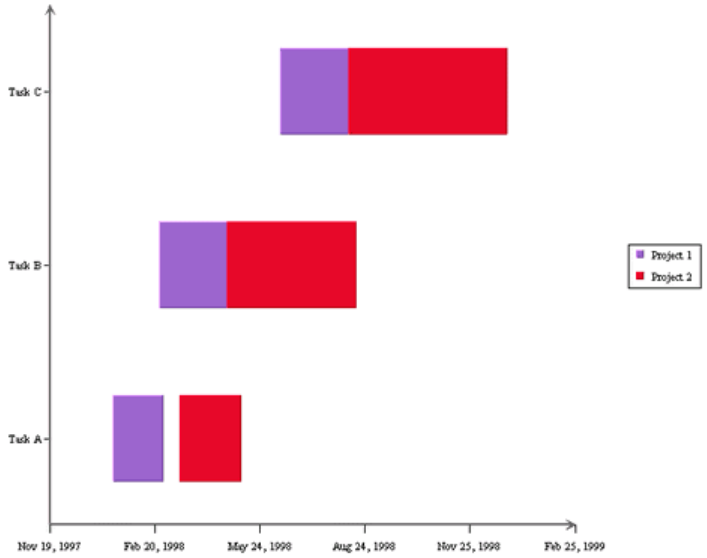
The following code demonstrates how to create the aforementioned Gantt chart.

```
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/Docs72/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.

10.B.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.

10.B.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});
```

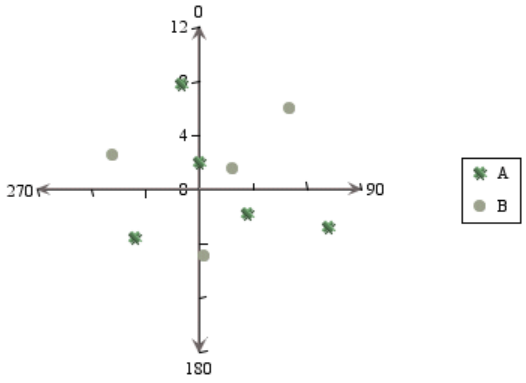
10.B.11.2. Creating the Chart

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/Docs72/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.

10.B.12. Date/Time Based Zoom Charts

EspressChart 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.

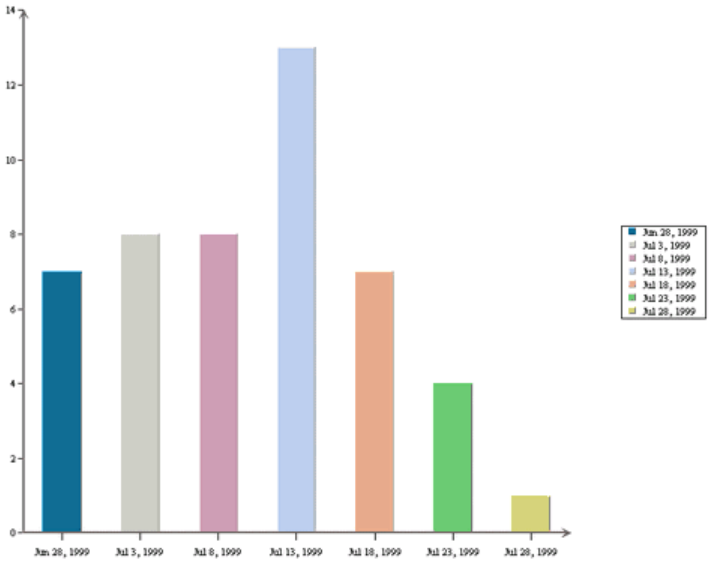
Zooming can be achieved by setting the attributes and then refreshing the chart. The attributes are set using the `quadbase.util.IZoomInfo` interface.

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/Docs72/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.

10.B.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.

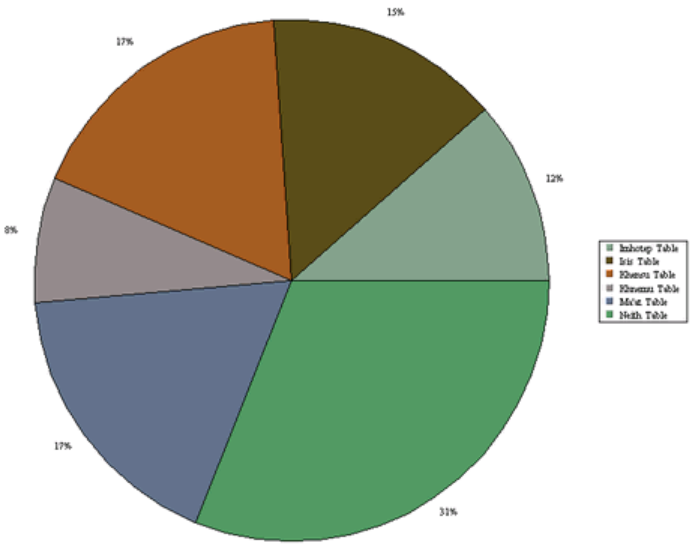
Given below is an example of a chart that uses a parameterized query. The database against which the query is run is WoodView HSQL, so you will need to add database HSQL JDBC driver (`hsqldb.jar`) to your classpath. The driver is located in the `<EspressChartInstall>/lib` directory.

```
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/Docs72/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.

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 EspressChart 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.

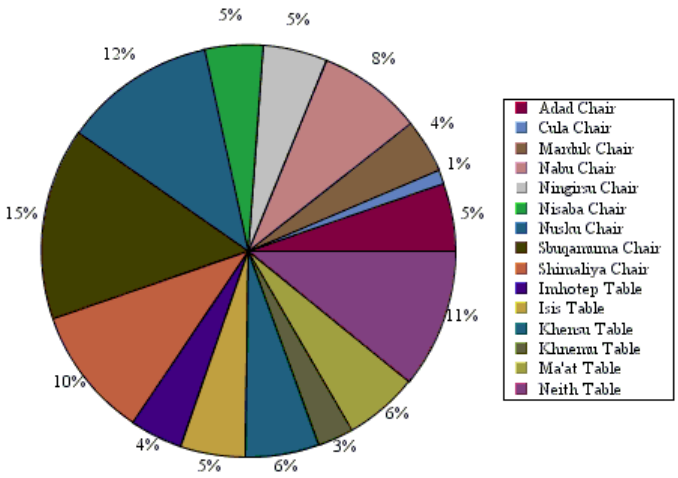
Given below is an example of a chart that uses a multi-value parameterized query. The database against which the query is run is WoodView HSQL, so you will need to add database HSQL JDBC driver (`hsqldb.jar`) to your classpath. The driver is located in the `<EspressChartInstall>/lib` directory.

```
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/Docs72/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.

10.B.14. Drill-Down Charts

EspressChart 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.

10.B.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 7.3 - 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.

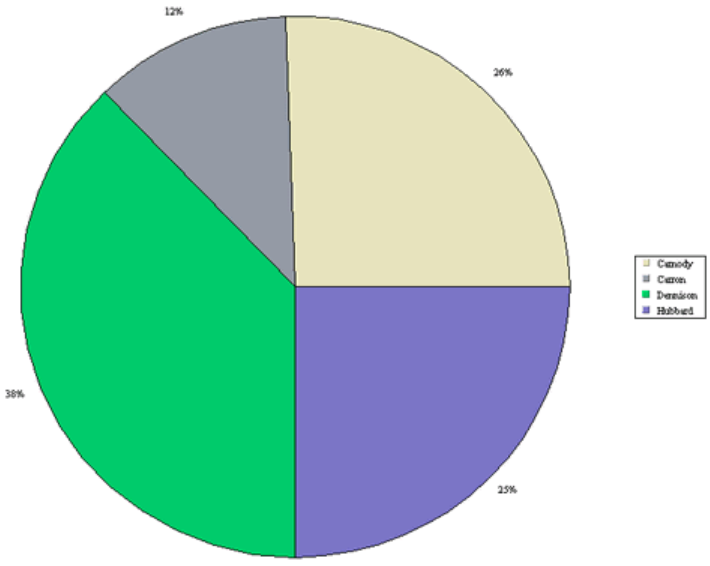
Given below is an example of a parameter drill-down chart. The database against which the query is run is Wood-View HSQL, so you will need to add database HSQL JDBC driver (`hsqldb.jar`) to your classpath. The driver is located in the `<EspressChartInstall>/lib` directory.

```
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;

}
```

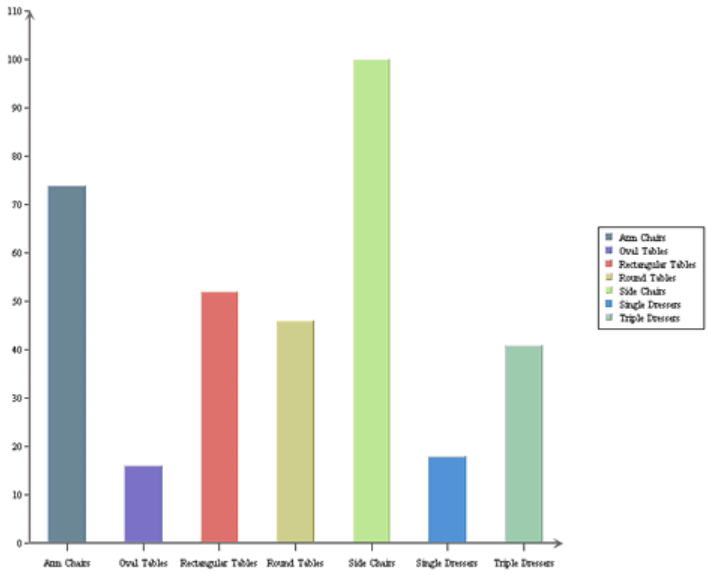
Full Source Code [https://data.quadbase.com/Docs72/help/manual/code/src/ParameterDrillDownChart.zip]

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.

10.B.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.

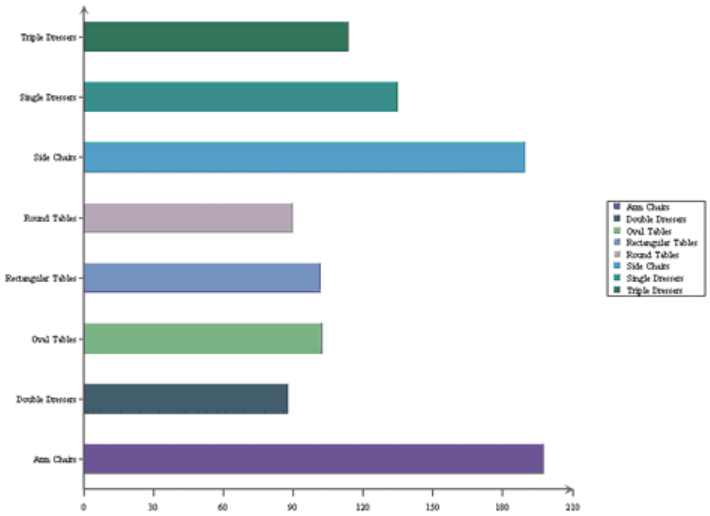
Given below is an example of a data drill-down chart. The database against which the query is run is WoodView HSQL, so you will need to add database HSQL JDBC driver (`hsqldb.jar`) to your classpath. The driver is located in the `<EspressChartInstall>/lib` directory.

```
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;

}
```

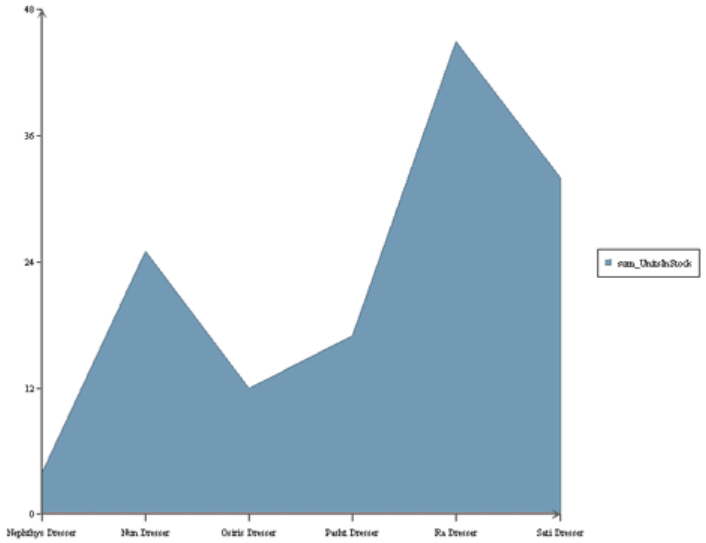
Full Source Code [https://data.quadbase.com/Docs72/help/manual/code/src/DataDrillDownChart.zip]

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.

10.B.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.

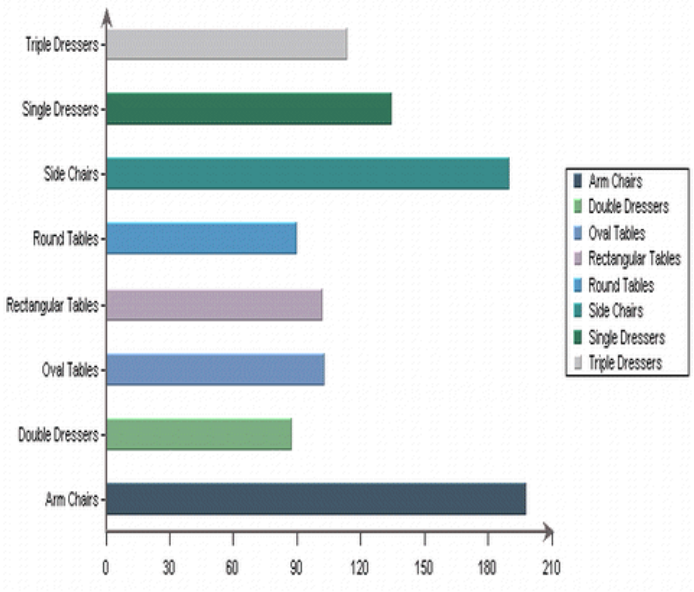
Given below is an example of a data drill-down chart. The database against which the query is run is WoodView HSQL, so you will need to add database HSQL JDBC driver (`hsqldb.jar`) to your classpath. The driver is located in the `<EspressChartInstall>/lib` directory.

```
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;
}
```

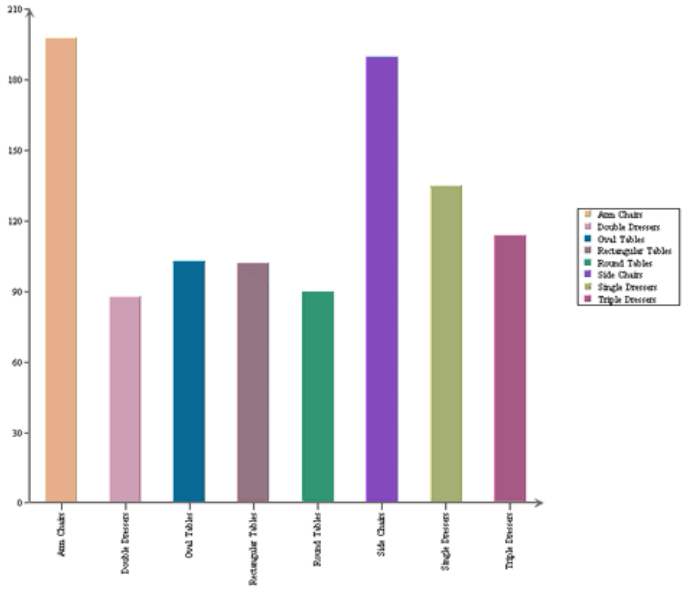
Full Source Code [https://data.quadbase.com/Docs72/help/manual/code/src/DynamicDataDrillDownChart.zip]

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.

Chapter 11. Servlets and JavaServer Pages

11.1. Servlets

11.1.1. Introduction

As described on the Oracle Web site (https://www.oracle.com/java/technologies/servlet-technology.html), "Servlets are the Java platform technology of choice for extending and enhancing Web servers. Servlets provide a component-based, platform-independent method for building Web-based applications, without the performance limitations of CGI programs. And unlike proprietary server extension mechanisms (such as the Netscape Server API or Apache modules), servlets are server- and platform-independent. This leaves you free to select a "best of breed" strategy for your servers, platforms, and tools."

The sections below describe how to set up the example servlet provided with EspressChart (located in `EspressChart/help/examples/servlet/DatabaseJPEG`). Please note that each section deals specifically with the example database servlet. They can however be used as a guide for setting up and running your own servlets as well as the other example servlets given.

In addition to the database servlet, other servlet examples have also been provided. They range from the simple showing of a chart to the streaming of the chart directly to the browser to showing drill-down charts as a static image. Setup and running of these examples can be done using the following sections as a guide.

11.1.2. Setup

1. Make sure that the EspressManager is running
2. Next, depending on what servlet server (runner) that you are using, follow the guideline below. Then run `ExportChart2.html` (located in the `EspressChart/help/examples/servlet/DataFile` directory) from a browser.

Note

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-thread model for simplicity's sake. However, EspressChart API can also be used in a multi-thread environment.

The following uses the DataFile servlet example under `EspressChart/help/examples/servlet`.

11.1.3. Running Under

11.1.3.1. Apache Tomcat 11

Checking Tomcat version

Please be sure you have Apache Tomcat 11 installed and configured to use Java 21.

There are two installation file versions for Windows operating system:

zip file uses `startup.bat` and `shutdown.bat` files for starting and stopping the server, that can be found after extraction of the archive in bin directory

exe file installs Windows service and uses “Apache Tomcat Monitor” application

EspressManager should be running

Please, run EspressManager by executing `EspressManager.bat` in `<EspressChart install dir>`.

Virtual directory mapping

In order to make example work correctly, you need to map EspressChart install directory as a virtual directory under Apache Tomcat's webroot.

Please open `server.xml` file (placed in `<Tomcat install dir>\conf\` directory) with an editor and add the following:

```
<Context path="/EspressChart"
 docBase="<EspressChart install dir path>"
 privileged="true">
    <Manager pathname=""/>
</Context>
```

between `<Host>` and `</Host>` xml tags and save the changes.

For example:

```
<Context path="/EspressChart" docBase="C:/
EspressChart" debug="0" privileged="true">
    <Manager pathname=""/>
</Context>
```

Note

You can also install EspressChart to `<Tomcat install dir>\webapps\ROOT` directory to avoid virtual directory mapping. However, this is not recommended.

Classpath setting

You should include `EspressAPI.jar` and `servlet-api.jar` files in the classpath.

`EspressAPI.jar` file can be found in `EspressChart\lib` directory, `servlet-api.jar` in `<Tomcat install dir>\lib\` directory.

Here is the example classpath setting (using command line console):

```
set classpath=%classpath%;<Tomcat install dir>\lib
\servlet-api.jar; <EspressChart install dir>\lib
\EspressAPI.jar
```

Note

Classpath does not necessarily need to be set, but it simplifies `.java` files compilation (you do not need to add classpath there).

Tip

You can set Classpath variable also using MyComputer -> Administrate -> Details tab window.

Compiling the example file (`ExportServlet2.java`)

Compile command:

(if classpath environment variable was not set in step 4 of this guide)

```
javac -classpath "<Tomcat install dir>\lib
\servlet-api.jar; <EspressChart install dir>\lib
\EspressAPI.jar" ExportServlet2.java
```

or simply:

(if classpath was set properly)

```
javac ExportServlet2.java
```

Copying the class file to Tomcat's deploying directory

Please copy `ExportServlet2.class` file to `<Tomcat install dir>\webapps\ROOT\WEB-INF\classes` directory.

Note

Please note that `<Tomcat install dir>\webapps\ROOT\WEB-INF\classes` directory does not need to exist by default and you will have to create it.

Adding necessary libraries for running the servlet to lib dir

In order to run the example, you also need to add necessary libraries to your web application's lib directory. Please copy `EspressAPI.jar` file placed in `EspressChart\lib` directory to `<Tomcat install dir>\webapps\ROOT\WEB-INF\lib` directory.

Note

Please note that `<Tomcat install dir>\webapps\ROOT\WEB-INF\lib` directory does not need to exist by default and you will have to create it.

Registering your servlet application in web.xml file

Before you can run your servlet application you need to modify your application's `web.xml` file. It is placed in `<Tomcat install dir>\webapps\ROOT\WEB-INF\` directory.
For registering ExportServlet2, please add the following code:

```
<servlet>
        <servlet-name>ExportServlet2</servlet-
name>
        <servlet-class>ExportServlet2</servlet-
class>
    </servlet>

    <servlet-mapping>
        <servlet-name>ExportServlet2</servlet-
name>
        <url-pattern>/servlet/ExportServlet2</
url-pattern>
    </servlet-mapping>
```

between `<web-app>` and `</web-app>` xml tags.

Note

You can also enable automatic invoking of all servlets that are placed in webapp directory. Then 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. For more information, consult documentation of your Tomcat server.

Starting Tomcat server

If you used zip installation file, please run `startup.bat` (`startup.sh` for Linux) file (placed in `<Tomcat install dir>\bin` directory).

Before starting Tomcat, make sure that `CATALINA_HOME` points to the Tomcat installation directory. If you want to use a particular Java for starting Tomcat, also set `JRE_HOME` to the Java 21 installation directory.

```
set "CATALINA_HOME=<Tomcat install dir>"
set "JRE_HOME=<Java 21 install dir>"
cd /d <EspressChart install dir>
<Tomcat install dir>\bin\startup.bat
```

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 start again) in order to run the example.

Running the example

The last step is running the example. To run the example, simply run `ExportChart2.html` file (placed in `EspressChart\help\examples\servlet\DataFile` directory) in your browser. You can set the path to your data file or you can use the default one that comes with EspressChart installation. By clicking on *Get Chart Image* button, the ExportServlet2 is called and the chart image is exported and then printed on the page.

For Tomcat, the file can be opened from the mapped EspressChart context, for example: **`http://localhost:8080/EspressChart/help/examples/servlet/DataFile/ExportChart2.html`**.

Note

Please be sure that EspressManager is running before you run the example.

Note

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 `ExportChart2.html` file

From:

```
<form action="http://localhost:8080/
servlet/ExportServlet2" method="post">
```

To:

```
<form action="http://<machine_name>:8080/
servlet/ExportServlet2" method="post">
```

11.1.3.2. JRun

- Replace the machine name and port number in the html file to

  ```
  <FORM ACTION=http://<machine name>:8100/servlet/ExportServlet method=POST>
  ```

  where `<machine name>` is the name of the machine running the JRun server.

- Log into the "JRun Default Server Administrator"
- Click on *Java Settings* and then *Classpath*
- Add the path to `EspressAPI.jar` and `ExportLib.jar` in separate lines 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`
- Open up modified `ExportChart2.html` in a browser, input the parameters, and then click on button to get the chart.

11.1.3.3. ColdFusion Server

- Rename `ExportChart2.html` to `ExportChart2.cfm` and replace the machine name and port number in the cfm file to

  ```
  <FORM ACTION=http://<machine name>:8100/servlet/ExportServlet method=POST>
  ```

  where, `<machine name>` is the name of the machine running the JRun server.

- Log into the JRun Default Server Administrator
- Click on *Java Settings* and then *Classpath*
- Add the path to `EspressAPI.jar` and `ExportLib.jar` in separate lines 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`
- Open up the modified `ExportChart2.cfm` in a browser, input the parameters, and click on the button to get chart.

11.1.3.4. WebLogic 6.0

For the NT platform, to run EspressChart under WebLogic 6.0, install it under the `bea\wlserver6.0\config\examples\applications\examplesWebApp` directory. To access it, type in **`http://<web address or machine name>:7001/examplesWebApp/ EspressChart/index.html`**. Remember to start EspressManager first as you will be accessing the Chart Designer.

To set up and run the servlet example, first, you must go to the `wlserver6.0\config\examples` directory. Modify both `setExamplesEnv.cmd` and `startExamplesServer.cmd` files according to the following instructions:

- Add **`set ES_CHART=YOUR_WEBROOT\espresschart`** in the *@rem Set user-defined variables* section. If you installed WebLogic 6.0 under `c:\bea`, `YOUR_WEBROOT` is equal to `c:\bea\wlserver6.0\config\examples\applications`. The `ES_CHART` variable contains the path to the EspressChart 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 **`c:\bea;.; %ES_CHART%\lib\EspressAPI.jar; %ES_CHART%\lib\ExportLib.jar;`** to the *set CLASSPATH* field of the same file.

Next, follow the steps below (note that the files are under `espresschart\help\examples\servlet\Weblogic` directory):

- Put the `ExportChart2.html` in the `wlserver6.0\config\examples\applications\examplesWebApp` directory.
- Change the machine name in the code **`archon:7001`** to **`yourMachineName:7001`** in the `ExportChart2.html` file.
- Put the `ExportServlet2.java` file 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, under the `examplesWebApp` 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>ExportServlet2</servlet-name>
    <servlet-class>ExportServlet2</servlet-class>

        <init-param>

            <param-name>dataFileName</param-name>
            <param-value>help/examples/data/sample.dat</param-value>

        </init-param>
        <init-param>

            <param-name>category</param-name>
            <param-value>0</param-value>

        </init-param>
        <init-param>

            <param-name>series</param-name>
            <param-value>-1</param-value>

        </init-param>
        <init-param>

            <param-name>sumby</param-name>
            <param-value>-1</param-value>

        </init-param>
        <init-param>

            <param-name>value</param-name>
            <param-value>3</param-value>

        </init-param>
        <init-param>
```

```
                <param-name>chartType</param-name>
                <param-value>Column</param-value>

            </init-param>
            <init-param>

                <param-name>fileName</param-name>
                <param-value>c:\temp</param-value>

            </init-param>

        </servlet>

    <servlet-mapping>

        <servlet-name>ExportServlet2</servlet-name>
        <url-pattern>/ExportServlet2/*</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 `ExportServlet2.java` using the command lines listed below.

```
javac –d %EX_WEBAPP_CLASSES% ExportServlet2.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/ExportChart2.html`** to view the servlet example.

Tip

For troubleshooting please check for typing errors.

11.1.3.4.1. 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 EspressChart installation will be referenced as `<EC_INSTALL_DIR>`.

In order to run EspressChart under WebLogic 9.2, install it under the `<WL_INSTALL_DIR>\samples\server\examples\build\examplesWebApp` directory. To access it, type in **`http://<web address or machine name>:7001/examplesWebApp/EspressChart/index.html`**. Remember to start EspressManager first as you will be accessing the Chart Designer.

To set up and run the servlet example, first, you must 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_CHART=<EC_INSTALL_DIR>`**. If you installed WebLogic 9.2 under `c:\bea`, <EC_INSTALL_DIR> , it is equal to `C:\bea\weblogic92\samples\server\examples\build\examplesWebApp\EspressChart`. The `ES_CHART` variable contains the path to the EspressChart 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_CHART%\lib\EspressAPI.jar; %ES_CHART%\lib\ExportLib.jar; %ES_CHART%\lib\qblicense.jar;** to the *set CLASSPATH* field of the same file.

Next, follow the steps below (note that the files are under <EC_INSTALL_DIR>\help\examples\servlet\Weblogic directory):

- Put the DialServletDemo.java & DialServletExp.java files in the <WL_INSTALL_DIR>\samples\server\examples\build\examplesWebApp\WEB-INF\classes directory.
- Change the machine name in the code fragment **archon:7001** to **yourMachineName:7001** in the DialServletDemo.java file.
- Change the constructor bellow in the DialServletExp.java file.
  From:

```
chart = new QbChart(new Frame(), QbChart.VIEW2D, QbChart.DIAL, data,
 colInfo, "config/examples/applications/examplesWebApp/WEB-INF/classes/
dial.tpl");
```

  To:

```
chart = new QbChart(new Frame(), QbChart.VIEW2D, QbChart.DIAL, data,
 colInfo, "samples/server/examples/examplesWebApp/WEB-INF/classes/
dial.tpl");.
```

- 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>DialServletExp</servlet-name>
        <servlet-class>DialServletExp</servlet-class>

    </servlet>

    <servlet>

        <servlet-name>DialServletDemo</servlet-name>
        <servlet-class>DialServletDemo</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>DialServletExp</servlet-name>
        <url-pattern>/DialServletExp/*</url-pattern>

    </servlet-mapping>

    <servlet-mapping>

        <servlet-name>DialServletDemo</servlet-name>
        <url-pattern>/DialServletDemo/*</url-pattern>

    </servlet-mapping>
```

- Copy the `DialServletDemo.html` file to the `<WL_INSTALL_DIR>\samples\server\examples\build\examplesWebApp` directory.
- Edit the `DialServletDemo.html` file and change all machine names from **`archon:7001/servlet`** to **`yourMachineName:7001/examplesWebApp`**.
- Copy the `dial.tpl` file to the `<WL_INSTALL_DIR>\samples\server\examples\build\examplesWebApp\WEB-INF\classes` directory.
- Then, go to `<WL_INSTALL_DIR>\samples\server\examples\build\examplesWebApp\WEB-INF\classes` directory and compile `DialServletExp.java` & `DialServletDemo.java`. Please include `EspressAPI.jar`, `ExportLib.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/DialServletDemo.html`** to view the servlet example.

Tip

For troubleshooting, please check for typing errors.

11.1.3.5. WebSphere 3.5

- Open the WebSphere Administrator's Console and under the node (usually the machine name) <MACHINE_NAME> → Default Servlet Engine → Default App, click on the *Advanced* tab and add `EspressAPI.jar` and `ExportLib.jar`(in separate lines) to the Classpath. Then click on *Apply*
- Move the compiled class file (`ExportServlet2.class`) to the `<WebSphere installation directory>/AppServer/servlets directory`
- Modify the `FORM ACTION` tag to the following:

```
<FORM ACTION=http://<machine name>/servlet/ExportServlet2 method=POST>
```

- Start/Restart the Default Server under the node

11.1.3.5.1. 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 WebLogic installation will be referenced as `<WS_INSTALL_DIR>` and the location of the EspressChart installation will be referenced as `<EC_INSTALL_DIR>`.

To set up and run the servlet example, first, copy `<EC_INSTALL_DIR>\lib\EspressAPI.jar`, `<EC_INSTALL_DIR>\lib\ExportLib.jar` and `<EC_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 `<EC_INSTALL_DIR>\help\examples\servlet\DataFile` directory):

- Put the `ExportServlet2.java` file in the `<WS_INSTALL_DIR>\AppServer\profiles\AppSrv01\installedApps\<YourMachineName>Node01Cell\ivtApp.ear\ivt_app.war\WEB-INF\classes` directory.
- Insert the following code fragments in the `web.xml` file located in the `<WS_INSTALL_DIR>\AppServer\profiles\AppSrv01\installedApps`

\<YourMachineNam>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>ExportServlet2</servlet-name>
    <servlet-class>ExportServlet2</servlet-class>

</servlet>

<servlet-mapping>

    <servlet-name>ExportServlet2</servlet-name>
    <url-pattern>ExportServlet2</url-pattern>

</servlet-mapping>
```

- Edit the ExportChart2.html file and change the machine name and port from **yourmachine:8080** to **yourMachineName:9080/ivt**.
- Copy the ExportChart2.html file to the <WS_INSTALL_DIR>\AppServer\profiles\AppSrv01\installedApps\<YourMachineName>Node01Cell \ivtApp.ear\ivt_app.war directory.
- Edit the ExportServlet2.java file and change code fragment from http://yourmachine:8080/EspressChart/ to **<EC_INSTALL_DIR>**.
- Then, go to <WS_INSTALL_DIR>\AppServer\profiles\AppSrv01\installedApps \<YourMachineName>Node01Cell \DefaultApplication.ear\DefaultWebApplication.war\WEB-INF\classes directory and compile ExportServlet2.java. Please include EspressAPI.jar, ExportLib.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 → IBM WebSphere → 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/ExportChart2.html** to view the servlet example.

Tip

For troubleshooting, please check for typing errors.

11.1.3.6. 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 EspressChart installation will be referenced as <EC_INSTALL_DIR>.

To set up and run the servlet example, first, go to the <JB_INSTALL_DIR>JB_INSTALL_DIR>\bin directory. Next,modify the run.bat file according to the following instructions:

- Add **set ES_CHART=<EC_INSTALL_DIR>**. The ES_CHART variable contains the path to the EspressChart home directory in your machine.

- Add **%ES_CHART%\lib\EspressAPI.jar; %ES_CHART%\lib\ExportLib.jar; %ES_CHART%\lib\qblicense.jar;** to the *set JBOSS_CLASSPATH* field of the same file. Do not type in the double quote.

Next, follow the steps below (note that the files are under <EC_INSTALL_DIR>\help\examples\servlet\DataFile directory):

- Put the ExportServlet2.java file in the <JB_INSTALL_DIR>\server\default\deploy\jmx-console.war\WEB-INF\classes directory.
- Edit the ExportServlet2.java file and change code fragment from **http://yourmachine:8080/EspressChart/** to **<EC_INSTALL_DIR>**.
- 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>ExportServlet2</servlet-name>
    <servlet-class>ExportServlet2</servlet-class>

</servlet>

<servlet-mapping>

    <servlet-name>ExportServlet2</servlet-name>
    <url-pattern>/ExportServlet2</url-pattern>

</servlet-mapping>
```

- Edit the ExportChart2.html file and change part of URL **http://yourmachine:8080/servlet**to **http://yourMachineName:8080/jmx-console**.
- Copy the ExportChart2.html file to the <JB_INSTALL_DIR>\server\default\deploy\jmx-console.war directory.
- Then, go to <JB_INSTALL_DIR>\server\default\deploy\jmx-console.war\WEB-INF\classes directory and compile ExportServlet2.java. Please include EspressAPI.jar, ExportLib.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/ExportChart2.html** to view the servlet example.

Tip

For troubleshooting, please check for typing errors.

11.1.3.7. 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 EspressChart installation will be referenced as <EC_INSTALL_DIR>.

Follow the steps below (note that the files are under `<EC_INSTALL_DIR>\help\examples\servlet\DataFile` directory):

- Put the `ExportServlet2.java` file in the `<OR_INSTALL_DIR>\default-web-app\WEB-INF\classes` directory.
- Edit the `ExportServlet2.java` file and change code fragment from **`http://yourmachine:8080/EspressChart/`** to **`<EC_INSTALL_DIR>`**.
- 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>ExportServlet2</servlet-name>
    <servlet-class>ExportServlet2</servlet-class>

</servlet>

<servlet-mapping>

    <servlet-name>ExportServlet2</servlet-name>
    <url-pattern>ExportServlet2</url-pattern>

</servlet-mapping>
```

- Edit the `ExportChart2.html` file and change part of URL **`http://yourmachine:8080/servlet`** to **`http://yourMachineName:80`**.
- Copy the `ExportChart2.html` file to the `<OR_INSTALL_DIR>\default-web-app` directory.
- Then, go to `<OR_INSTALL_DIR>default-web-app\WEB-INF\classes` directory and compile `ExportServlet2.java`. Please include `EspressAPI.jar`, `ExportLib.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).
- 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/ExportChart2.html`**" to view the servlet example.

Tip

For troubleshooting, please check for typing errors.

11.1.3.8. 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 EspressChart installation will be referenced as `<EC_INSTALL_DIR>`.

Follow the steps below (note that the files are under `<EC_INSTALL_DIR>\help\examples\servlet\DataFile` directory):

- Put the `ExportServlet2.java` file in the `<JR_INSTALL_DIR>\servers\default\default-ear\default-war\WEB-INF\classes` directory.

- Edit the `ExportServlet2.java` file and change code fragment from `http://yourmachine:8080/EspressChart/` to **`<EC_INSTALL_DIR>`**.
- 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>ExportServlet2</servlet-name>
    <servlet-class>ExportServlet2</servlet-class>

</servlet>

<servlet-mapping>

    <servlet-name>ExportServlet2</servlet-name>
    <url-pattern>ExportServlet2</url-pattern>

</servlet-mapping>
```

- Edit the `ExportChart2.html` file and change part of URL `http://yourmachine:8080/servlet` to **`http://yourMachineName:8100`**.
- Copy the `ExportChart2.html` file to the `<JR_INSTALL_DIR>\servers\default\default-ear\default-war` directory.
- Then, go to `<JR_INSTALL_DIR>\servers\default\default-ear\default-war\WEB-INF\classes` directory and compile `ExportServlet2.java`. Please include `EspressAPI.jar`, `ExportLib.jar` and `j2ee.jar` in the classpath.

Tip

You can use `j2ee.jar` from the J2eeSDK installation located under `<J2EE_SDK_INSTALL_DIR>\lib directory`

- Copy the `EspressAPI.jar`, `ExportLib.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/ExportChart2.html`** to view the servlet example.

Tip

For troubleshooting, please check for typing errors.

11.1.3.9. 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 EspressChart installation will be referenced as `<EC_INSTALL_DIR>`.

Follow the steps below (note that the files are under `<EC_INSTALL_DIR>\help\examples\servlet\StreamingChart` directory):

- Put the `StreamChartServlet.java` file in the `<ORA_INSTALL_DIR>\j2ee\home\default-web-app\WEB-INF\classes` directory.
- Edit the `StreamChartServlet.java` file and change code fragment from `http://yourmachine:8080/EspressChart` to **`<EC_INSTALL_DIR>`**.
- 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>StreamChartServlet</servlet-name>
    <servlet-class>StreamChartServlet</servlet-class>

</servlet>

<servlet-mapping>

    <servlet-name>StreamChartServlet</servlet-name>
    <url-pattern>/servlet/StreamChartServlet</url-pattern>

</servlet-mapping>
```

- Edit the `StreamChartServlet.html` file and change part of URL `http://yourmachine:8080/servlet` to **`http://yourMachineName:8888/servlet`**.
- Copy the `StreamChartServlet.html` file to the `<ORA_INSTALL_DIR>\j2ee\home\default-web-app` directory.
- Then, go to `<ORA_INSTALL_DIR>\j2ee\home\default-web-app\WEB-INF\classes` directory and compile `StreamChartServlet.java`. Please include `EspressAPI.jar`, `ExportLib.jar` and `servlet.jar` in the classpath.

Tip

The `servlet.jar` file is located under `<ORA_INSTALL_DIR>\j2ee\home\lib` directory

- Copy the `EspressAPI.jar`, `ExportLib.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 → Programs → Oracle → Start Soa suite. 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/StreamChartServlet.html`** to view the servlet example.

Tip

For troubleshooting, please check for typing errors.

11.1.3.10. 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 EspressChart installation will be referenced as <EC_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.

Tip

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 <EC_INSTALL_DIR>\help\examples\servlet\DataFile directory):

- Put the ExportServlet2.java file in the <USER_DIR>\WEB-INF\classes directory.
- Edit the ExportServlet2.java file and change code fragment from http://yourmachine:8080/EspressChart/ to **<EC_INSTALL_DIR>**.
- Insert the following code fragments in the web.xmlfile located in the <USER_DIR>\WEB-INF directory.

```
<servlet>

    <servlet-name>ExportServlet2</servlet-name>
    <servlet-class>ExportServlet2</servlet-class>

</servlet>

<servlet-mapping>

    <servlet-name>ExportServlet2</servlet-name>
    <url-pattern>/servlet/ExportServlet2</url-pattern>

</servlet-mapping>
```

- Edit the ExportChart2.html file and change the part of URL http://yourmachine:8080/servletto **http://yourMachineName:8080/hello/servlet**.
- Copy the ExportChart2.html file to the <USER_DIR> directory.
- Then, go to <USER_DIR>\WEB-INF\classes directory and compile ExportServlet2.java. Please include EspressAPI.jar, ExportLib.jar and j2ee.jar in the classpath.

Tip

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 EspressAPI.jar, ExportLib.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 → Programs → Sun Microsystems → Application Server PE 9 → Start Default Server.
- Open your web browser and go to **http://yourMachineName:8080/hello/ExportChart2.html** to view the servlet example.

Tip

For troubleshooting please check for typing errors.

11.1.3.11. 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 EspressChart installation will be referenced as <EC_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.

Tip

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 webapps-simple.war** in the <USER_DIR> directory.

Follow the steps below (note that the files are under <EC_INSTALL_DIR>\help\examples\servlet\DataFile directory):

- Put the ExportServlet2.java file in the <USER_DIR>\WEB-INF\classes directory.
- Edit the ExportServlet2.java file and change code fragment from http://yourmachine:8080/EspressChart/ to **<EC_INSTALL_DIR>**.
- 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>ExportServlet2</servlet-name>
    <servlet-class>ExportServlet2</servlet-class>

</servlet>

<servlet-mapping>

    <servlet-name>ExportServlet2</servlet-name>
    <url-pattern>/servlet/ExportServlet2</url-pattern>

</servlet-mapping>
```

- Edit the ExportChart2.html file and change the part of URL http://yourmachine:8080/servlet to **http://yourMachineName:81/webapps-simple/servlet**.

- Copy the `ExportChart2.html` file to the `<USER_DIR>` directory.
- Then, go to `<USER_DIR>\WEB-INF\classes` directory and compile `ExportServlet2.java`. Please include `EspressAPI.jar`, `ExportLib.jar` and `j2ee.jar` in the classpath.

Tip

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 then create a war file by executing the following command: **`jar -cvf webapps-simple.war *`**.
- Copy the `EspressAPI.jar`, `ExportLib.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. Next 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/ExportChart2.html`** to view the servlet example.

Tip

For troubleshooting please check for typing errors.

11.1.3.12. Resing 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 EspressChart installation will be referenced as `<EC_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.

Tip

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 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):

- Put the ExportServlet2.java file in the <RES_INSTALL_DIR>\webapps\test\WEB-INF\classes directory.
- Edit the ExportServlet2.java file and change code fragment from http://yourmachine:8080/EspressChart/to **<EC_INSTALL_DIR>**.
- Insert the following code fragments in the resin-web.xmlfile located in the <RES_INSTALL_DIR>\webapps\test\WEB-INF directory.

```
<servlet-name="ExportServlet2" servlet-class="ExportServlet2"/>

<servlet-mapping url-pattern="/servlet/ExportServlet2" servlet-
name="ExportServlet2"/>
```

- Edit the ExportChart2.html file and change the part of URL http://yourmachine:8080/servlet to **http://yourMachineName:8080/test/servlet**.
- Copy the ExportChart2.html file to the <RES_INSTALL_DIR>\webapps\test directory.
- Then, go to <RES_INSTALL_DIR>\webapps\test\WEB-INF\classes directory and compile ExportServlet2.java. Please include EspressAPI.jar, ExportLib.jar and j2ee.jar in the classpath.

Tip

You can use j2ee.jar from the J2eeSDK installation located under <J2EE_SDK_INSTALL_DIR>\lib directory).

- Copy the EspressAPI.jar, ExportLib.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/ExportChart2.html** to view the servlet example.

Tip

For troubleshooting, please check for typing errors.

11.1.3.13. 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 installation will be referenced as <CF_INSTALL_DIR> and the location of the EspressChart installation will be referenced as <EC_INSTALL_DIR>.

Follow the steps below (note that the files are under <EC_INSTALL_DIR\help\examples\servlet\DataFile directory):

- Put the ExportServlet2.java file in the <CF_INSTALL_DIR>\wwwroot\WEB-INF\classes directory.
- Edit the ExportServlet2.java file and change code fragment from **http://yourmachine:8080/EspressChart/** to **<EC_INSTALL_DIR>**.

- Insert the following code fragments in the `web.xml`" file located in the `<CF_INSTALL_DIR>\www-root\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>ExportServlet2</servlet-name>
    <servlet-class>ExportServlet2</servlet-class>

</servlet>

<servlet-mapping>

    <servlet-name>ExportServlet2</servlet-name>
    <url-pattern>/servlet/ExportServlet2</url-pattern>

</servlet-mapping>
```

- Edit the `ExportChart2.html` file and change the part of URL `http://yourmachine:8080` to **`http://yourMachineName:8500`**.
- Copy the `ExportChart2.html` file to the `<CF_INSTALL_DIR>\wwwroot` directory.
- Then, go to `<CF_INSTALL_DIR>\wwwroot\WEB-INF\classes` directory and compile `Export-Servlet2.java`. Please include `EspressAPI.jar`, `ExportLib.jar` and `j2ee.jar` in the classpath

Tip

You can use `j2ee.jar` from the J2eeSDK installation located under `<J2EE_SDK_INSTAL-L_DIR>\lib directory`).

- Copy the `EspressAPI.jar`, `ExportLib.jar` and `qblicense.jar` files to the `<CF_INSTAL-L_DIR>\runtime\lib` directory.
- Start the ColdFusion application server. For Windows platforms, the ColdFusion application server is installed to run as a service; therefore, 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/ExportChart2.html`** to view the servlet example.

Tip

For troubleshooting, please check for typing errors.

11.1.4. Running the Servlet

After setting up the servlet per the instructions given above, open `ExportChart2.html` (located in the `EspressChart/help/examples/servlet/DataFile` directory) from a browser. Select from the parameters given and then click on *Get Chart Image*.

Note

You may have to clear the memory cache and disk cache every time in order get the latest result.

11.2. JavaServer Pages (JSP)

11.2.1. Introduction

JavaServer Pages allow 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 look like HTML files in many ways. However, whenever a JSP is run, the page gets converted to a normal servlet while the static HTML is printed.

JavaServer Pages is basically an extension of the Servlet technology. However, there is one very big advantage in using JSPs. JavaServer Pages technology separates the user interface from content generation enabling designers to change the overall page layout without altering the underlying dynamic content.

EspressChart provides a few JSP examples. Here, we will show you how to run the JSP example with JSWDK. The example we will be looking at has been modified from the datafile servlet example and is located under `Espress-Chart/help/examples/jsp/Chart`. This example can also be used as a guide to run the other JSP examples as well as your own JSP code.

11.2.2. Running Under

11.2.2.1. Apache Tomcat

- Please include `EspressAPI.jar`, `ExportLib.jar` and `servlet.jar` in the classpath. For instance,

```
set classpath=c:\netscape\suitespot\docs\espresschart\lib\EspressAPI.jar;
 c:\netscape\suitespot\docs\espresschart\lib\ExportLib.jar;
c:\tomcat\common\lib\servlet.jar;.
```

- Create a chart subdirectory under `webapps\examples\jsp\`. Therefore, if Tomcat is installed under `c:\`, you now have a new directory `c:\tomcat\webapps\examples\jsp\chart`
- Copy `ExportChart.jsp`, `ExportChartResponse.jsp` and `error.jsp` to the `examples/jsp/chart` directory
- Create another directory, also called **`chart`** under the `tomcat\webapps` directory. Thus, you now have `c:\tomcat\webapps\chart`
- Place the file `CreateChart.java` in the `tomcat\webapps\chart` directory. Next, modify the URL in the `getFileLocation()` by replacing `http://yourMachineName/EspressChart` to the correct URL. Suppose that your machine name is *Mach1* and you have installed EspressChart under the `Web` server directory. Then you specify **`http://Mach1/EspressChart`** as the URL for `getFileLocation()`
- Compile `CreateChart.java`. `CreateChart.class` should now exist in the same directory
- Start the Tomcat server and start the EspressManager
- Open a web browser and go to **`http://yourMachineName:8080/examples/jsp/chart/ExportChart.jsp`** (just substitute your actual machine name for *yourMachineName*)
- Specify the options that you want and then click submit to get a chart in return.

11.2.2.2. WebSphere

- Add `EspressAPI` and `ExportLib.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`** as the name of the web application. Make sure to uncheck the *Enable File Servlet* option, check *Server Servlet by classname*, and click on *Next*
- Choose *Default Servlet Engine* as the node (keep expanding the tree till can you select the *Default Servlet Engine*) and click on *Next*
- Change the *Web Application Web Path* to **`/Quadbase`** and click on *Next*
- Add `EspressAPI.jar` and `ExportLib.jar` in separate lines to the *Classpath* and click on *Finish*

- Restart the Default Server
- Create a directory under your web server's document root called **`Quadbase`** and move the `.jsp` files and the `java` file there.
- Modify CreateChart.java to
  - remove (or comment out) the package and
  - to replace the export command with

```
chart.export(QbChart.JPEG, "<Absolute path to location of web server
 document root>/EspressChart/temp", 500, 400);
```

- Modify the `.jsp` files to replace `chart.CreateChart` with **`CreateChart`**
- Create a directory under your web server's document root called **`EspressChart`**
- Start your browser and open **`http://<machine name>/Quadbase/ExportChart.jsp`** . Input the parameters and click on the button to get the chart

11.3. Saving a Chart to File versus Sending it to a Browser

You can use both servlets and JSPs to return a chart back to the browser. The chart can either be shown in an applet (i.e., it can be interactive) or be shown as a static image (i.e., a jpeg, a png etc.). Net traffic and bandwidth availability should be the key factors in deciding whether to use applets or static images.

The charts 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 chart is saved on the server side and the file location is sent to the client browser which then shows the chart. In the second scenario, the chart itself is sent, either as a string or as an output stream, to the client browser.

11.3.1. Saving the Chart to a File

Servlets and JSPs can save charts to files if the charts shown need to be re-used or a copy is needed. Here, charts are saved to a file on the server side (whether the chart is shown in a JNLP (More about Applets in JNLP: Section 2.6 - Run Applets in WebStart with JNLP file) or as a static image 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 chart. The browsers might also sometimes cache a previous chart (in the case of a static image) and they will not show the new chart unless refreshed/reloaded. Since the files are saved on the server side, one must be careful not to override the files before the client browser has seen the chart. They must be purged periodically in order to avoid taking up server space.

In both cases, a chart is exported to the desired format.

```
QbChart chart = new QbChart (...........);
....
```

```
....
....
int format = QbChart.CHT;    // here set to an interactive chart.  For a
 static chart change to QbChart.JPEG
chart.export(format, "chart", 500, 500);
```

An HTML file can then be returned to the browser, which redirects the page to a JSP file, which contains the chart location within the JSP file so that the browser can view it.

```
<?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"/>
   <jar href="lib/jsqlparser.jar"/>
   </resources>
   <security>
   <all-permissions/>
   </security>
   <applet-desc
      name="Chart Viewer"
      main-class="quadbase.chartviewer.Viewer"
      width="640"
      height="480">
 <param name="filename" value="help/examples/ChartAPI/data/test.tpl"/>
 <param name="preventSelfDestruct" value="true"/>
  </applet-desc>
   <update check="always" policy="always"/>
</jnlp>
```

Chart Viewer has been used here. Please note that in the above case, EspressManager MUST be running.

In the case of a static image, a simple `<img src>` tag giving the location of the static image is sufficient to display the chart.

Using this method, you can separate the pages returning the chart from your servlet/JSP and design them well in advance and outside of your servlet/JSP. You can use a database or a hashtable to keep track of the charts generated and show them again as needed, as well as have the charts re-generated again periodically.

For examples, please refer to the files in the `help/examples/servlets/DatabaseJPEG` directory and `help/examples/jsp/Chart` directories.

11.3.2. Sending the Chart Directly to a Browser

Servlets and JSPs can send charts directly to the browser so that they need not 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 chart cannot be made (unless it is printed from the browser). Therefore, the chart has to be re-generated if it is needed again.

Sending charts to the browser differs from saving them to files, as here, charts are sent either as an outputstream (in the case of static images) or as string (in the case of an interactive chart).

For a static image, on the servlet side, an outputstream is created and the chart 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("image/html");
            // where response is the response to the servlet
            OutputStream toClient = res.getOutputStream();
            QbChart chart = new QbChart (...........);
            ....
            ....
            ....
            chart.export(QbChart.JPEG, toClient, 500, 500);

        }

    }
```

On the HTML part, an `<img src>` tag with the servlet enclosed is sufficient to view the chart image.

```
<HTML><BODY>

 <img src="http://machine_name:8080/servlet/OutputStreamServlet>

</BODY></HTML>
```

You can also output the map file as an outputstream and embed it within your generated HTML file. Generating the map file as an outputstream can be done using the following method in the `QbChart` class:

```
export(int format, OutputStream image, OutputStream mapFile, String
 fileName, int w, int h, boolean generateMapFile, int option);
```

For an interactive chart, the chart is exported as a string.

```
QbChart chart = new QbChart (...........);
String buffer = chart.exportChartToString();
....
....
....
```

On the HTML side, you can use Chart Viewer to view the interactive chart. Here, the HTML is embedded in the servlet code itself for applet(More in: Section 2.6 - Run Applets in WebStart with JNLP file) as shown :-

```
toClient.println("<applet-desc main-class=\"quadbase.chartviewer.Viewer\"
 width=600 height=600 >");
toClient.println("<PARAM name=\"ChartData\" value=\"" +
 chart.exportChartToString() +                "\">");
toClient.println("</applet-desc>");
toClient.println("<PARAM name=\"ChartData\" value=\"" +
 chart.exportChartToString() +
                  "\">");
toClient.println("</applet>");
```

For examples, please refer to the files in the `help/examples/servlets/StreamingJPEG` and `help/examples/servlets/StreamingChart` directories.

On a JSP, you have to create a Java Bean object (`myStreamChart`), which creates a chart and returns it as a String object. The JSP page has to be modified as shown :-

```
<jsp:useBean id="myStreamChart" scope="page"
 class="streamingChart.CreateChart" />
<jsp:setProperty name="myStreamChart" property="*"/>
<applet-desc main-class="quadbase.chartviewer.Viewer" width=600 height=600 >
<PARAM name="ChartData" value="<%= myStreamChart.export()%>">
</applet-desc>
```

where `myStreamChart` java bean will contain the following code :-

```
    public string export() throws Exception {

        QbChart chart = new QbChart(....);

        return chart.exportChartToString();

    }
```

For an example, please refer to the files in `help/examples/jsp/streamingChart`.

In order to send a static image directly to the browser, a combination of JSP and servlet is used.

Chapter 12. Deployment

12.1. Introduction

EspressChart consists of several components: Chart Designer, Chart Viewer, EspressManager, and Chart API. Chart Designer is used to create charts in a GUI environment. Chart Viewer is a JNLP (More about Applets in JNLP: Section 2.6 - Run Applets in WebStart with JNLP file) that is used to view a chart (saved in .cht, .pac or.tpl format). Chart API is used to create charts programmatically. Scheduler is used to schedule exports for charts. Finally, EspressManager serves as a user administrator and handles data and data buffering.

Note that `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 Chart Designer **must** be used in conjunction with EspressManager, an option is provided for Chart Viewer and Chart API to work without connecting to EspressManager.

Connecting to EspressManager requires you to 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:

```
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).

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:

```
Qbchart.useServlet(true);
Qbchart.setServletRunner("http://someMachine:somePortNumber");
Qbchart.setServletContext("EspressChart/servlet");
```

will connect to EspressManager running at `http://someMachine:somePortNumber/Espress-Chart/servlet`.

Please note that these methods exist in the `QbChart` and `QbchartDesigner`.

For Chart Viewer a JNLP (More about Applets in JNLP: Section 2.6 - Run Applets in WebStart with JNLP file), 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.

12.2. Deploying with EspressManager

This section deals with deploying chart components that work in conjunction with EspressManager. To learn more about the interaction, please refer to the Section 10.3 - Interaction with EspressManager.

Note that any references to a data source or to a chart file, using the relative URL reference (i.e. `help\examples\data\sample.dat`) are relative to the directory from where EspressManager is running in.

12.2.1. Chart Designer

As mentioned before, Chart Designer must be used in conjunction with EspressManager. Chart Designer can be called using the API. Note that in this scenario, you will have to add `ChartDesigner.jar` in your CLASSPATH and place a copy of the images (`EspressChart/images`), and the backgroundImages (`EspressChart/backgroundImages`) directories under the working directory of the `.class` file.

Instead of copying the directories relative to the working directory when invoking the Chart Designer from API, an option to set the location `images/` and `backgroundImages/` directory is also available. Simply use a `QbchartDesigner` constructor with the parameter `imagesPath`. For more details about the `QbchartDesigner` constructor or the `imagesPath` parameter, please refer to the Chapter 10 - EspressChart Chart API.

To connect to EspressManager running as an application, you use the following two API methods in QbchartDesigner 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 `QbchartDesigner` 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 `QbChartDesigner` constructors.

12.2.2. Chart Viewer

Deploying Chart Viewer is similar to deploying Chart Viewer. Chart Viewer can be used in Java Web Start Application to show a chart saved in a `.cht` or `.tpl` file (either generated by Designer or API).

EspressViewer.jar must be included in the HTML file as the archive in order to deploy Chart Viewer in a Java Web Start Application.

To use Chart Viewer, construct an HTML page with the proper applet code (for more details, please refer to Chapter 9 - Chart Viewer). 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 Chart Viewer:

```
server_address (server address), server_port_number (port number)
```

To connect to EspressManager running as a servlet, you pass the following parameters to the Chart Viewer:

```
comm_protocol (servlet), comm_url (machine name/IP address and port number),
 servlet_context (servlet context)
```

12.2.3. Chart API

Chart API can be used in Java Web Start Application. Chart API can also be used in a servlet or jsp environment to generate server-side charts.

To deploy Chart API, `EspressAPI.jar` and `ExportLib.jar` must be in the CLASSPATH (for application and servlet/jsp environment) or included in the HTML file.

To connect to EspressManager running as an application, you would use the following methods in `QbChart` 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 `QbChart` 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 `QbChart` constructors.

12.3. Deploying without EspressManager

This section deals with deploying components that work independent of EspressChart. To learn more about the interaction, please refer to the Interaction with EspressManager section under the EspressChart API Overview chapter.

Generating charts independently of EspressManager results in a slightly better performance as the interaction of EspressManager and the chart or chart component is removed.

An important consideration is that any references to a data source or to a chart source, that is relative (i.e. for example `help\examples\data\sample.dat`) are relative to the working directory from where the JSP, JNLP file 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 chart templates can be moved to locations relative to the working directory, as dictated by the code.

12.3.1. 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 independently 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 chart 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:\EspressChart\TestApplet`, then for the chart to be displayed successfully, `help\examples\data\sample.dat` must exist within `D:\EspressChart\TestApplet` (i.e., `D:\EspressChart\TestApplet\help\examples\data\sample.dat` must exist).

12.3.2. Chart API

To deploy Chart API, `EspressAPI.jar` and `ExportLib.jar` must be in the CLASSPATH (for application and servlet/jsp environment) or included in the JSP or JNLP file.

Chart API can also be used without connecting to EspressManager by adding the line of code below:

```
QbChart.setEspressManagerUsed(false);
```

This method **must** be called prior to any `QbChart` instantiation.

As with Chart Viewer, any relative references to the chart 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:\EspressChart\TestApplication`, then for the chart to be displayed successfully, `help\examples\data\sample.dat` must exist within `D:\EspressChart\TestApplication` (i.e., `D:\EspressChart\TestApplicationhelp\examples\data\sample.dat` must exist).

12.4. Deploying in a Non-Windows Enviroment

EspressChart is a Pure Java tool for generating charts. As such, it does not contain graphical libraries for generating colors,fonts, and other AWT information. For that, Java relies on the system's (on which the charts 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.

This is usually not the case for non-Windows environments (such as Unix and Linux). 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 the 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.

12.4.1. 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).

12.4.2. JVM 1.4+ in Headless Mode

You can use a 1.5+ JVM run-time parameter to run applications using EspressChart. 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.

12.5. Platform Specific Issues

12.5.1. AS/400

12.5.1.1. Running under NAWT

To run any application using EspressChart 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
```

12.5.1.2. Running under Headless Mode

To run any application using EspressChart 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);
```

12.5.2. Linux/Unix

12.5.2.1. Running under X

To run any application using EspressChart 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.

12.5.2.2. Running under Headless Mode

To run any application using EspressChart 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);
```

Chapter 13. Internationalization

13.1. Internationalizing EspressChart

EspressChart provides many different features that allows you to generate charts for just about any locale and language. Different internationalization features can require different system or setting configurations depending on your specific requirements.

13.1.1. Specifying Locales

EspressChart allows different time zones and locales for charts. They are not limited to the locale on the machine on which they are created. The locale can be set through API right before the chart is run.

13.1.1.1. Locale-Specific Formatting

EspressChart allows you to set locale-specific formatting for chart elements, including date and numeric data. Locale-specific formatting allows data elements to be displayed in the correct format for the particular locale that is being used for the chart. Locale-specific options are available in the data formatting dialog for Chart Designer (see Section 6.7.2.1 - Axis Label Formatting).

The locale for charts, as well as the time zone, can be set at run-time through the API. Note that this only affects the date, time, and data formatting.

13.1.1.2. Setting Time Zones and Locales Using the API

To set the time zone or locale, you can use the methods in QbChart (applies to the entire chart) or use LocaleDateTimeFormat and LocaleNumericFormat objects (applies to a specific object in the chart). The following code fragments shows how to do this with two different approaches.

```
chart.setLocale(Locale.UK);
chart.settime zone(time zone.gettime zone("GMT"));
```

Full Source Code [https://data.quadbase.com/Docs72/help/manual/code/src/LocaleChart.zip]

Exported Results [https://data.quadbase.com/Docs72/help/manual/code/export/LocaleChart.png]

Here the formats are applied to the entire chart. 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 "Sales" column
chart.gethDataPoints().setLabelFormat(columnIndex, currencyFormat);
```

Full Source Code [https://data.quadbase.com/Docs72/help/manual/code/src/LocaleObject.zip]

Exported Results [https://data.quadbase.com/Docs72/help/manual/code/export/LocaleObject.png]

13.1.2. Language and Encoding

EspressChart 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 EspressChart 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 charts in that language.

13.1.2.1. EspressChart Language Translation

Internationalization is supported through the use of an `.xml` file, `Language.xml` (located in the `<EspressChart 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="CHARTDESIGNER" dir="quadbase/chartdesigner/designer">

      <File name="ChartMenubar.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. EspressChart will replace `<Code>` with `<Text>` in the Java Swing UI and/or in the EspressChart 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 EspressChart root directory and enter the following command:

```
java -classpath "./lib/EspressAPI.jar;.;"
 quadbase.internationalization.TranslateWizard -file:<fileName> -
enc:<encoding>
```

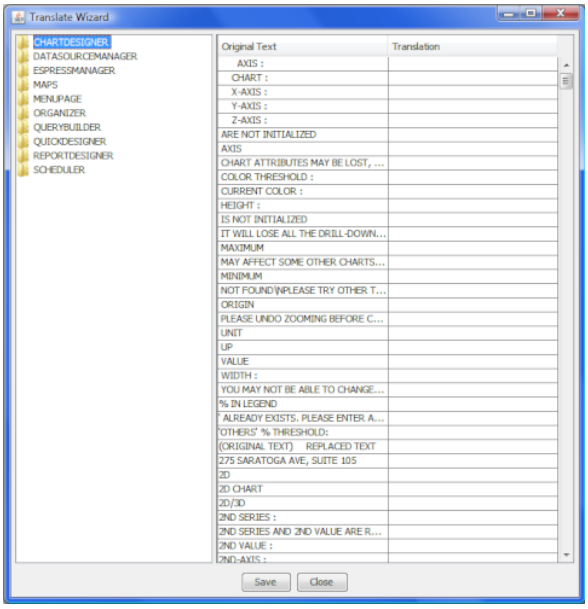
where `<fileName>` represents the xml file that will store the translation and `<encoding>` represents the encoding for the translation. For correct results, the proper encoding must be specified. If the file name and encoding are not provided, the default file name (`Language.xml`) and default encoding (`Cp1252`) will be used.

An example of the above command is given below:

```
java -classpath "./lib/EspressAPI.jar;.;"
 quadbase.internationalization.TranslateWizard -file:Chinese.xml -enc:gbk
```

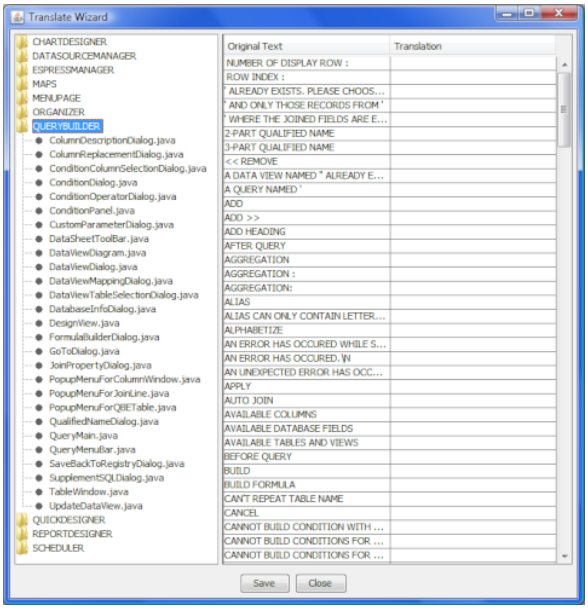
Where `gbk` is the encoding used to save `Chinese.xml` file.

When the Translate wizard is started, it appears as below:



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.



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 EspressChart. For example, the REPORT-DESIGNER and ORGANIZER products do not exist in EspressChart and hence any changes made in those products will not show up.

You can use `Language.xml` (or the copy you modified) in EspressChart by adding the `-file` and `-enc` arguments to the java application or java JNLP (More about Applets in JNLP: Section 2.6 - Run Applets in WebStart with JNLP file) that starts EspressManager, ChartDesigner and/or Scheduler.

13.1.2.1.1. 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 EspressChart directory from a console window and use the following command:

```
java -classpath ".;.\lib\EspressAPI.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 EspressChart, newfile refers to the `Language.xml` included in the upgraded EspressChart 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.

13.1.2.2. Displaying Foreign Characters

Foreign characters can be easily displayed in Designer and Viewer. To display foreign characters in a chart, you will need to have fonts for that language installed in your system. Then, in the 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 Oracle 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 EspressChart (Dialog, Serif, Monospaced, etc.) will display with foreign characters.

13.1.2.3. Entering Foreign Characters

In order to enter foreign characters into any EspressChart interface (for example, ChartDesigner), 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 13.1.2.2 - Displaying Foreign Characters.

Once these settings have been applied, foreign characters can be entered in labels, queries, formulas, and parameters in charts.

13.1.2.4. XML Encoding

By default, EspressChart use UTF-8 character set for encoding when writing to XML. This includes data registry files, XML chart 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 ChartDesigner and EspressManager. For ChartDesigner, you can do this by modifying the `ChartDesigner.bat/sh` file, or modifying the applet page used to launch the ChartDesigner.

To change the XML encoding, add the following argument to the `<EspressChartInstallDir>\ChartDesigner.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 ChartDesigner 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 2.3 - Starting EspressManager.

Apendix A. Parameter Server

A.1. Writing a Parameter Server

Chart Viewer supports push technology by means of interacting with a parameter server. A programmer can write a parameter server using the class `quadbase.chartviewer.ParamServer` to supply updated data continuously for Chart Viewer to plot the chart.

On the client side, the applet(More in: Section 2.6 - Run Applets in WebStart with JNLP file) syntax is either:

```
<applet-desc
name="Chart Viewer"
main-class="quadbase.chartviewer.Viewer"
width="640"
height="480">

<param name="filename" value="example.cht">
<param name="ParameterServer" value="machine:portno">

</applet-desc>
```

or

```
<applet-desc
name="Chart Viewer"
main-class="quadbase.chartviewer.Viewer"
width="640"
height="480">

<param name="filename" value="example.tpl">
<param name="ParameterServer" value=":portno">

</applet-desc>
```

Once the browser has loaded Chart Viewer class and the chart data from the web server, Chart Viewer attempts to connect to the parameter server as specified. In the first case, Chart Viewer will try to connect to the specific machine and port number identified in the parameter. In the second case, a default machine is used: the web server machine. For security reasons untrusted applet(More in: Section 2.6 - Run Applets in WebStart with JNLP file) are not allowed to open a connection to other machines. The parameter server can interact with Chart Viewer in order to update and manipulate the records held by Chart 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.chartviewer.Paramserver`.

Upon opening a connection, the server will supply data to Chart 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 Chart 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 Chart Viewer to repaint the chart using the new data.

The following Java program provides an example of using the parameter server class to generate new charts. Chart 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.chartviewer.ParamServer;

// Sample Program to update the data in EspressChart Viewer
// using Parameter Server
public class pserver extends Thread {

    protected int port = 1997; // port no for chart 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 chart 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 chart 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) {}
        } }

}
```

Chart Viewer would have read a JNLP/JSP file of the form:

```
<?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="filename" value="col2d.cht">
<param name="ParameterServer" value=":1997">
</applet-desc>
<update check="always" policy="always"/>
</jnlp>
```

Here, the Java class that Chart Viewer reads is the name of a chart file `col2d.cht` and the next line specifies the machine and port to which Chart Viewer should open a connection. EspressManager will then provide information to Chart Viewer at regular intervals such as updated records so that Chart Viewer can refresh the chart it displays.

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
```

A.1.2. Constructor

```
public ParamServer(DataInputStream in, DataOutputStream out) throws
 IOException
```

Constructor for Parameter Server

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.chartAPI.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 Chart Viewer to repaint the chart

checkRecord

```
public Boolean checkRecord(Object rec[])
```

Auxiliary function to check whether the record type given matches the record type used in Chart 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

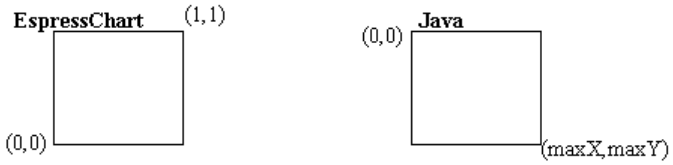
convertRecord

```
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.

Apendix B. Customizing Chart Layout



Component Layout Differences between EspressChart and Java

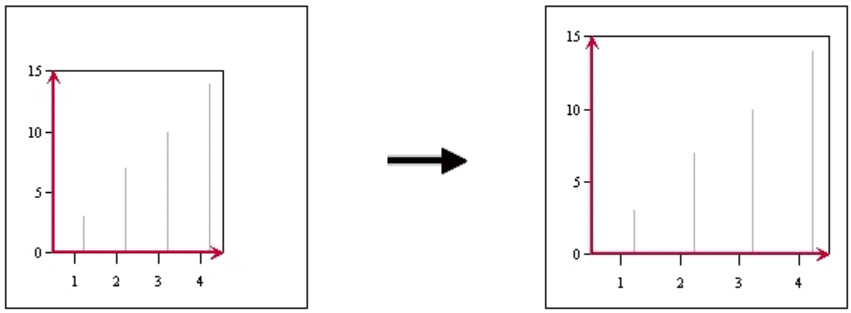
EspressChart describes its component layout in relative coordinates. On the other hand, Java uses layout format in terms of pixels. Due to backward-compatibility issues, the EspressChart layout format will not be changed. The issues pertaining to relative & absolute position is less relevant in Chart 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.

B.1. Changing the Chart Plot Area



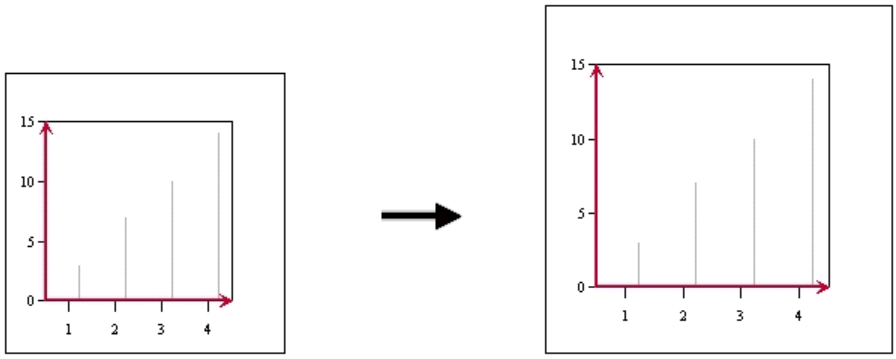
Enlarging the Chart Plot Area

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.

B.2. Changing the Canvas Area



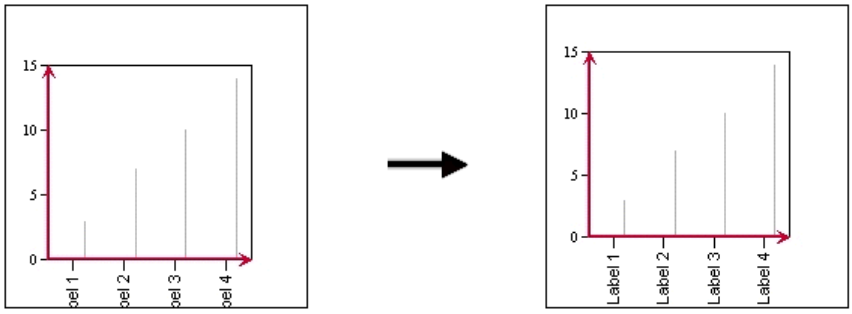
Enlarging the Canvas Area

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.

B.3. Fit Charts Elements

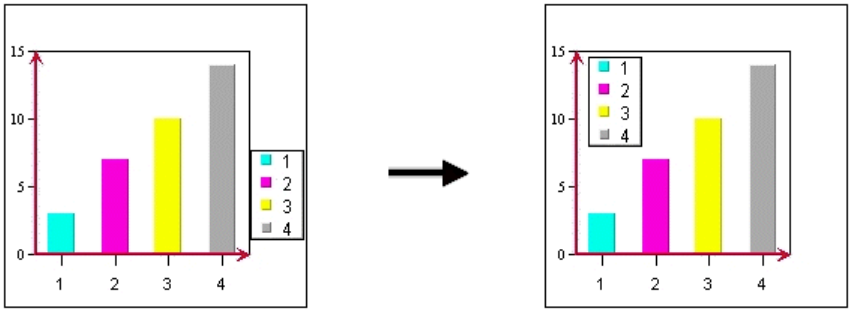


Fitting Chart Elements onto Canvas

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
// EspressChart is able to fit a whole chart
// on canvas. Default value is FALSE
chart.gethCanvas().setFitOnCanvas(true);
```

B.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.

B.5. Attaching Labels to Datapoints

Depending on your needs, it is sometimes necessary to have labels attached to data points. EspressChart 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

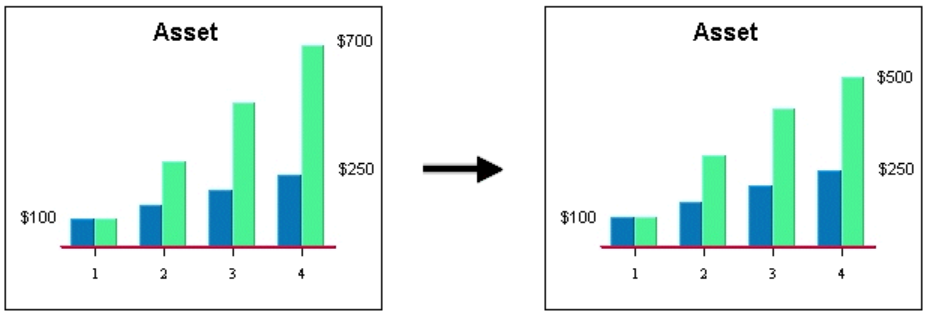


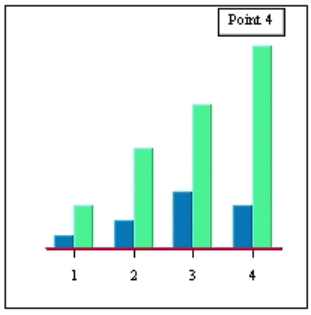
Chart with Original Data Changing to Chart with New Data

```
// 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 EspressChart'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.