

Chart FX 6.2 Manual



SoftwareFX
Any Chart, Anywhere

www.softwarefx.com

Information in this document is subject to change without notice and does not represent a commitment on the part of Software FX, Inc. The software, which includes the information contained in any databases, described in this document is furnished under a license agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of those agreements. It is against the law to copy the software on any medium except as copy of the software for backup purposes. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than the purchaser's personal use, without the express written permission of Software FX, Inc.

Software FX, Inc. disclaim all warranties, either express or implied, including but not limited to implied warranties of merchantability and fitness for a particular purpose, with respect to the instructions contained in this manual.

In no event shall Software FX, Inc. be liable for any damages whatsoever including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss, even if Software FX, Inc. has been advised of the possibility of such damages. Because some states do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.

© 2004 Software FX, Inc. All rights reserved.

Printed in the United States of America

Chart FX is a registered trademark of Software FX, Inc.

Other products and brand names are trademarks or registered trademarks of their respective owners.

Table of Contents

CHART FX 6.2	1
Why Chart FX?	3
About this manual	4
GETTING ADDITIONAL HELP	5
Chart FX Resource Center	7
Dynamic Help / IntelliSense (For Visual Studio and VB users)	8
INTEGRATING WITH YOUR DEVELOPMENT ENVIRONMENT	13
Microsoft Visual Studio .NET	15
Borland C# Builder	16
Delphi 6 and 7	17
Microsoft Visual Basic 6.0	18
Microsoft Access	19
Installing Chart FX for .NET for Web Development Purposes	20
Chart FX for .NET Compact Framework	22
CHART FX GALLERY	28
Bar Charts	29
Stacked Charts	30
3D Bar Charts	31
Area Charts	33
Line Charts	34
Pie Charts	36
Scatter Plots (XY Charts)	38
Financial Charts	39
Bubble Charts	40
Radar and Polar Charts	41
Surface and Contour Plots	42
Pareto Charts	44
Combination Charts	45
Maps	46
Statistical Charts	47

DESIGN TIME EXPERIENCE.....	49
The Chart FX for .NET Design Time Wizard	51
Visual Studio Properties List	56
The Chart FX for Java Designer	57
PASSING DATA TO CHART FX.....	59
Getting Started	61
Series and Points	62
Setting Legends	63
Passing Data Using Databases	68
Passing Data Using XML	69
Crosstab Data Provider	71
VISUAL ATTRIBUTES	75
General Attributes	77
Per-Series Attributes	78
Borders	79
Transparency	81
Color Palettes	82
Gradients	84
Handling Fonts and Titles	87
AXES	91
Introduction	93
Recognizing Numerical and Categorical Axes	94
The Axis Object	95
Axes Scaling	96
Axis Labeling	98
Manipulating Dates and Times in a Categorical Axis	100
Tilting, Inverting and Positioning Axes	101
Axis Gridlines and Tickmarks	102
Interlaced Grids	103
Axis Scrolling	104
Creating Additional Axes	105
Axis Panes	107
Axis Sections	109

DATA ANALYSIS	111
Data Editor	113
Constant Lines	114
Stripes	115
Highlighting	116
Conditional Attributes	118
Per Point Attributes	119
END USER EXPERIENCE	121
Introduction	123
Toolbar & Menubar	124
Context Menus	125
Changing Series and Point Visual Attributes	126
Changing Axis Settings	127
3D Rotation and Perspective	128
Changing Underlying Data Using the Data Editor or Marker Dragging	129
Tooltips	130
Showing Tooltips in Chart Images	131
Exporting Charts to Other Productivity Applications	132
Zooming	133
Highlighting	134
Scrolling	135
Annotation	137
Printing	138
Personalization	139
Drilldown	141
BI Capabilities, Slicing & Dicing Data (Chart FX OLAP)	143
Additional Extensions and End User Experience	144
CONTROLLING THE CHART FX OUTPUT	145
Selecting the Chart's Output	147
Image Map Generation	149
Additional Output Writers	151
Chart Export	153

ADVANCED FEATURES	155
Real Time Charts	157
Customizing the Chart FX Toolbar	160
Working with Subcommands	165
Capturing Mouse Events	168
The Chart FX Annotation Extension	170
Annotation ToolBar	174
Bitstream Chart Generation	175
 PERFORMANCE AND SCALABILITY	 177
Tuning Settings and Features for Performance and Scalability	179
Chart Render Size and Format	180
Chart Return Mechanisms and Web Farms	181
Chart FX PSS	182
 CHART FX EXTENSIONS	 185
Introduction / FAQ	187
Chart FX Maps	188
Chart FX OLAP	192
Chart FX Statistical	194
Chart FX Financial	198
Chart FX Real-Time	201
Chart FX Wireless	203
 APPENDIX A – GLOBALIZATION	 205
 APPENDIX B – PRINTING.....	 209
 APPENDIX C - PASSING DATA USING DATABASES.....	 215
Chart FX for .NET	217
Chart FX for Java	222
 INDEX.....	 225

Chart FX 6.2



Overview

Component software development increases developer productivity, cuts programming time and produces more robust applications, by allowing developers to assemble applications from generic, tested, standardized components. The move to component software, sparked by the success of visual development tools like Visual Basic and Visual Studio, is one of the most prominent trends in the software industry.

For over a decade, Chart FX has been at the forefront of the component industry helping thousands of developers integrate charts and other graphical displays in their applications. Chart FX is not just a simple charting control but a sophisticated data visualization and analysis solution that significantly reduces development costs and improves the responsiveness of the organization by connecting people with information in a graphical way.

Why Chart FX?

Besides a powerful charting engine supporting the most complete collection of chart types, you may also want to consider Chart FX for its following attributes:

Multiplatform.

Chart FX is the only charting and data visualization product capable of delivering native components for the most popular development platforms, including .NET, Java and COM (ASP & ActiveX). Chart FX shares the same charting engine and API across all these platforms allowing developers to easily migrate their applications without the need to retrain or sacrifice the look and feel of the final product. This multiplatform capability is important considering the diverse and heterogeneous nature of IT infrastructures in the modern enterprise.

Extensibility.

While some charting solutions are monolithic, slow and unreliable, Chart FX features a powerful extensibility mechanism. These Chart FX Extension can be stitched together, allowing developers to quickly adapt to data sources enabling organizations to quickly and economically have access to specialized solutions that meet unique business needs. Chart FX Extensions are entire products targeting specific functionality while meeting the needs of specific markets. For example, adding interactive maps or vectorized images to your applications is as easy as integrating Chart FX Maps. In addition you can build a Business Intelligence application by connecting Chart FX OLAP to your multidimensional data source. Please visit our the Extensions section of this manual or our online web site at: <http://www.softwarefx.com/extensions/> for more detailed information.

End User Experience.

Although Chart FX is a development product, it provides many runtime and end user features as well. The Chart FX end user interface significantly increases end users ability to analyze data. For example, end users can access most Chart FX functionality by clicking chart elements or using toolbars and menus on demand. Also, through royalty free runtime components your users can export or reuse charts in most popular productivity applications, such as Microsoft Office.

Scalability & Performance.

Integrating a charting solution to your application or web server could pose crucial server performance, scalability and security issues. Actually, many developers don't become conscious this until they do real-life stress testing and unveil serious performance and scalability issues that may jeopardize a successful or timely deployment of the final application. You will learn, Chart FX has been designed to work in intricate server architectures and you can expect a substantial increase in your Web server's throughput, commonly measured in requests per second and avoid security issues associated with generating and storing server side charts.

Elegant and professional presentation layer

Many vendors have focused on using the OS drawing capabilities to "beautify" or "animate" the charts, in most cases these drawing abuses not only have serious implications in server performance and scalability but also result in chart caricatures or mockeries that make your application look amateurish and provide little or no additional analysis capabilities to end users. Instead, Chart FX offers developers the ability to quickly and easily integrate impressive charts into their applications while retaining blazing speed and performance, making Chart FX the perfect data visualization solution for your application.

Seamless integration and Object-oriented API.

Chart FX seamlessly integrates to the most popular development tools like Microsoft's Visual Studio .NET, Visual Basic and Borland Delphi, among others. Additionally, The Chart FX API is consistent throughout the different Chart FX versions, making it easier and more accessible to create and maintain applications

Code-generation assistants

Chart FX features Wizards and Designers that help you achieve the most stunning charts and at the same time quickly access the code-behind and learn most of the interesting features Chart FX has to offer.

Unparallel Technical Support

When in need you can quickly search our integrated and centralized Resource Center or Support Site (<http://support.softwarefx.com>) for tutorials and samples.

About this manual

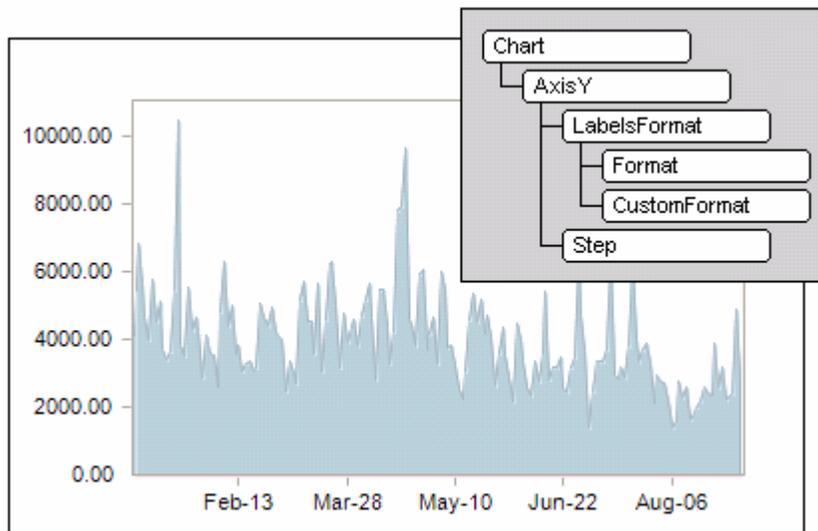
This manual is a guide to familiarize you with Chart FX features. It is not intended to be an API Reference. If you want to refer to the Chart FX API, including Objects, Properties, Methods and Events, please refer to the "Chart FX API Reference". This electronic help file provides detailed information, including syntax, samples and remarks about the Chart FX API.

Which products are covered in this manual?

You can use this manual for the following core products:
Chart FX for .NET 6.2, Chart FX Internet 6.2, Chart FX Client Server 6.2, Chart FX for Java 6.2

How to read this manual?

Instead of writing platform-specific code in each of the sections described in this manual, we have complemented each screen shot with a diagram of the object model, or API used. This diagram serves as a pointer for you to locate the appropriate properties or methods in the Chart FX API Reference.



This writing technique allows us to focus on highlighting product features rather than code while allowing us to maintain a single manual for all the platforms Chart FX supports. This not only helps the environment, but also translates in reduced printing costs and savings that are passed on to our customers.

Finally, most of the samples and features highlighted in this manual are also available electronically which will allow you to simply copy and paste code in your preferred platform and development tool.

Getting Additional Help



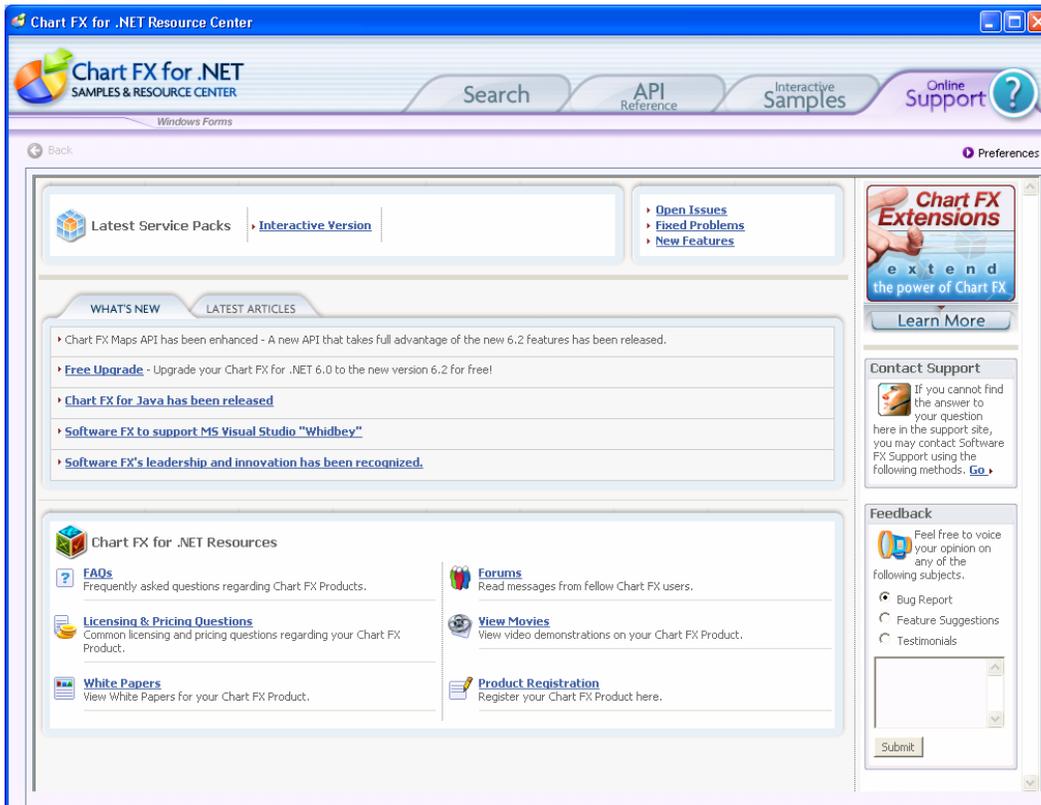
Overview

There are a variety of different resources available to developers when trying to get help using Chart FX. The largest collection of information supporting the product can be found in the Chart FX Resource Center, which includes the API references, tutorial programmer's guide, an Internet reference guide and samples. Below are some additional ways to get help for Chart FX:

- Chart FX Resource Center (API, Tutorial, Internet Reference, Samples)
- Dynamic Help / IntelliSense within Visual Studio .NET
- Javadocs
- Support Web Sites (support.softwarefx.com)
- Newsgroups (news.softwarefx.com)
- 30-Day Free Trial Support
- Paid Support Account

Chart FX Resource Center

The Chart FX Resource Center is a compilation of information supporting .NET, COM or Java development using Chart FX. Included in the resource center are the API references, tutorial programmer's guide, an Internet reference, and plenty of samples to help you find the answers to your questions.



The Chart FX Resource Center features 3 distinct tabs: The API Reference, The Interactive Samples and Support.

The **API Reference** tab documents the supported classes of Chart FX, this section describes every property, method and event available in the component. If you are looking for a particular feature, the API reference is a good starting point.

The **Interactive Samples** tab is designed to allow you to cut and paste code into your projects to quickly produce charts and reduce your learning curve. The samples have also been created in C# and Visual Basic, so you will not have to spend time converting any of the samples.

The **Support** tab (requires internet connectivity) provides a centralized page for accessing our knowledge base and gain access to a vast community of developers using Chart FX.

Finally, you can use the **Search** tab to quickly find about a specific topic or keyword within the abovementioned resources.

Dynamic Help / IntelliSense (For Visual Studio and VB users)

The Chart FX API reference has been integrated into the Visual Studio .NET Dynamic Help. This allows you to access Chart FX help files directly from the Visual Studio .NET development environment.

The screenshot shows the Visual Studio .NET development environment with the Chart FX API Reference and Dynamic Help windows. The main window displays the Chart FX API Reference for the Chart Class, including a bar chart titled "Product Sales" showing sales for Product A, Product B, and Product C from January to May. The Dynamic Help window is open, showing the Chart Class API Reference. The Index window is also open, showing the search results for the Chart Class. The search results window shows the following table:

Title	Location
Chart Class	Windows Forms API Reference
Chart Members	Windows Forms API Reference

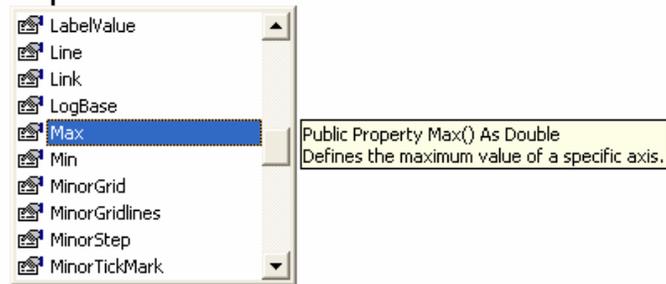
Dynamic Help Window
Main Help Frame
Search/Index Window
Search/Index Results Window

One of the greatest advantages of integrating Chart FX help files within the Visual Studio Dynamic Help is that the Dynamic Help is constantly updating to reference whatever is currently selected by your cursor. Whenever you wish to reference additional information, you can select the appropriate link in the Dynamic Help window or just use the F1 key. Other aids that the Chart FX Dynamic Help integration with Visual Studio provides are:

- Selecting a Chart FX property from the property dialog and pressing F1 will automatically load the corresponding property help page in the Main Help Frame.
- Filtering allows you to select exactly what information the Dynamic Help references.
- When a supported property or method is selected in the Code Editor, a tooltip will be displayed with a description of that property or method.
- When a property is selected in the property dialog, a description of that property can be seen in the description window below the dialog.
- When the Chart control has focus in the design form, pressing F1 will activate the Dynamic Help specific to the chart.

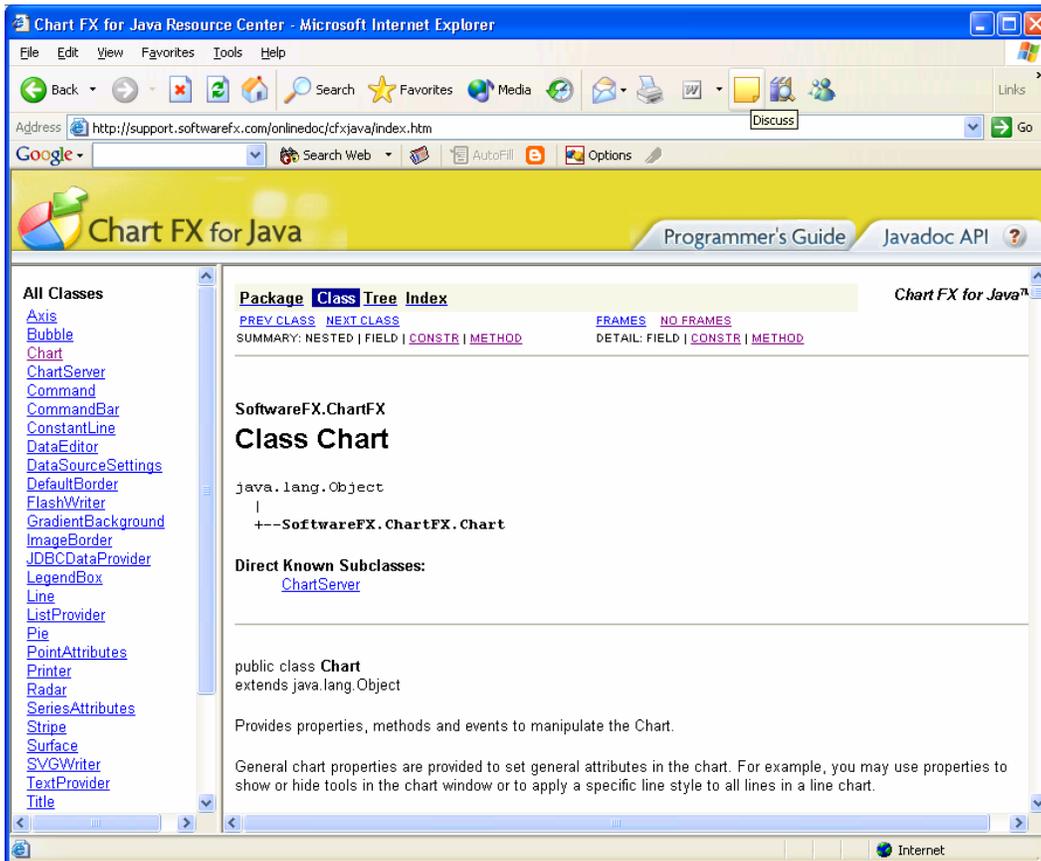
The IntelliSense feature in Visual Studio can also act as an aid by prompting developers with possible coding options and information. When a function or statement is entered into the Code Editor, its complete syntax and arguments are shown in a ToolTip. In addition, languages that supports early binding display available functions, statements, constants, and other values in a list that you can choose from.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.  
    Chart1.BackColor = Color.White  
    Chart1.BorderObject.Color = Color.Black  
    Chart1.AxisX.
```



Javadocs (Chart FX for Java users only)

As with most Java related software, Javadocs is the standard for API documentation and contains the classes, properties and methods pertaining to various modules of Chart FX for Java. The Javadocs are located online at: <http://support.softwarefx.com/online/doc/cfxjava/index.htm> along with Programmer's Guide.



The screenshot shows a Microsoft Internet Explorer browser window displaying the Chart FX for Java Javadoc page. The browser's address bar shows the URL <http://support.softwarefx.com/online/doc/cfxjava/index.htm>. The page features a yellow header with the Chart FX for Java logo and navigation tabs for "Programmer's Guide" and "Javadoc API".

On the left side, there is a "All Classes" sidebar listing various classes such as [Axis](#), [Bubble](#), [Chart](#), [ChartServer](#), [Command](#), [CommandBar](#), [ConstantLine](#), [DataEditor](#), [DataSourceSettings](#), [DefaultBorder](#), [FlashWriter](#), [GradientBackground](#), [ImageBorder](#), [JDBCDataProvider](#), [LegendBox](#), [Line](#), [ListProvider](#), [Pie](#), [PointAttributes](#), [Printer](#), [Radar](#), [SeriesAttributes](#), [Stripe](#), [Surface](#), [SVGWriter](#), [TextProvider](#), and [Title](#).

The main content area displays the documentation for the `SoftwareFX.ChartFX` package, specifically the `Class Chart`. It shows the class hierarchy: `java.lang.Object` is the superclass, and `SoftwareFX.ChartFX.Chart` is the subclass. The page also lists "Direct Known Subclasses" as [ChartServer](#).

The class definition is shown as:

```
public class Chart
    extends java.lang.Object
```

The description states: "Provides properties, methods and events to manipulate the Chart." and "General chart properties are provided to set general attributes in the chart. For example, you may use properties to show or hide tools in the chart window or to apply a specific line style to all lines in a line chart."

Support Web Site

Support for Chart FX is available on the web for free. <http://support.softwarefx.com> was designed to help you quickly and easily find the information you need. In this site, you can view knowledgebase articles, sample code, documentation, product updates and other important technical support information. In order to use the support site, you are required to have a Windows-based system with a browser that supports client-side components (both Internet Explorer and Netscape are supported!).

Software FX Support Site:
<http://support.softwarefx.com>

Support Email: support@softwarefx.com

Newsgroups

Newsgroups are a great way to receive feedback from the Chart FX developer community of users as well as Software FX Support Team. The Software FX staff is dedicated to providing the best products possible, so monitoring the Newsgroups is a daily routine, plus access to the Newsgroups is free. You can access the Software FX Newsgroups at:

<news://news.softwarefx.com>

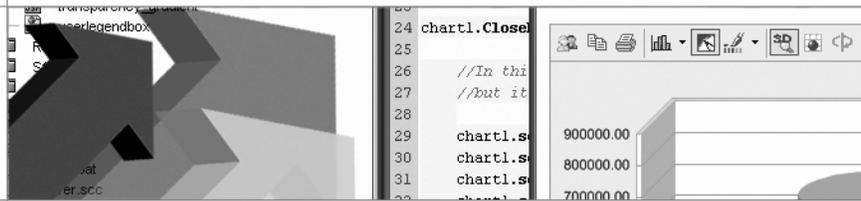
30-Day Trial and Paid Support Accounts

Registered users of Chart FX may sign up for a support account online. The account is free for the first 30 days. If you require further assistance after 30 days, you may renew your support account for a yearly fee. You can create your support account at the following URL:

<http://support.softwarefx.com/Accounts>

Support Account Phone: 561-392-2023, from 9AM to 5PM EST.

Integrating with your Development Environment

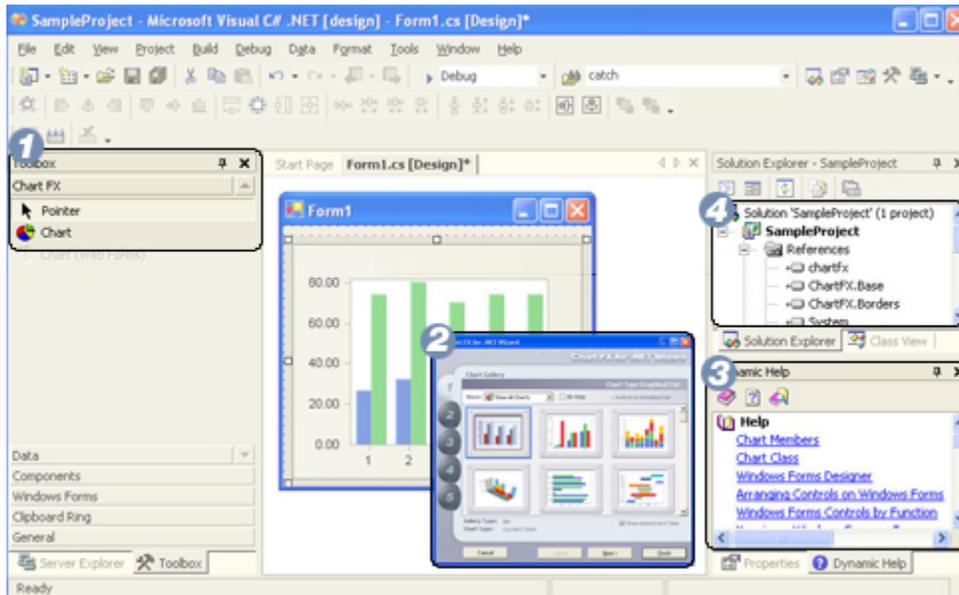


Overview

The following sections will describe the procedures required to integrate Chart FX into your preferred development environment. Development tools allow for developers to combine the visual components with code in order to create your final applications. Chart FX as also integrated other features into supported IDE's (VS.NET) such as context sensitive help and design time wizards.

Microsoft Visual Studio .NET

Chart FX for .NET has many robust features, some of which are specifically designed for users of Microsoft Visual Studio .NET. IntelliSense functionality, context sensitive help, design time wizards, etc, are all available when selecting Microsoft Visual Studio .NET as your development environment.



Automatic Integration from Installation

When you install Chart FX for .NET on a machine with VS.NET, the Chart FX installer automatically configures various critical design time areas within VS.NET.

1. A new Toolbox tab is added called Chart FX as shown above. Within the tab, two chart objects are created – one for WinForm applications and the other for WebForm applications. These controls become enabled when you enter design time mode with either a Win or Web form respectively.
2. The Chart FX wizard automatically launches upon adding a chart to your form. This Wizard aggregates the most commonly used features into a simple step by step process – allowing you to obtain your desired chart with little interaction with code or the properties box.
3. The associated help document for the entire API along with some sample code is conveniently integrated with VS.NET. As you develop, the dynamic help displays topics relative to the code you are currently working on.
4. Chart FX automatically includes the necessary references for supporting your chart within your application.

Note: If VS.NET is open while the Chart FX installer runs, the toolbox tab may not be created. To create it manually and add Chart FX, you will need to browse the .NET Controls list or simply browsing to the appropriate assembly on your computer.

Borland C# Builder

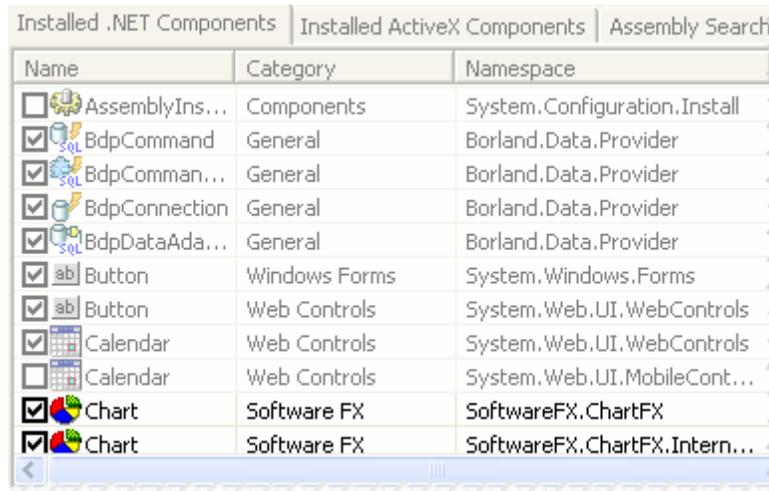
Chart FX for .NET may also be integrated into Borland's C# Builder as well. Below are the steps to integrate Chart FX in C# Builder:

Windows Forms Integration

- Right-click on the Tool Palette background
- Select Customize .NET Components
- Press the "Select an Assembly..." button
- Locate and Select the "**ChartFX.dll**" assembly in the Chart FX installation directory.

Web Forms Integration

- Right-click on the Tool Palette background
- Select Customize .NET Components
- Press the "Select an Assembly..." button
- Locate and Select the "**ChartFX.Internet.dll**" assembly in the Chart FX installation directory.

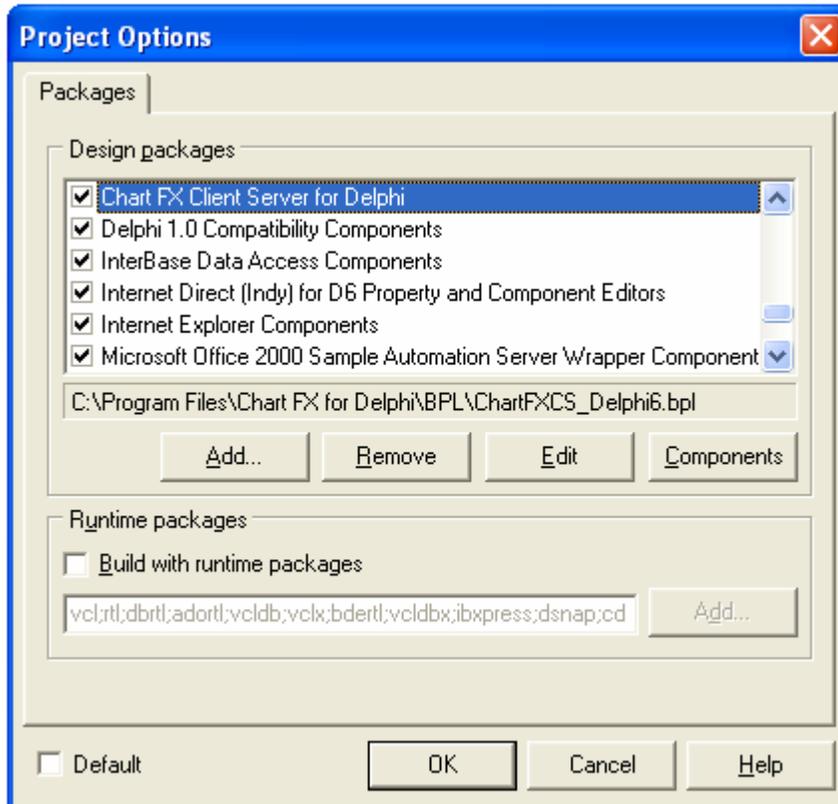


Delphi 6 and 7

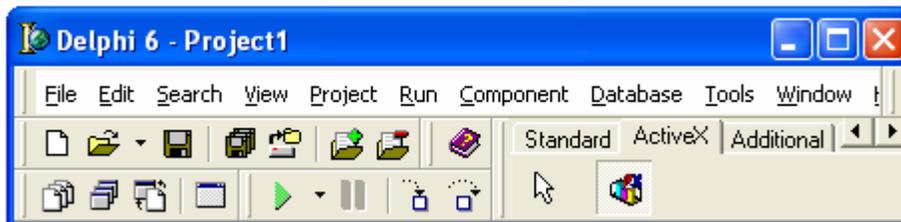
Chart FX Client Server includes a pre-assembled wrapper that ensures Chart FX will function exactly how developers expect. Below are the steps to integrate Chart FX for Delphi.

Adding Chart FX .bpl to the Delphi Toolbox

- Open your Borland Delphi IDE.
- Select Component -> Install Packages from IDE menu to access the Project Options dialog.
- Select the Add and browse to the appropriate install package (.bpl) for your version of Borland Delphi and click OK.
 - C:\Program Files\Chart FX for Delphi\BPL\ChartFXCS_Delphi6.bpl (Delphi 6)
 - C:\Program Files\Chart FX for Delphi\BPL\ChartFXCS_Delphi7.bpl (Delphi 7)
- Click OK.



The Chart FX control will be added to the ActiveX tab of the Delphi toolbar that can be selected and dragged to your application form:

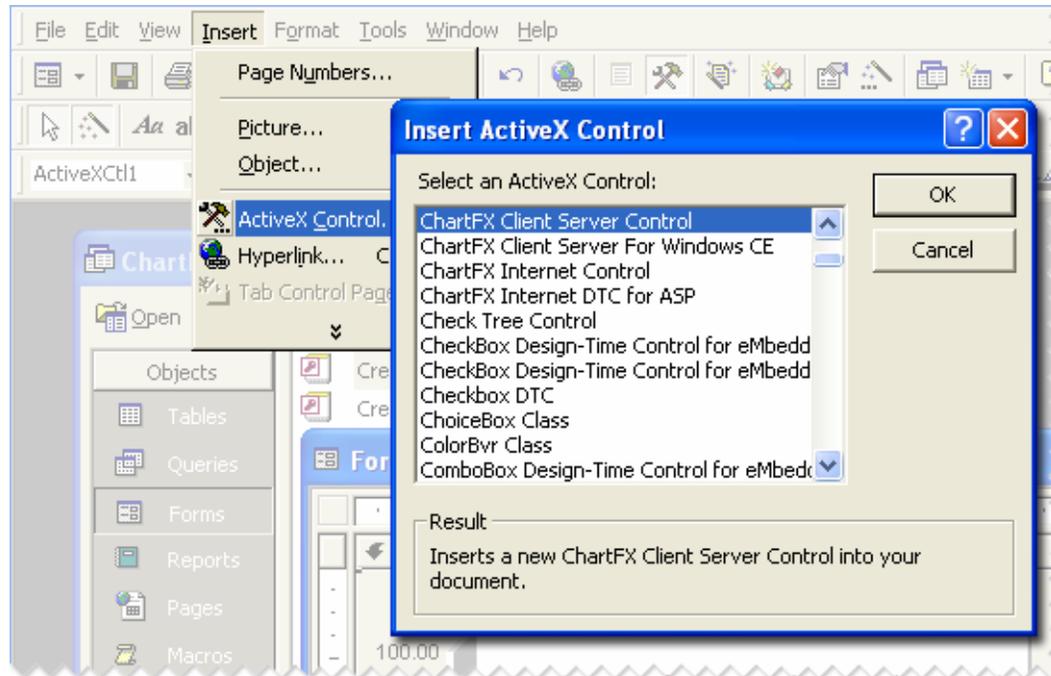


Microsoft Visual Basic 6.0

Chart FX Client Server may be integrated into Microsoft Visual Basic 6. Here are the steps to include the component and reference the necessary module file:

To Include the Chart FX ActiveX in your project

- From the Project menu select Components...
- Choose Chart FX Control from the Controls list.

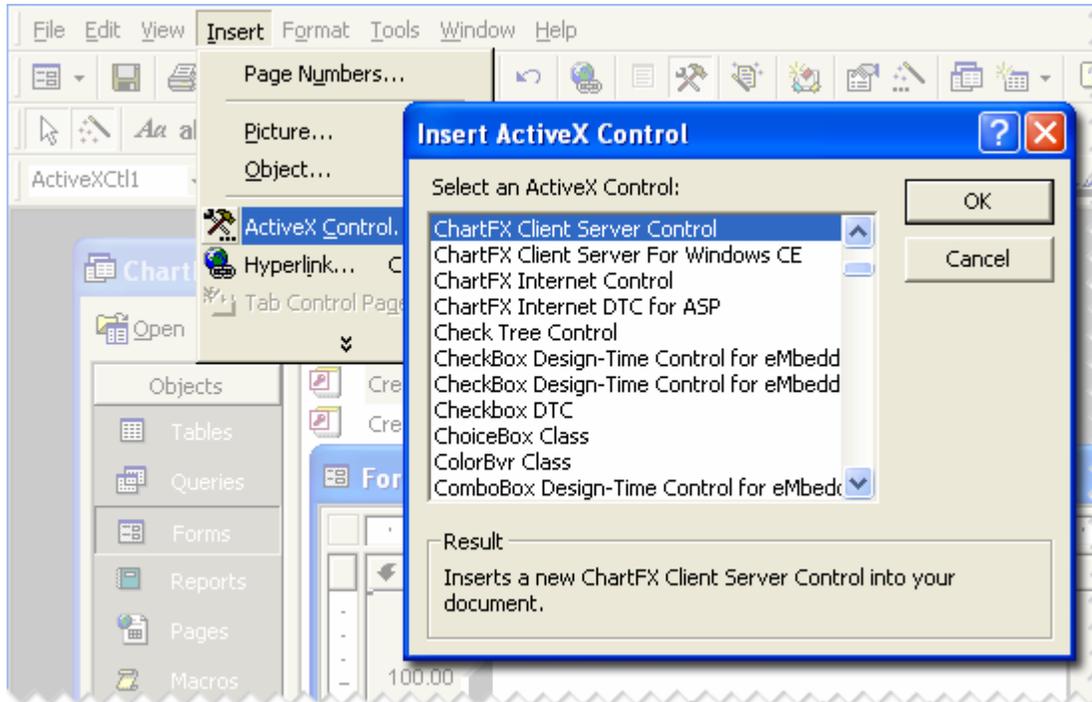


Microsoft Access

Chart FX Client Server may also be integrated into Microsoft Access Forms. The following are the steps needed to include the component.

To Include the Chart FX ActiveX control in your Access form or report

- Open your form/report for design
- From the Insert menu select ActiveX Control...
- Resize the control to the desired size



Installing Chart FX for .NET for Web Development Purposes

Before you install Chart FX for .NET, you must have previously installed the following components on your development machine: .NET Framework & Internet Information Server (IIS)

Chart FX for .NET provides 2 distinct installations:

Development Seat: This will copy and configure all the necessary files, including documentation and samples, on the machine that has been provisioned with an appropriate development environment. This installation will setup all the necessary assemblies so you can integrate charts and debug your Web Forms and Windows Forms (if applicable) projects.

Please note a Chart FX test server assembly will be configured on your local machine so you can test web applications before they are deployed to a production server.

Production Server: This will copy and configure all the necessary files and components on your production or "live" server to respond to ASP.NET pages that display charts. Please note this setup file should not be used on the machines that are used for development purposes.

Important Licensing Information: Some of the Chart FX for .NET assemblies are restricted from distribution. Please make sure you read and abide by the Chart FX for .NET License Agreement before installing, using or deploying assemblies on multiple servers or development machines.

Chart FX for .NET and Single Page Web Applications

While Web Forms introduces a new programming model for web applications built around server-side controls, many programmers will still prefer to use and code in ASP.NET in the same way ASP applications are built today; that is combining ASP.NET and HTML code in a single file.

Unfortunately, most IDEs are oriented toward the two-file, code-behind model, and there is limited support for single-file pages. Nevertheless, a single-file page has the advantage that it can be easier to deploy, since the source code is deployed along with the Web Forms page. Essentially, the code is physically in the .aspx and compiled automatically when it is first loaded.

Integrating a chart into a single .aspx file is as simple as creating and instantiating the chart object, setting the desired chart properties and finally invoking a method (**GetHtmlTag** or **GetHtmlData**) that allows the Chart FX server component to generate the appropriate HTML output (images or active components) supported by the browser posting the page request. Please see the Resource Center for a single .aspx page sample.

Configuring your Web Server for Single Page Web Applications

When you create a WebForm application using VS.NET and Chart FX for .NET, an application directory is created in IIS for the project folder and all of the dependencies for the web application are copied from the installation directory to a newly created /bin subdirectory.

If you copied a single page web form application into that application folder, the Chart FX control would use the /bin folder of that application to locate the required assemblies for compilation. However, if you placed the .aspx file containing the chart in a regular directory that did not have the /bin folder containing the dependencies, your chart would fail.

In order to run these single page web applications, you will need to recreate the /bin folder. You can do so by creating a /bin folder on the root directory of your web server (i.e. **c:\inetpub\wwwroot\bin**). Inside this folder, you will need to place all the Chart FX for .NET assemblies your charts require. You may also place your **ChartFX.Internet.Config** file in this location.

By default the core dependencies for Chart FX for .NET (WebForms) are:

ChartFX.Internet.dll
ChartFX.Base.dll
ChartFX.Borders.dll

Note: When manually creating this directory, you must maintain the contained assemblies yourself. The service pack update will not automatically update these components.

ChartFX.Internet.Config

Chart FX for .NET supports the use of configuration files that may be used to store Web form application settings. A variety of different settings may be controlled from this XML configuration file. Some of the supported attributes include the **Relative path**, **RequiredRoot**, **DefaultWriter**, etc. Please review the table below for a complete explanation of the supported configuration tags:

Name	Tag	Description
Relative	<Relative>	Specifies the Relative path that will be used to generate and store temporary chart files.
Required Root	<RequiredRoot>	Specifies the Required Root URL used for downloading chart files and client assemblies.
Default Writer	<DefaultWriter>	Used to configure the image writer used for all charts. Flash, SVG, etc.
Default Writer Settings	<DefaultWriterSettings>	Used to configure attributes for the configured DefaultWriter. Useful mostly with PSS.
Absolute	<Absolute>	Specifies the Absolute path that will be used to generate and store temporary chart files.
Extensions Folder	<ExtensionsFolder>	Specifies the folder used to locate the ImageWriter Extension server assemblies.
Extensions Mask	<ExtensionsMask>	Specifies the 'mask' used for finding the ImageWriter extension assemblies. Default is "ChartFX.Writer".
MainClient	<MainClient>	Used to set the MainClient property True/False.
Client Version	<ClientVersion>	Specifies the MainClient assembly version.
Trace Mode	<TraceMode>	Allows you to customize the detail level of Chart FX error messages.

To utilize this configuration, developers must include this file in the /bin directory of web application residing on the server. At chart creation, this configuration file will be loaded and used by the Chart FX server assemblies. Therefore it is required that this file be manually copied to the web server.

Here is an example of a valid ChartFX.Internet.Config file:

```
<CfxIESettings>
  <Relative>/ChartFX62/Temp</Relative>
  <RequiredRoot>http://yourServer.com</RequiredRoot>
</CfxIESettings>
```

Chart FX for .NET Compact Framework

Pocket Chart FX for .NET is a charting component for rapidly developing applications that extends enterprise data visualization and analysis capabilities to mobile devices. With Visual Studio® .NET, and Chart FX® for .NET Compact Framework you can quickly build powerful graphical applications that connect to your mission critical data and run on Smart Devices, anytime, anywhere!

The Pocket Chart FX for .NET provides the following advantages:

1 Third-generation charting component for mobile and smart devices

As opposed to other vendors which provide first-generations of their .NET Compact Framework products, Software FX has been building and offering charting components for Microsoft's mobile environments for over a decade. Actually, many mission critical applications have been built and successfully deployed with Pocket Chart FX, ActiveX predecessor to the Pocket Chart FX for .NET. This has allowed us to learn and adapt to real development needs rather than "featurism" or "demoware" which provides little value to developers and their mobile applications. With Pocket Chart FX for .NET you can expect a component with the right feature set, portability and memory footprint for your mobile applications.

2 Adherence to .NET Compact Framework design guidelines (runs 100% natively in Mobile devices running .NET Compact Framework)

As a GDI+ intensive component, Chart FX has been developed with coding practices that can help improve draw speeds when using Graphics draw calls. This is particularly important when considering applications in a mobile platform where memory, CPU speed, and other resources are at a premium. For example, Chart FX minimizes the number of draws that occur by keeping track of the items that need to be redrawn thus improving performance of the final application. Additionally, considering that a PocketPC screen resolution is typically 240x320, we made sure chart elements that usually take a lot screen space were removed from the default chart settings and charts accommodate and display well in a "portrait" rather than a "landscape" orientation; even the Chart FX default color palette has been changed to improve end users ability to read charts in small Pocket PC screens.

3 Support for other mobile related technologies and tools like SQL Server CE and Pocket Access

SQL Server CE extends Microsoft SQL Server to Microsoft Windows CE-based mobile devices, while providing developers with a consistent programming model for rapid application development. Chart FX can read from native SqlCEConnections and SqlCEDataAdapters for easy chart population. The following C# code demonstrates how easy it is to populate a chart from SQL Server CE data source.

```
SqlCeConnection myConn = new SqlCeConnection(@"Data Source = \Test.sdf");
myConn.Open();
SqlCeCommand myCommand = new SqlCeCommand(sSql, myConn);
SqlCeDataAdapter sqlCeAdapter = new SqlCeDataAdapter(myCommand);
DataSet dsChart = new DataSet();
sqlCeAdapter.Fill(dsChart);
```

With SQL Server CE developers can access essential relational database functionality in a small footprint: a robust data store; an optimizing query processor; and reliable, scalable connectivity capabilities. Additionally, Chart FX can read from Pocket Access2 or if you are populating the chart from other data sources like text or XML files, you can use an easy and accessible Pocket Chart FX for .NET generic API to populate charts.

If you are new to SQL Server CE we recommend the following resources:

<http://msdn.microsoft.com/msdnmag/issues/01/06/sqlce/>

<http://samples.gotdotnet.com/quickstart/CompactFramework/doc/adonetdatabinding.aspx>

or refer to the SQL CE on the Microsoft web site at:

<http://www.microsoft.com/sql/ce/>

4 Small footprint assemblies provide simple deployment as well as a rich design-time experience.

Pocket Chart FX for .NET provides a design-time experience assembly compiled against the .NET Framework (ChartFXCFDesigner.dll) that integrates seamlessly in Visual Studio .NET allowing you to setup chart attributes and properties easily. This assembly provides identical capabilities and features as the run-time assembly compiled against the Compact Framework (ChartFXCF.dll). This consistency ensures there are no features in the design time version of the control that will not be available in the run-time version limited to work in the .NET Compact Framework. Additionally, measuring around 300KB both assemblies adhere to the "small is good" principle for mobile applications.

5 A familiar API consistent with other Chart FX products increasing developer productivity by minimizing the learning curve while allowing you to easily port your existent desktop or web based applications to mobile devices with minimal effort.

Perhaps one of the main advantages Chart FX provides is a consistent API and Object model throughout the platforms that are supported throughout its family of products (.NET, COM and Java). This means, developers can leverage their knowledge in a particular Chart FX product to move or port an application to a completely different platform. Even though Pocket Chart FX for .NET is a subset of our server and desktop based products, developers can expect to have the same development experience, ease-of-use and API when integrating Pocket Chart FX for .NET in their mobile applications.

6 Free for development purposes to existent Chart FX for .NET or Chart FX Developer Studio users.

Pocket Chart FX for .NET is bundled with our leading .NET products. If you are an existent Chart FX for .NET or Chart FX Developer Studio owner please visit our support site or contact our sales department to obtain your free copy of Pocket Chart FX for .NET. Runtime fees may apply. For additional information on licensing and pricing for deployment please contact our Sales department at sales@softwarefx.com

Is the entire Chart FX functionality available in .NET Compact Framework(CF) version?

No! Unfortunately, due to the constraints of mobile devices, the .NET Compact Framework is a subset of the .NET Framework. This means many features of Windows, GDI, and the .NET Framework that you have come to know and use in your Windows Forms or Web Forms applications are not available. Additionally, there are also a number of features that have been simplified and have a special impact in some effects while displaying chart within .NET CF environment. For example, setting the pen width is not available in the .NET CF. This means you will not be able to change width for any of the borders or lines in a chart. In general, this means you will see a variety of differences between the full-featured Chart FX for .NET and the Pocket Chart FX for .NET.

As a directive for performance reasons, we specifically decided to drop functionality from Pocket Chart FX for .NET rather than re-implement it.

Besides limitations imposed in the .NET Compact Framework itself, Pocket Chart FX for .NET also lacks the end user experience their desktop and browser-based counterparts offer. For example, Pocket Chart FX for .NET does not display toolbars or dialogs end users usually interact with to further customize charts. This rather difficult decision helps preserve the limited screen real estate in mobile environments and avoid UI conflicts with your application. In any case, implementing this functionality is simple property settings in your UI. For example, if you want to change the gallery type you can simply add a menu item to your application and set a chart property when the user selects the menu item.

Note: Although Pocket Chart FX for .NET does not provide built-in UI you can process mouse related events to implement drill-down and process other mouse related interactions. For additional information and samples, please visit the Chart FX for .NET Resource Center.

Getting Help and sample code

The Chart FX Resource Center not only provides specific samples for Pocket Chart FX for .NET but has also been provisioned with the .NET Compact Framework emulators so you can see each sample running in a virtual Pocket PC environment. To activate, simply select the Smart Devices radio button at the top of the Resource Center. The following picture depicts a live Pocket Chart FX for .NET sample in the Resource Center:

The screenshot displays the Chart FX for .NET Resource Center interface. The top navigation bar includes 'Search', 'API Reference', 'Interactive Samples', and 'Online Support'. The main content area is titled 'Windows Forms' and shows a 'Type of Application' dropdown set to 'Smart Devices'. The left sidebar contains a tree view of sample categories, with 'Axis Handling' expanded. The central code editor shows the following C# code:

```
//Populating the chart1
chart1.OpenData(SoftwareFX.ChartFX.CF.COD.Values, 2, 5);
chart1.Value[0, 0]=0.5;
chart1.Value[0, 1]=0.75;
chart1.Value[0, 2]=0.91;
chart1.Value[0, 3]=0.98;
chart1.Value[0, 4]=0.72;

chart1.Value[1, 0]=232815;
chart1.Value[1, 1]=678192;
chart1.Value[1, 2]=655234;
chart1.Value[1, 3]=891230;
chart1.Value[1, 4]=533988;

chart1.CloseData(SoftwareFX.ChartFX.CF.COD.Values);
SoftwareFX.ChartFX.CF.SeriesAttributes series0 = chart1.Series[0];
SoftwareFX.ChartFX.CF.SeriesAttributes series1 = chart1.Series[1];

series0.Gallery = SoftwareFX.ChartFX.CF.Gallery.Area;
series1.Gallery = SoftwareFX.ChartFX.CF.Gallery.Bar;

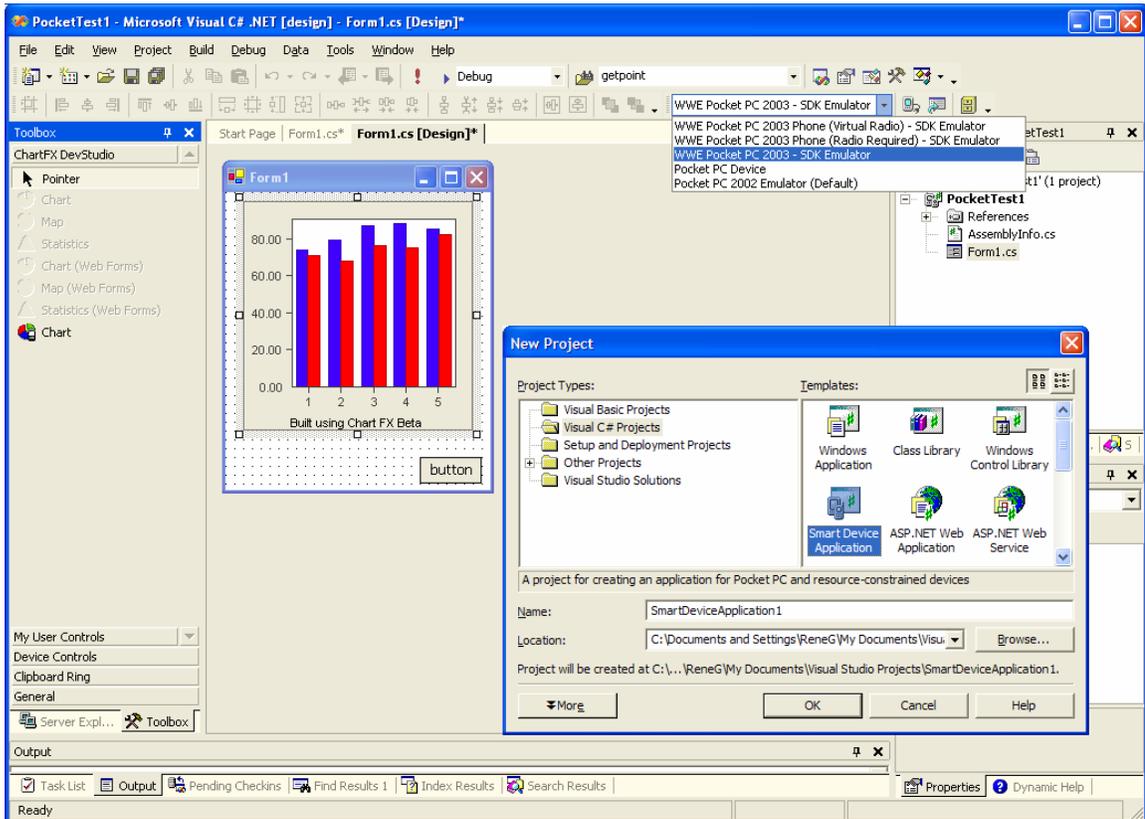
series0.YAxis = SoftwareFX.ChartFX.CF.YAxis.Secondary;
SoftwareFX.ChartFX.CF.Axis axis = chart1.AxisY2;
axis.Min = 0.5;
axis.Max = 1;
axis.Step = 0.05;
axis.Visible = true;
axis.TextColor = Color.Blue;
axis.GridColor = Color.Blue;
chart1.AxisY.Gridlines = true;
```

Below the code editor, there are options for 'Code Format': Standard Code, Cosmetic Code, and Comments. On the right, a virtual Pocket PC emulator displays a bar chart with two series: a blue area series and a green bar series. The chart has a secondary Y-axis on the right side. The emulator screen shows a 'Form1' window with a time of 5:50.

Additionally the Chart FX Resource Center features an API Section where all Chart FX methods and properties can be accessed. Please be aware that those objects, properties and methods supported in the .NET Compact Framework will be clearly marked with "Supported by the Pocket Chart FX for .NET" to distinguish them from the standard Chart FX for .NET API.

Creating your first Smart Device application with Pocket Chart FX for .NET

The .NET Compact Framework is installed with Microsoft Visual Studio .NET 2003. When you create a smart device project in Visual Studio .NET, your project references the .NET Compact Framework assemblies and files. The following screen depicts a current project containing a chart control and the Create New Project option for Smart Devices dialog:



Once Visual Studio .NET has finished creating the project open Form1, you can simply select the Chart control that has been designed for Smart Devices projects. This icon will be automatically integrated by the Chart FX setup and can be found in a separate tab in the Visual Studio .NET Toolbox. Simply Drag it onto the form, resize it to fill most of the form, and then double click on the form to begin editing the Form1_Load event handler. When the form loads, we need to place some data into the form. For additional information on how to populate the chart with data please refer to the Pocket Chart FX for .NET documentation.

Note: If you can't locate the Pocket Chart FX for .NET icon in any of the tabs available in Visual Studio simply right click on the toolbar and select Customize Toolbox. When the Customize Toolbox dialog box comes up, select the .NET Framework Components tab and click Browse. Navigate to the Chart FX default install directory and select ChartFXCFDesigner.dll. Click OK and you should see your control appear in the Toolbox.

How do I deploy an application using Pocket Chart FX for .NET?

When you compile and/or deploy your application, the Smart Device Extensions substitute the design time version of your assembly with the runtime version. This is done via a new attribute exposed in System.CF.Design.dll. This new attribute, RuntimeAssemblyAttribute, allows you to specify the assembly that should be used at compile/runtime to replace the design time assembly. The attribute takes a single argument, the fully qualified type name of your runtime assembly. The full name of the Pocket Chart FX for .NET assembly looks like:

```
"ChartFXCF, Version=6.2.1630.25984, Culture=neutral, PublicKeyToken=null"
```

The first part is the name of the assembly without the file extension ".dll" on the end. The second part is the version of the assembly. The third part is the culture; if you have not defined one, neutral is the appropriate setting. The fourth part is the public key of the strong name. If you have NOT signed (applied a strong name) the assembly, null is the appropriate value. You use the fully qualified type name with the RuntimeAssemblyAttribute like this:

```
[assembly: System.CF.Design.RuntimeAssemblyAttribute("ChartFXCF, Version=6.2.1630.25984, Culture=neutral, PublicKeyToken=null")]
```

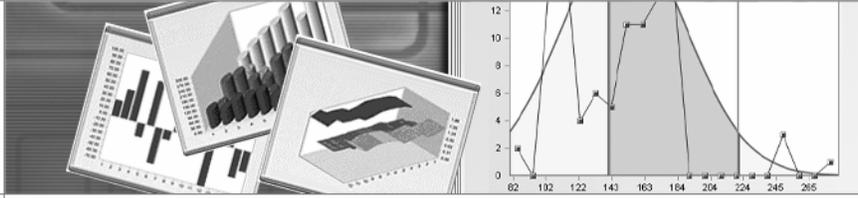
Note: As soon as you attempt to run this application you will see a message in the chart placeholder indicating you should invoke SetLicenseString to run the chart. For deployment, you may be subject to run time licensing fees. For more information on licensing, you should contact Software FX to obtain valid license information for Pocket Chart FX for .NET. Please contact Software FX for additional pricing and licensing information.

Requirements:

Pocket Chart FX for .NET is supported on Pocket PC 2000, Pocket PC 2002, Windows Mobile 2003-based Pocket PCs and Smartphones and embedded systems running Windows CE .NET 4.1 and later.

Additional information on the .NET Compact Framework available at:
<http://msdn.microsoft.com/mobility/prodtechinfo/devtools/netcf/default.aspx>

Chart FX Gallery



Overview

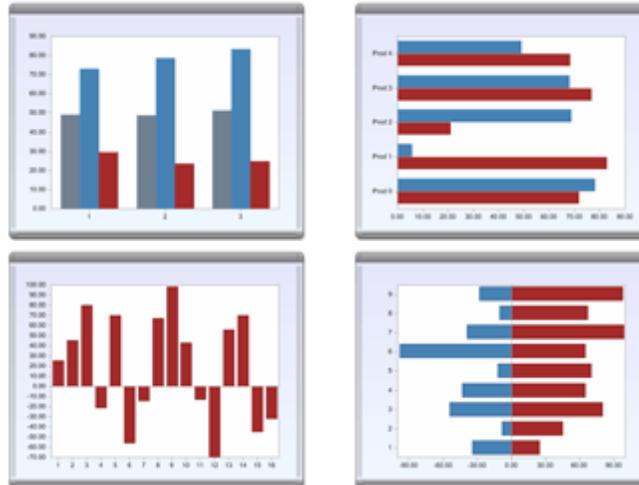
Data can be represented in many formats. Chart FX provides a multitude of gallery types to address these needs of exposing data – from the most simple bar or pie chart to the most complex statistical or financial charts. This chapter briefly describes the available galleries for Chart FX.

Bar Charts

Bar charts are the most common type of business chart and are especially useful for comparative analysis. You can use a bar chart to illustrate comparisons among individual items. Bar charts are also useful for comparing classes or groups of data. In bar charts, a class or group can have a single category (single series) of data, or they can be broken down further into multiple categories (multiple series) for greater depth of analysis.

Passing data to a bar chart is as simple as passing one or more data series and tags or labels accompanying each data series. These labels will be displayed in the categorical or time axis (X Axis).

When more than one series is passed to the chart, Chart FX will display, by default, the bars side by side in a 2D style. For example, if you want to represent projected vs. actual revenue for the different months of the year, the bar chart will be the best chart type to use. The following figure depicts several bar charts:



Bar and Column charts

Some charting vendors refer to column and bar terms in different ways. As a rule of thumb, a column chart is a chart displaying vertical bars while the bar chart term is commonly used for horizontal bars. However, to some people this difference is not decipherable. For this reason, Chart FX utilizes the term Bar chart in the same way for both, vertical or horizontal, displays.

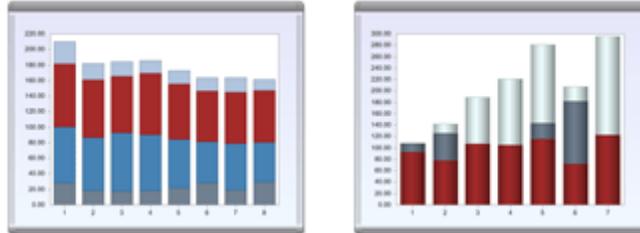
To make a bar chart horizontal, please use the Gantt chart type. For additional information please check the Chart FX API in your electronic documentation.

Stacked Charts

Another way to represent bar charts is through stacking. In this way, a single bar on the chart can show data for more than one category of data. Stacked bar charts show the relationship of parts to the whole. Stacking techniques differ depending on whether you are representing divisions within data label categories or stacking two separate numeric categories.

Stacked bar charts offer similar complexity to clustered bar charts by adding together component value items within chart bars or areas. By stacking items and assigning a different color to each item, you can effectively display trends among comparable or related items, or visually emphasize a sum of several indicators.

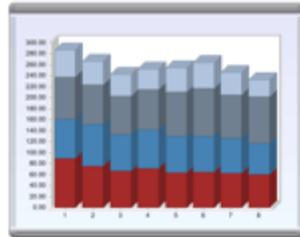
To create a Stacked bar chart, you need more than one series. Each series contains the additive value rendered as an additional segment along the length of the bar. Additionally, Chart FX lets you stack the bar charts vertically or horizontally as well as in groups of stacked bars as depicted in the following figure:



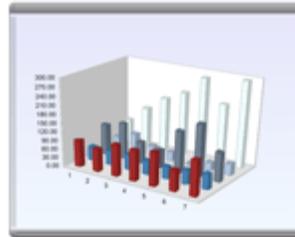
3D Bar Charts

Chart FX allows you to set a 3D look to your bar charts. Sometimes these charts are harder to read and analyze but present a nicer look for presentation purposes. Chart FX supports two types of 3D effects: Isometric and Oblique projections.

In Isometric 3D View, charts appear in the chart space as 3D objects with depth. In contrast, the Oblique 3D view allows you to set rotation angles and a perspective value for your charts. The following two pictures depict an Isometric and Oblique projections, respectively:



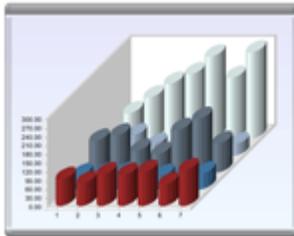
Isometric Projection



Oblique Projection

Note: That does not mean that you are plotting three dimensions of data or using three dimensions of space to represent data (XYZ Coordinates). In contrast, Chart FX will always use two dimensional data. In other words, 3D is simply a visual effect that can be turned off.

Clustered charts



Cluster normally applies to 3D bar charts and this effect allows you to change your chart perspective so that the Z-axis data extended in the third dimension is shown as clusters displayed in the foreground. This charting option is useful when Z-axis bars are hard to distinguish in standard bar formats.

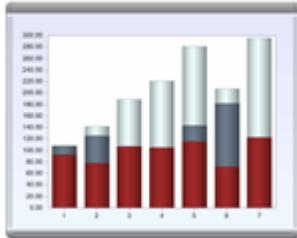
To create a clustered bar chart, you need more than one series. Each series occupies its own position along the Z-axis data. The following picture depicts a clustered bar chart:

In 2D bar charts, you can use clustered bar charts to juxtapose categories in one label item. For example, use clustered bars to compare stores of different types. Alternatively, cluster bars can be used to compare two different value items, such as Amount of Sales and Units Sold.

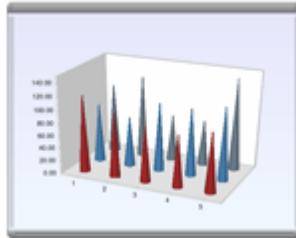
Changing the bars appearance

Besides rectangular bars, Chart FX allows you to change the marker to cylinders, cones and floating cubes. These markers support both 2D and 3D effects.

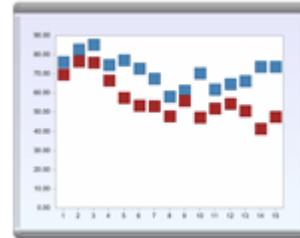
The following figures display some of these special bar chart types:



Cylinders



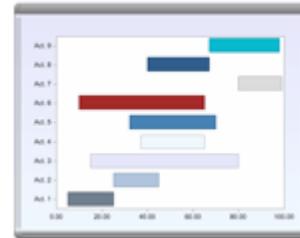
Cones



Floating Cubes

Bar chart with initial values

Most Bar charts display bars commencing at the axis minimum point and extending across the chart area all the way to the value that has been given to the bar. In some cases, bars are required to have an initial value and an ending value, which forces the bar to draw differently. For example, a Gantt chart which allows a graphical display of activities in a project, each activity (bar) has a commencing and an ending date; this forces the bar to be displayed with an initial value as depicted in the figure to your right.



The trick in these chart types is simply to pass an additional initial value to the chart using the **IniValue** property. You can also assign a particular field in your dataset to represent initial values for the chart you are creating. For additional information, please refer to the "Passing Data" chapter in this manual or any of the samples located in the Chart FX Resource Center.

If you are interested in Gantt charts, please check our Gantt extension at <http://www.softwarefx.com/extensions/>.

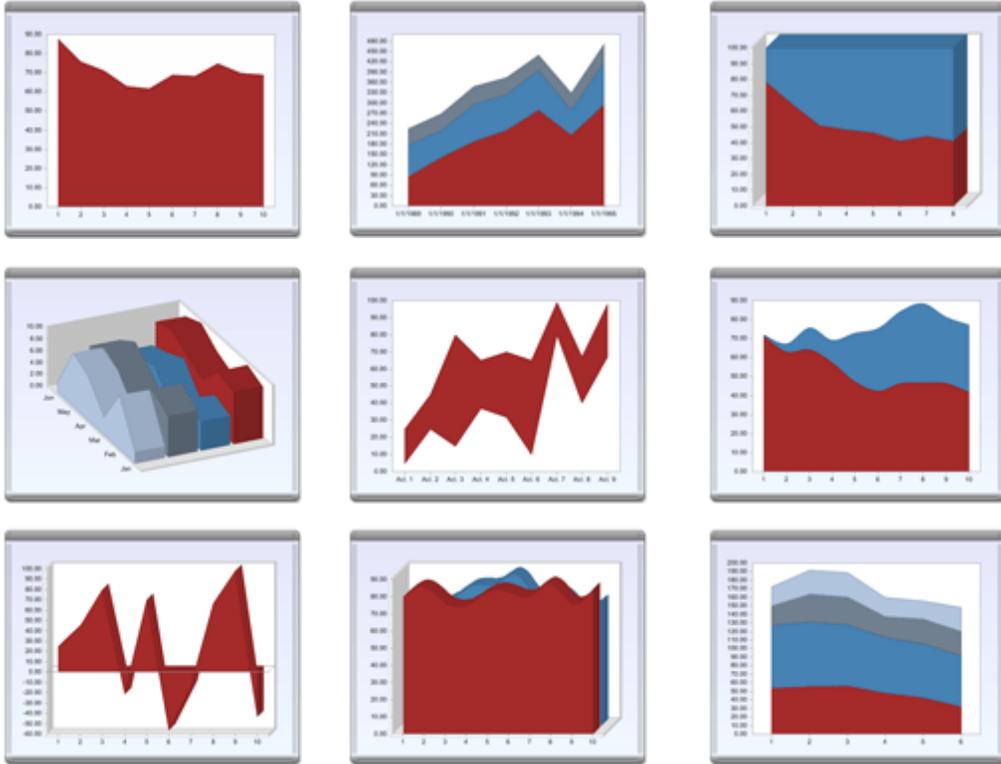
Note: Initial values can be used on both horizontal and vertical bar charts.

Area Charts

Area charts are essentially bar charts with the discontinuous breaks removed along the horizontal axis. Data is not broken into discrete bars but appears in a continuous ebb and flow as defined against the Y axis. Consequently, area charts are particularly useful for emphasizing the magnitude of change over time. In addition, area charts can be used for the same purposes as bar charts.

Because area charts do not break data along the horizontal axis, they are most useful for charting three dimensions of data. The Z-Categories pane should be used to either project data into a third spatial dimension, or to stack two categories of data in a stacked area chart.

Most of the settings introduced and described in the Bar charts section before (Stacked, Clustered, etc) also apply to Area charts. The following table presents different variations of Area charts supported by Chart FX:



Line Charts

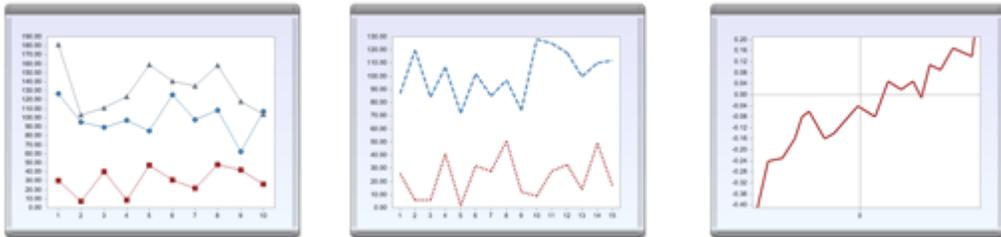
Line charts show trends in data at equal intervals and are effective for comparing highs and lows in a continuum. You can create multiple lines by adding additional data series to the chart. In a 2D line chart additional lines will be drawn in the main chart area and, in the case of 3D line charts, lines could be displayed at the same level or clustered in the Z axis.

Line charts have one advantage over bar charts. They do not allow one set of data to obstruct the representation of another. Since lines are thin compared to bars, the data displayed in the front does not block out the data behind. As a result, data that is not easily represented in bar or area charts work well in line charts. Many more dimensions of data can be superimposed without impairing the chart's effectiveness. Other common uses of line charts include:

- Display long data rows
- Interpolating between data points
- Extrapolating beyond known data values (forecast)
- Comparing different graphs
- Finding and comparing trends (changes over time)
- Recognizing correlations and co-variations between variables
- If the X axis requires an interval scale
- Displaying interactions over two levels on the X axis
- When convention defines meaningful patterns (e.g. a zigzag line)

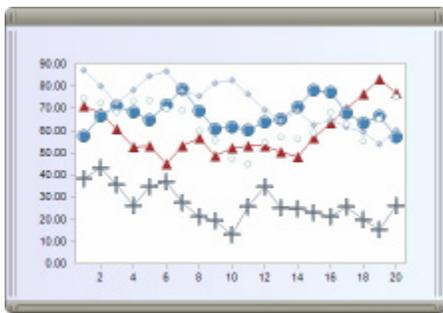
Most line charts are accompanied by a timeline or date based X axis. For additional information on how to pass dates to the chart's x or categorical axis, please refer to the Axes section in this manual.

Passing data to a line chart is processed in the same way as Bar charts. Additionally, Chart FX allows you to control many line settings such as width, color and marker styles. The following table presents different variations of Area charts supported by Chart FX:



Note: Most line charts are accompanied by a timeline or date based X axis. For additional information on how to pass dates to the chart's x or categorical axis, please refer to the "Configuring the chart's axes" section in this manual.

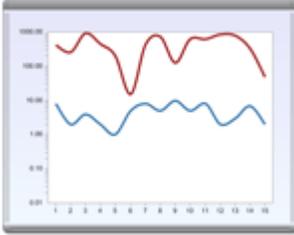
Marker shapes and styles



In most charts, end users recognize different data series by their color. In line charts, however, the color difference is barely visible, especially in printed charts. For this reason, many developers choose to recognize each data set with visible markers in each of the connecting points between the lines.

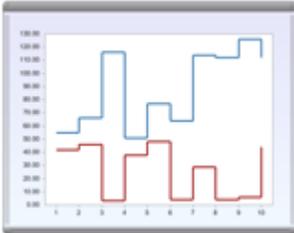
Chart FX provides an extensive API (**PointAttributes** object) that allows developers to customize any setting relating to the line chart markers. Including but not limited to: shape, color, size, label and the frequency of display in the chart. The following chart depicts a line chart with customized chart markers.

Curve charts



Instead of a line, you can choose to use curve charts. When used, curve charts will use a spline algorithm to draw a curve between the points in the chart, rather than connecting the points directly with a straight line. Just as lines, curves also exposed an extensive API to configure the curve color width and markers.

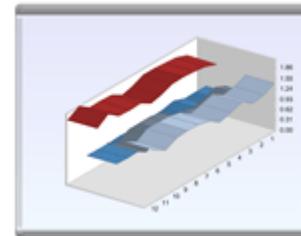
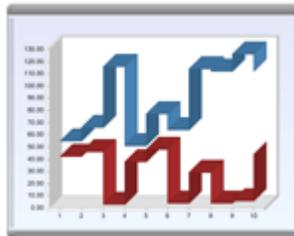
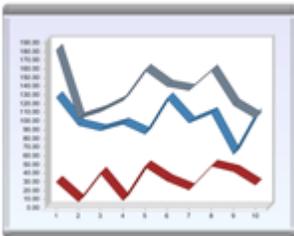
Step lines



Rather than connecting the points with a straight line, these charts display one set of data as steps in a line. These charts are used to illustrate trends between more than two members of nominal or ordinal scales. The following picture depicts a step line chart.

Ribbon (Strip) charts

When a line chart is displayed in 3D, most software refers to these charts as Ribbon or Strip charts. For simplicity purposes, Chart FX refers to these as line charts as well. The 3D flag attribute setting is the only notable difference. The following is an example of some of the most common line charts displayed as a Ribbon chart:



Most of the 3D settings mentioned in the Bar chart section (Perspective, Rotation, etc) also apply to 3D Lines or Ribbon charts.

Note: Lines can also be combined with other chart types, such as bars and area charts, to achieve combination charts. For more information on Combination charts please refer to subsequent pages in this chapter.

Pie Charts

Pie charts are used to show classes or groups of data in proportion to the whole data set. The entire pie represents all the data, while each slice represents a different class or group within the whole. These chart types are used for:

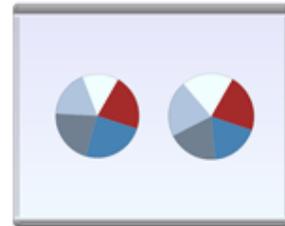
- Conveying approximate proportional relationships (relative amounts) at a point in time
- Comparing part of a whole at a given point in time
- Exploded: emphasizing a small proportion of parts (exploding pie)

Caution! Certain limitations and restrictions exist when using pie charts. For example, Pie charts can not represent values beyond 100%. Also, pie charts should not be used for exact comparisons of values or if proportions vary greatly, because estimating angles is difficult for people. Additionally, Pie charts are only suited for presenting a few percentage values; therefore they are not good to represent large sets of data as a big number of slices in the chart will not be easy to read. Also do not use multiple pies to compare corresponding parts; instead you may use a grouped bar chart to do comparative analysis.

Because pie charts do not display a categorical or x axis, they process data differently than other chart types in Chart FX. Normally, you will want to create a pie chart by passing a single data series and accompanying each value in the data series with a label that could be displayed on each resulting slice.

If more than one data series is passed to the chart, Chart FX will create as many pies as series are in the data set resulting in multiple pie charts in the chart area.

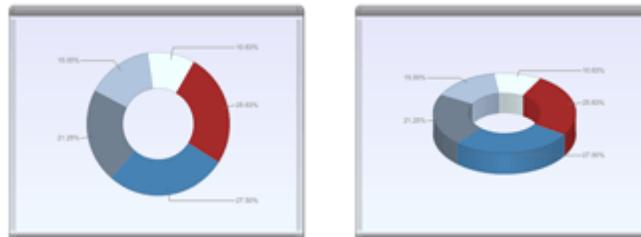
Note: Multiple Pie charts reduce the chart's readability as each pie takes an unusually big space in the chart area. Therefore, we recommend you limit the amount of data series when using pie charts. This is especially true if you have selected a reduced space for your chart or you have decided to display labels in the resulting pie charts.



The chart displayed on the right contains two data series.

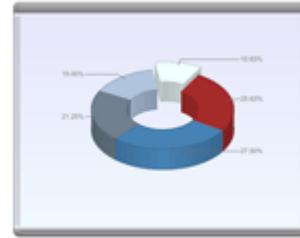
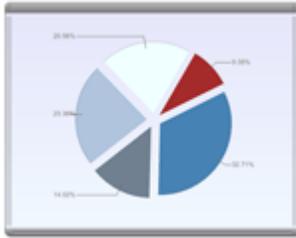
Doughnut charts

Like a pie chart, a doughnut chart shows the relationship of parts to a whole. The following table shows different variations (2D and 3D) of a doughnut chart:



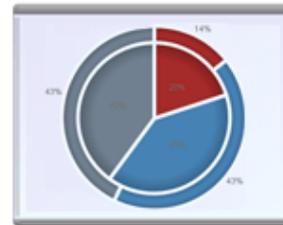
Exploding pie charts

A common practice to highlight certain data points (the highest or the lowest) in a pie or a doughnut chart is to separate one or more slices from the rest of the chart. This effect can be easily achieved using the **SeparateSlice** property available in the **Point** object.



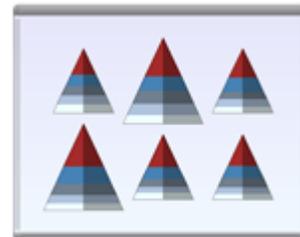
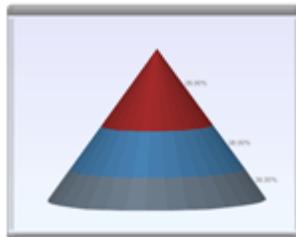
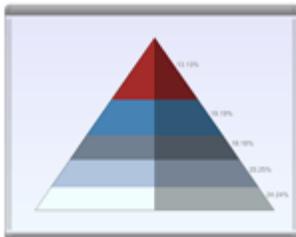
Stacked Pie (Concentric pie charts)

When a pie chart contains more than one data series, you can achieve concentric pies by turning on the **Stacked** property in the chart. This will simply instruct Chart FX to display a single chart with concentric pies, as depicted in the following figure:



Pyramid charts

Pyramid charts are used to show hierarchical lists. You can use them to show business goals or customer requirements. Like a pie chart, a pyramid chart does not display an x axis. Several pyramid charts are presented in the following table:



Note: Since the most important item is usually placed at the top of the pyramid, it is important to mention that the first item in the data array will be displayed at the top of the pyramid. You must make sure you order your data array.

Scatter Plots (XY Charts)

Scatter Plots (also called scatter diagrams or XY charts) are used to investigate the possible relationship between two variables that both relate to the same "event."

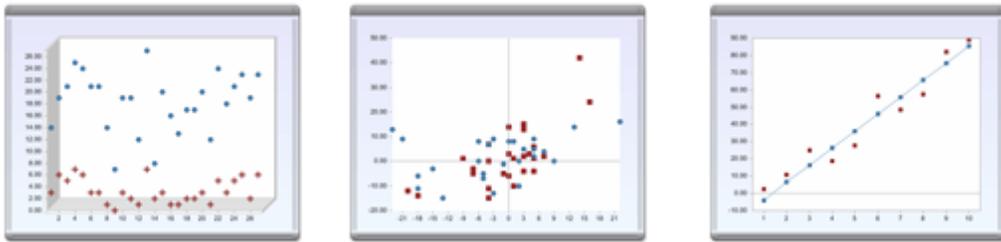
Each point in a XY Plot is defined by two coordinates (X, Y). This results in several important aspects regarding XY charts:

When passing data to a scatter plot you must have two sets of numbers to define a point in the chart. If you are passing data using the API, you must use the Value as well as the XValue properties. If passing data from a database, you must use the Datatype property to indicate which field will represent the X coordinate and which field will represent the Y coordinate of each point in the chart.

XY charts do not have a categorical axis. In other words, both the X and Y axis become a numerical axis and, therefore, you can set any axes values such as minimum, maximum, logbase and other settings related to a numerical axis.

In most cases, scatter or XY Plots are created with point markers with no connecting lines. Nevertheless, Chart FX supports scatter plots (that is, charts which points are defined by two coordinates) in many chart types like Lines, Curves, Area, Step Lines, Surface, Bubble and Contour.

The following table highlights some XY Plots.



Marker shapes and styles

Like line charts, scatter plots use point markers as an important way to differentiate data series in the chart. Scatter plots support all chart marker properties described in the Lines chart section of this manual.

For additional information or samples on scatter or XY plots, please refer to the Chart FX Resource Center.

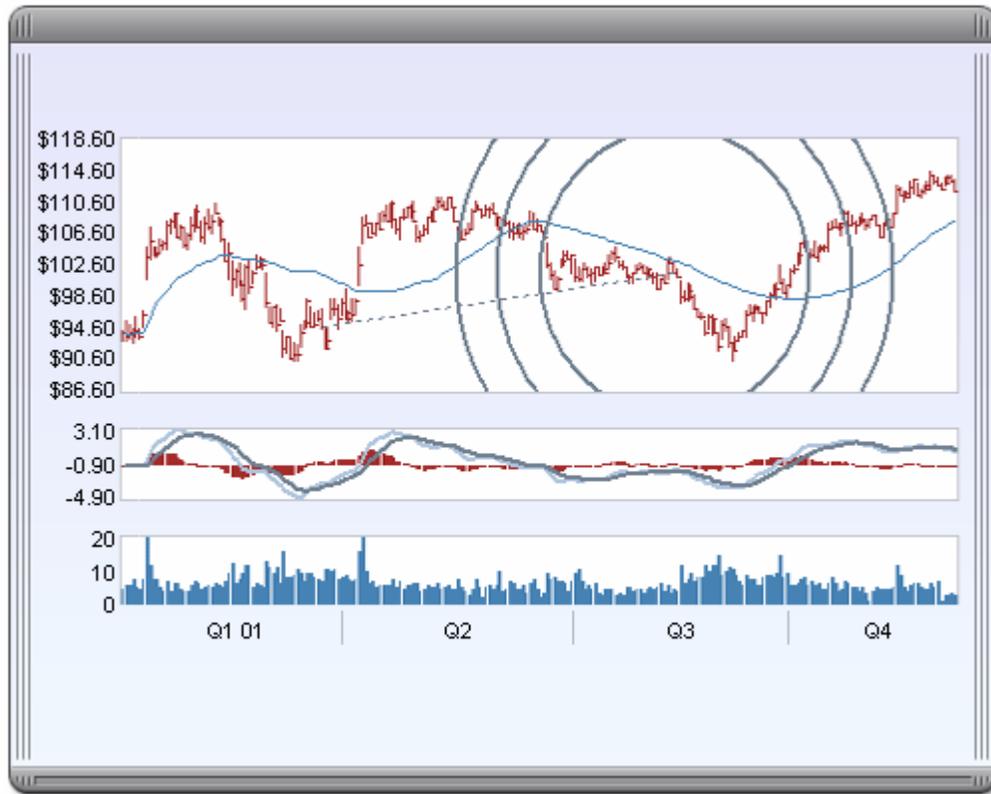
Financial Charts

Compared to traditional bar charts, many traders consider hi-lo-close and candlestick charts more visually appealing and easier to interpret. Financial charts provide an easy-to-decipher picture of price action. Immediately a trader can see and compare the relationship between the open and close as well as the high and low. The relationship between the open and close is considered vital information and forms the essence of these charts.

With Chart FX, you can create Hi-Lo-Close, Open-Hi-Lo-Close and candlestick charts. Passing data to these charts is essentially the same process as passing data to any chart type in Chart FX. You must, however, make sure the series are passed in a certain order so Chart FX can determine which data series represents the Open, Hi, Lo and Close values. For more information on passing data to a financial chart, please refer to the Chart FX Resource Center.

Combination charts and Axis Panes

In Financial charts it is common practice to use several chart types (For example, a candlestick chart with a volume series displayed as a bar chart). Additionally, many end users prefer these charts to be separated in sections or panes. Chart FX supports such features.

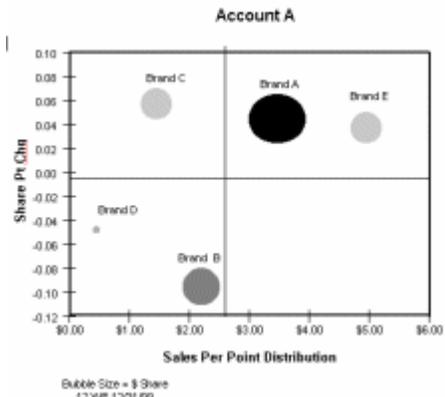


Note: If you are interested in Financial Charts, please check Chart FX Financial which provides not just financial charts but a whole set of Technical Analysis features and stock market analyses.

Bubble Charts

Bubble charts are charts that display circles of different sizes representing a condition. Bubble charts are frequently used in market and product comparison studies

In a way, bubble charts are very similar to scatter charts in that each bubble position is defined by two coordinates. Additionally, the size of the circle at each point reflects an additional dimension of the chart. Because of this, Bubble charts allow three-variable comparisons allowing for easy visualization of complex interdependencies that are not apparent in two-variable charts.



The following table displays some Chart FX bubble charts:

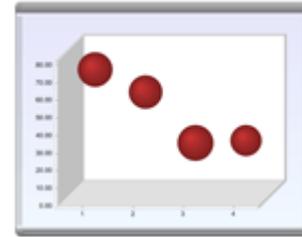
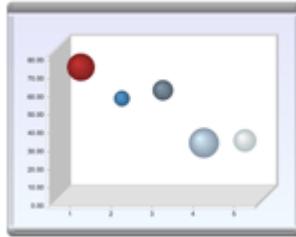
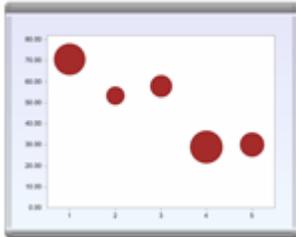


Chart FX uses two series of data to create the most elementary bubble chart. By default, the first series passed to the chart is used for the actual point values (Y Values), while the second series is used to control the size of the bubble marker.

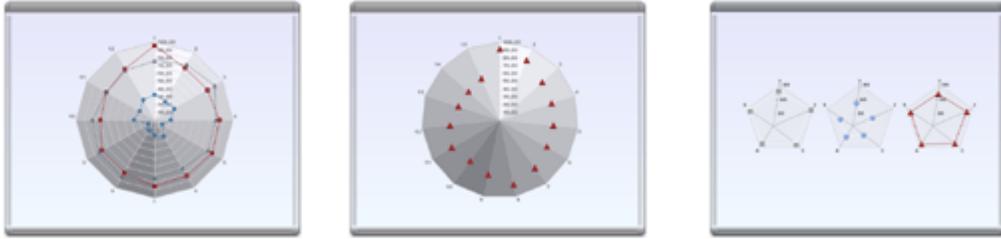
There are several factors that determine the size of the bubble; one being the actual value for the second series, and second being the axis scale which that second series is assigned. This feature actually gives the developer more control over the size of the bubble by providing the possibility of assigning that series to a Secondary Y axis. This scale can be modified independent from the primary Y axis used for the chart values, thus preserving the original display of the axis range. A bubble will reach maximum size when its value is equal to the Y-Axis Max.

Additionally, bubbles can be colored and labeled independently. 2D and 3D bubble charts are also supported.

Radar and Polar Charts

Radar charts are useful when you want to look at several different factors all related to one item. Radar charts have multiple axes along which data can be plotted. In a radar chart, a point close to the center on any axis indicates a low value. A point near the edge is a high value. When interpreting a radar chart, end users will check each axis as well as the overall shape.

The following table highlights a series of radar charts:



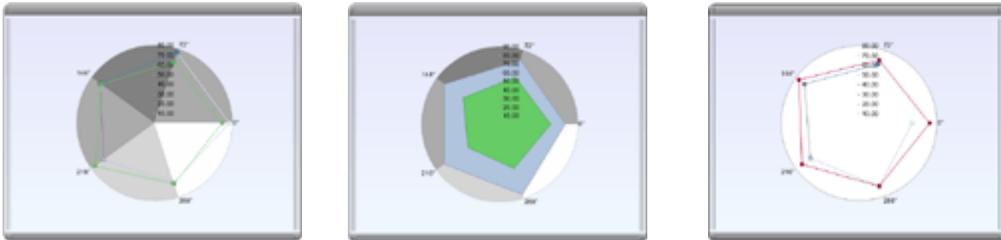
Marker shapes and styles

Like line charts, radar charts use lines and markers as in the chart area. Radar charts also support all chart marker and line properties described in the Lines chart section of this manual.

Polar Charts

Additionally, the Chart FX Polar extension is a complimentary add-on that allows you to plot circular charts based on polar coordinates. The properties of the Polar Object allow you to customize unique visual attributes of the Polar charts. In order to make the members available, you should include a reference in your project and create the "Polar" gallery object. For additional information on how to integrate Polar charts to your applications, please refer to the Chart FX Resource Center.

The following table highlights some Polar charts:



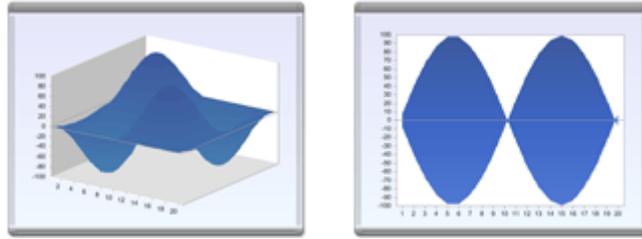
Note: Each point in a polar chart takes two coordinates the angle and the distance from the center. As a consequence, you must make sure you have two data series available when creating polar charts.

You can download Chart FX Polar extension for free at <http://www.softwarefx.com/extensions/>

Surface and Contour Plots

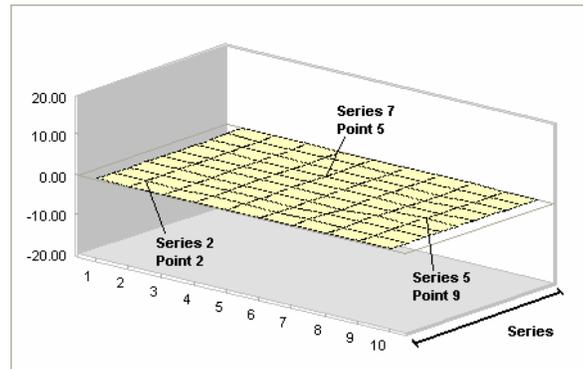
A surface plot is a 3D chart displaying data plotted over a surface area. A contour chart is normally a 2D representation of a surface chart. Surface and contour charts are suitable for comparing two groups of data combinations.

The following table depicts some surface and contour charts:

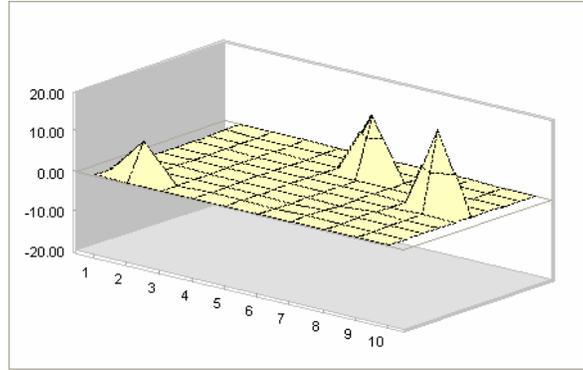


Passing data to a surface chart

In a surface chart, the number of points determines the width of the chart while the number of series determines the depth of the chart. For example, if we want to display a 3D surface chart with 10 Points and 10 Points per series with all values set to zero, it would look like:



As depicted in the figure, you must now locate which data points you want to change in order for the chart to start looking like a surface plot. For example, if we change the value of the points and series highlighted in the figure, the resulting chart will look like the following:



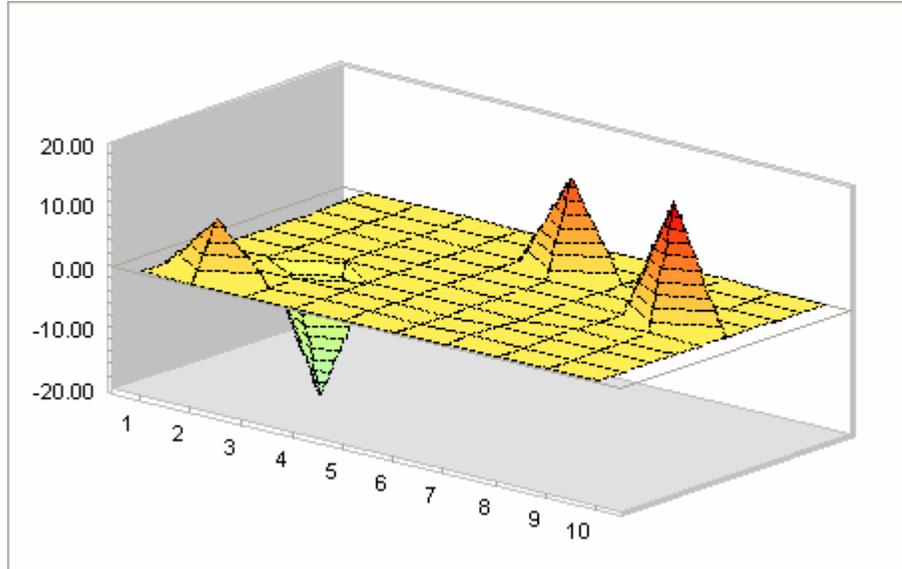
Setting colors to a Surface chart

One of the most important settings in a surface plot is the level of detail that the chart contains. Displaying different colors according to the value of each data point increases the detail of the chart. The color will gradually change from the Color of the first series to its AlternateColor.

Notice although values have been set, the level of detail shown in the chart may not be enough for the scale selected for the Y Axis (-20, 20). We need the surface plot to show the significance of the value using an additional method such as color.

To achieve this, simply change the step of the primary Y Axis to achieve the level of detail you need in the surface plot. For example, in the figure shown above the step has been set to 10, not allowing the surface to display enough variance. However, changing the Step to 2 will allow greater detail to be applied.

An alternative way to do this is by using the axis **MinorStep** property to get a gradual change in color without modifying axis **Step** as shown below:



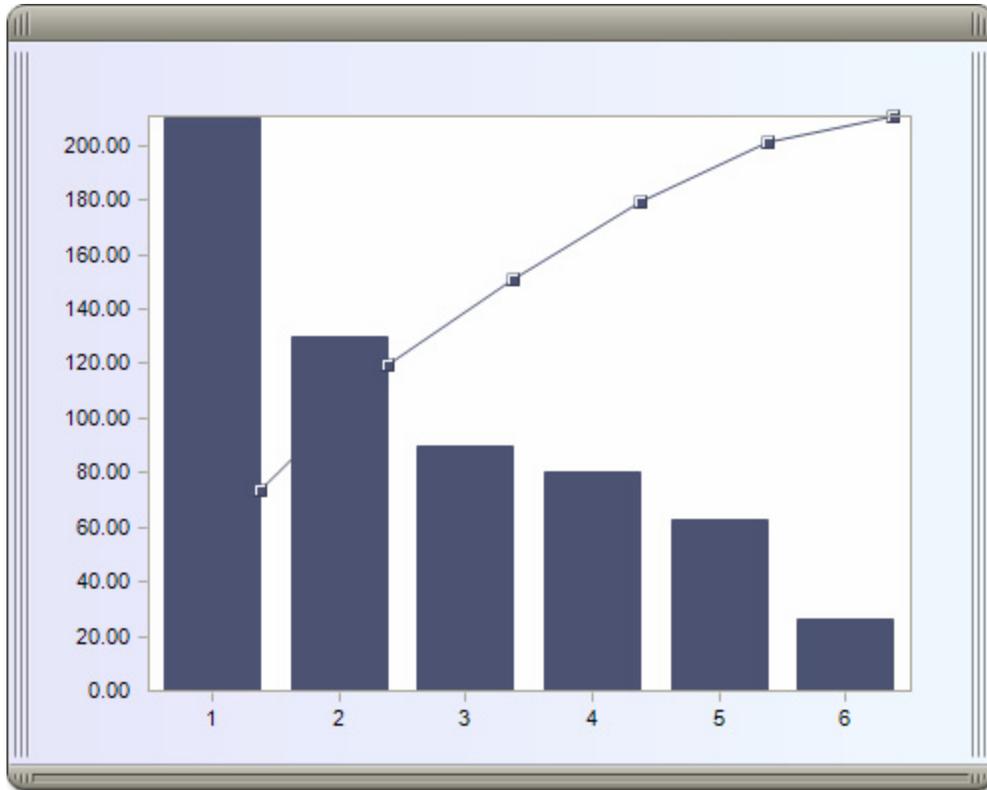
Pareto Charts

Vilfredo Pareto, a turn-of-the-century Italian economist, studied the distributions of wealth in different countries, concluding that a fairly consistent minority – about 20% – of people controlled the large majority – about 80% – of a society's wealth. This same distribution has been observed in other areas and has been termed the Pareto effect.

The Pareto effect even operates in quality improvement: 80% of problems usually stem from 20% of the causes. Pareto charts are used to display the Pareto principle in action, arranging data so the few vital factors that are causing most of the problems reveal themselves. Concentrating improvement efforts on these few will have a greater impact and be more cost-effective than undirected efforts.

Pareto charts display a bar chart with values in the primary Y axis and a line chart automatically calculated and displayed by Chart FX representing the cumulative percentage at each of the bars. Normally, the values are sorted so the 80% - 20% effect could be easily deciphered.

The following picture depicts a Pareto chart:



Note: Chart FX DOES NOT sort the data for you. Therefore, you must sort the data before passing it to a Pareto chart.

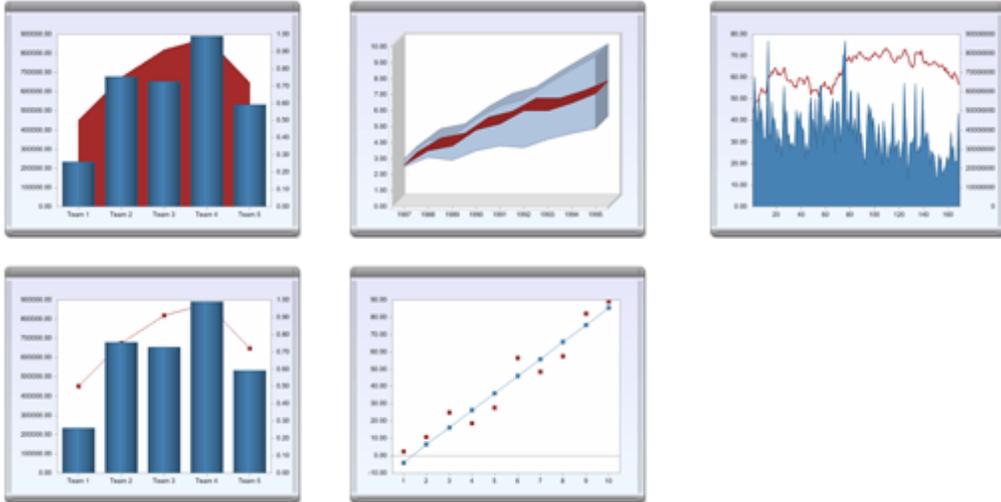
Combination Charts

Combination charts combine some of the strengths of all chart types available in Chart FX. For example, Solid bars can be used for the most important data against which other dimensions are represented in lines. In this way, emphasis is given to a portion of data based on its importance. A combination chart is especially useful for comparing two numeric values, such as amount and units of sales.

To create a combination chart, you must have a multiple series chart and then set the gallery type for each individual series in the chart using the Series.Gallery property. This will set each gallery independently. The following table presents some of the most common combination charts.

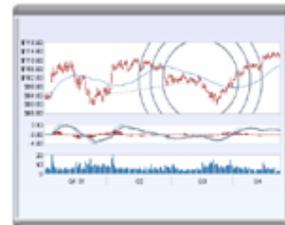
Chart FX supports Combination charts in 2D as well as 3D modes.

The following table presents some samples of combination charts:



Axis panes and Secondary axes.

When data series are not related, or when the scale of values differ greatly, you may want to use Axis Panes or Additional numerical axes in the chart. These techniques will allow you to enhance the chart's readability. For example, in a financial chart. The price section has a much lower scale than the volume (which ranges in volume). In this case, you may want to use a secondary Y axis that ranges in the millions and assign the volume series to such axis. In this particular case, it is also common practice to separate both charts in sections or panes as depicted in the picture on your right.



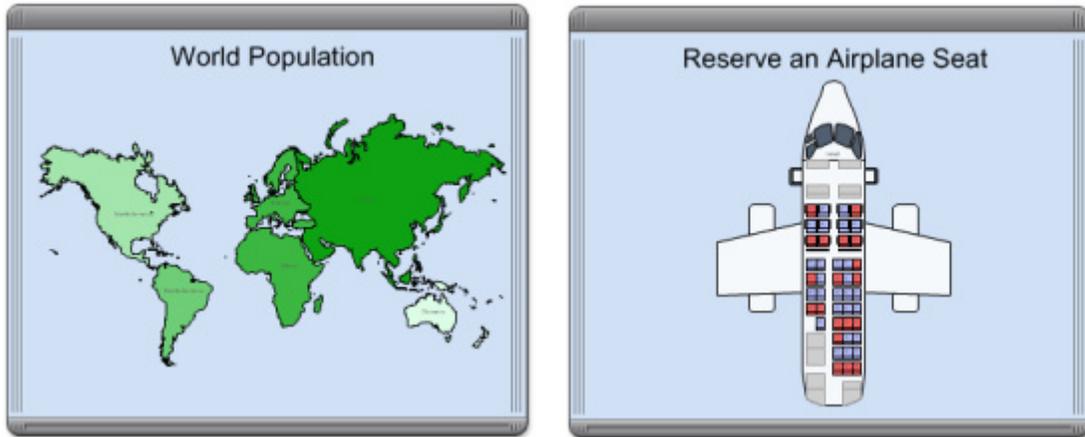
Note: Not all chart types can be combined.

Maps

Sometimes the usual bar, pie and line charts are not enough to represent your numerical data. Chart FX Maps allows you to extend the powerful charting capabilities of Chart FX by adding vectorized images that can represent your numerical data in other context of scenarios. For example, considering representing sales data by territory, wouldn't it be better to use a color-coded map rather than bar chart? Imagine this map is also interactive and allows your end users to drill down the map to more detailed territories like counties or zip codes.

The Chart FX Maps extension provides a full library of dynamic maps that you can integrate with Chart FX to display any type of data graphically through the objects in a map. Additionally, Chart FX Maps not only offers a comprehensive library of maps ready to integrate into your application, but it also allows the developer to add any number of maps through the popular Scalable Vector Graphics format (SVG) from Adobe®. Simply pass or connect your data, define a few conditional attributes and Chart FX will immediately reflect these conditions as colors in the map.

The following is just a small sample of what you can do with Chart FX Maps:



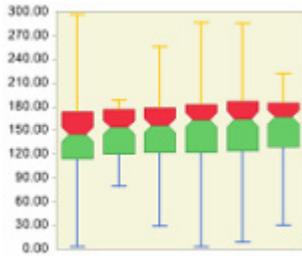
Note: Despite Chart FX having the ability to overlay elements on top of maps, Chart FX Maps should not be used for GIS or positioning purposes.

Chart FX Maps is sold separately. For additional information on Chart FX Maps, please visit our web site at <http://www.softwarefx.com/extensions/>

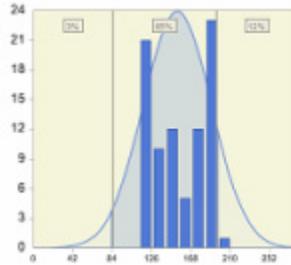
Statistical Charts

The Chart FX Statistical extension allows you to quickly apply statistical analysis without the task of incorporating complex mathematical algorithms in your code. Simply configure the extension to compute the desired calculations and Chart FX Statistical will deliver the desired chart, statistical index and studies.

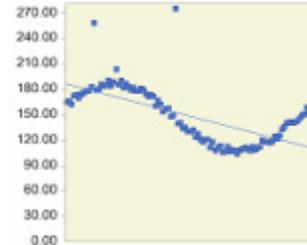
Chart FX Statistical offers a wide variety of new chart types, such as histograms, box whisker and Statistical Process Control charts (a.k.a. Six Sigma charts). The following table presents some of the charts you can achieve with Chart FX Statistical.



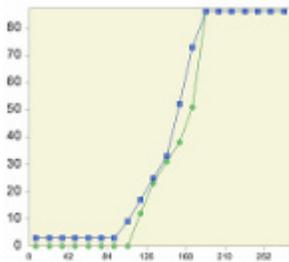
Box whiskers



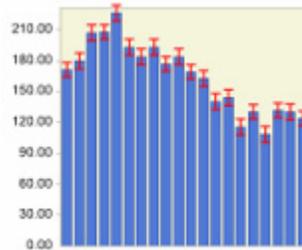
Histogram



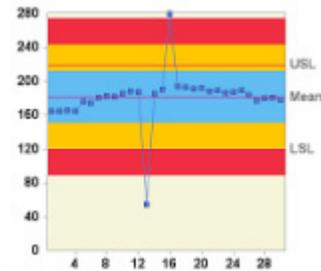
Linear regressions



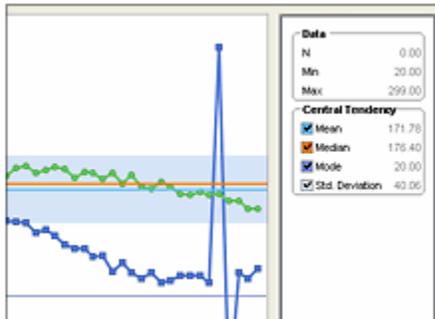
Cumulative frequency (Ogive)



Bar chart with error bars



Six Sigma or SPC



The Chart FX Statistical User Interface

Additionally, Chart FX Statistical provides end users with a customizable statistical legend. This legend is used to display analysis studies and calculations that are performed on sample data. It also serves an interface for end users to display interactive analysis studies added to the chart. Interactive studies may be added and removed from the chart display by selecting the appropriate check boxes.

Chart FX Statistical is sold separately. For additional information on Chart FX Statistical, please visit our web site at <http://www.softwarefx.com/extensions/>

Design Time Experience



Overview

Here at Chart FX, we make every effort possible to make sure your experience is productive, intuitive, enjoyable, and most importantly simplified. As a result, we put great effort in assisting you during the design time phase of your projects.

This chapter outlines features such as Design Time Wizard, integrated help, and customized environment settings among others.

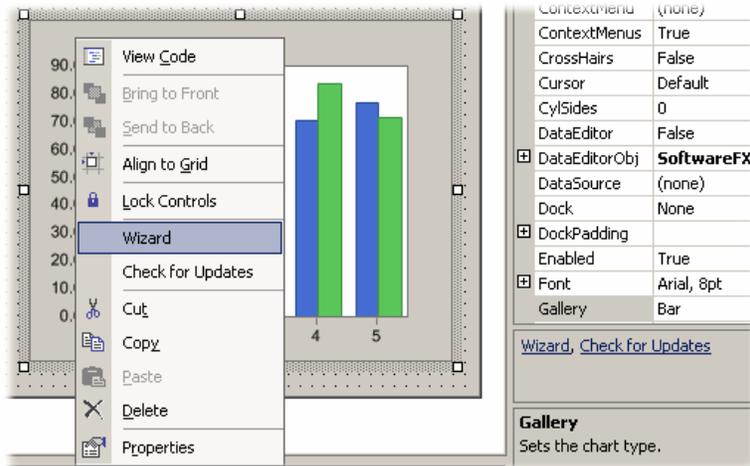
The Chart FX for .NET Design Time Wizard

As an added feature for developers, Chart FX for .NET provides a design time wizard that allows you to easily customize general settings for a myriad of charts with minimal programming effort. The Wizard walks you through a series of steps (in the form of dialogs) to complete the task of building a chart. This helps you familiarize yourself with properties used to customize particular chart elements and create sophisticated charts without the need of referencing the Chart FX API or Resource Center help pages. With this in mind, the Wizard was designed to provide support for the most commonly used Chart FX properties.

Accessing the Wizard

You may access the Chart FX for .NET Wizard in a few different ways:

When you drop a chart control from the toolbox on a form, the Wizard automatically appears. You may open the wizard manually by right clicking the chart and choosing Wizard at design time.



Setting properties using the Wizard

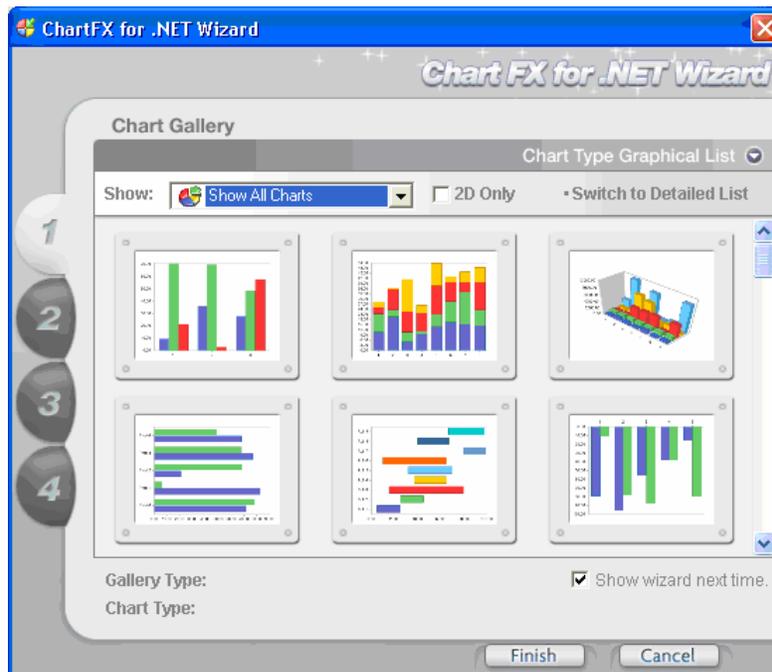
Creating a chart using the Wizard is easy. The first screen displayed will be the Chart Gallery configuration dialog. From there you will continue to set the chart's color scheme, visual elements and other miscellaneous properties. Located at the bottom of the Wizard are the navigation controls that consist of forward, back, finish and cancel commands. The navigation controls are available on all Wizard dialog pages.

Step 1 - Chart Gallery Dialog

The Chart Gallery Dialog allows you to select from a variety of chart types including Bars/Columns, Area, Lines, Bubble, Pie/Doughnut/Pyramid, Scatter, Financial, Surface, Contour and Combination charts. By selecting a particular category from the gallery combo box, the available charts will be displayed in the main frame. You can then select a chart by using the mouse cursor.

There are also options available to filter 2D charts as well as to modify the display style for the available gallery types. You can select between graphical or detailed list styles.

Located in the bottom right corner of the Chart Gallery Dialog is the "Show wizard next time" checkbox. This checkbox can be used to disable the Wizard. Once you have selected the chart type you can click next to continue.



Note: Please note that the Wizard charts have a set number of points and series. When you pass your data to a selected chart, the appearance may vary based on the actual values.

Step 2 - Color Scheme Dialog

The Color Scheme Dialog allows you to select the palette to be applied to the chart. From time to time Chart FX introduces new palettes; so check the latest service pack for new palette updates.

Step 3 - Visual Elements Dialog

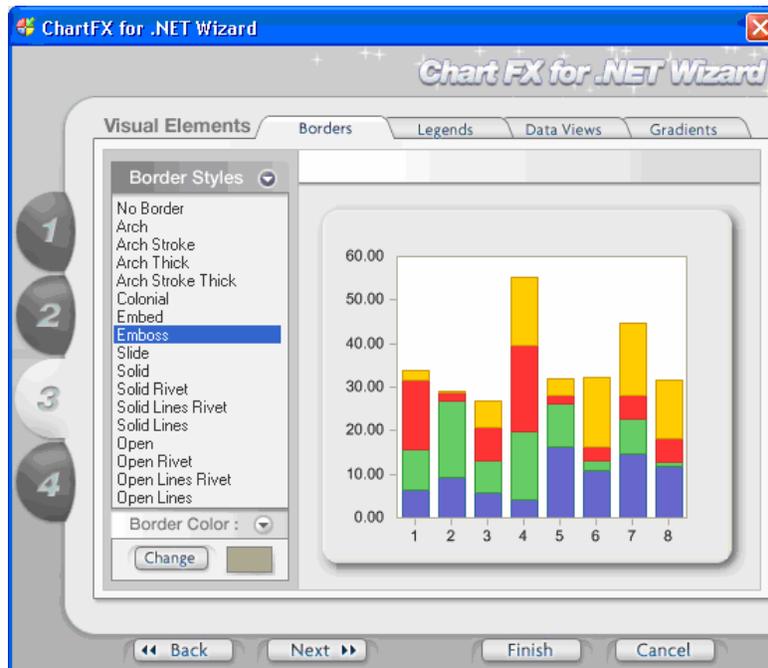
The Visual Elements Dialog provides a means of including Chart FX Tools as well as cosmetic attributes to charts using the Wizard. Located near the top of the dialog are a series of tabs marked Borders, Legends, Data Views and Gradients. By selecting the appropriate tab, you can assign attributes specific to that chart element.

The **Borders** tab allows you to assign an ImageBorder to your chart. You can also select the border color.

The **Legends** tab allows you to add the series legend box to the chart and control the positioning, alignment and border style attributes.

The **Data Views** tab allows you to add and customize the data editor. Using the supported controls, you can assign position, alignment, border style, interlaced grid, and header attributes.

The **Gradients** tab allows you assign a gradient to the background or series markers in a chart. From the Wizard you can select the gradient type and beginning and ending gradient colors.



Step 4 - Miscellaneous Dialog

The Miscellaneous Dialog gives access to title, X Axis, Y Axis and point label properties. Near the top, a unique set of tabs is available for customizing the various options. The tabs have the following functions:

- **Chart Title:** allows you to set text for the title, configure the alignment of the text, change font and control the dock area of the title relevant to the chart.
- **X Axis:** allows you to show or hide the axis, assign a title, set a grid, and control label formatting (Angle or Staggered).
- **Y Axis:** allows you to show or hide the axis, assign a title, set a grid, and control label formatting.
- **Point Labels:** allows you to show or hide point labels, control alignment and set label angle.



Note: Please note that none of the properties or settings available via the design time wizard is data-related.

Finishing the Wizard

At any point during the creation of a chart using the Wizard, you may select to finish applying the current chart to your form. If you do not want the settings configured in the Wizard to be applied to the chart, you can simply select "Cancel" and return to the previous design time development environment.

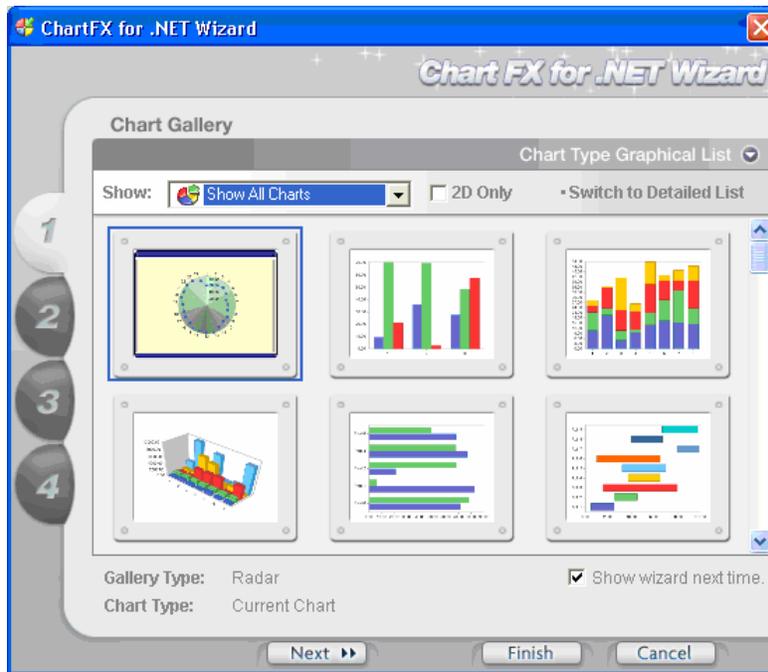
Disabling the Wizard

When developers have reached the point where the Wizard is no longer needed to assist in creating charts, they can disable the Wizard from automatically running when a chart control is dropped to a form. To disable the Wizard, simply uncheck the "Show wizard next time" checkbox located at the bottom right corner of the Chart Gallery wizard screen. If you need to access the Wizard again, you will need to use the right click menu or link described in the "Accessing the Wizard" section.

Re-entry Capabilities

Re-entry capability refers to the idea that the Wizard remembers the previously configured chart's properties. For example, you create a Radar chart using the Wizard and select finish to apply the selected chart to your form. Once the chart has been applied to the form, you decide that a title should be added to the top of the chart. When you access the Wizard again, your chart will be the first chart available in the Wizard.

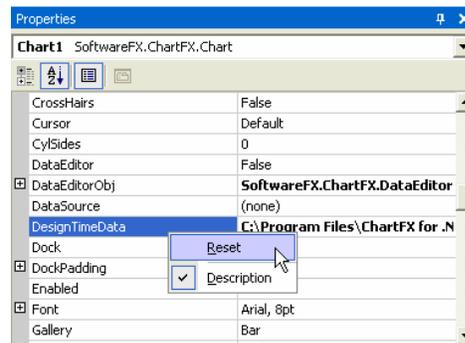
The Wizard can also be used after you have started assigning properties using the properties dialog window. Let's say you drop a chart and configure a palette and title. When you open the Wizard, those settings will already be applied to the first available chart in the collection. Below is a screenshot of the Wizard after a radar chart has been created and the Wizard has been reopened.



Design-Time Data

Many of the charts that can be created from the Wizard use random data to populate the chart; however, there are cases when random data will not allow a chart to represent what it was designed to. In these cases, Chart FX allows data to be passed through the use of a property called DesignTimeData. As with random data, charts designed with fixed data will continue to use the fixed data until any data related method (i.e. OpenData or CloseData) is called within the application. Calling any data related method instructs Chart FX to disregard the fixed data set with the DesignTimeData property and use the data passed by the developer. If a data related method is called but no data is passed, the chart will be populated with random data that may cause undesirable results.

The Chart object supports two design time properties; NSeries and NValues, which control the number of series and points a chart will contain. Normally, when you set these properties at design time, Chart FX calculates the values for those points randomly. However, if you have the DesignTimeData property set, you may not adjust the NSeries and NValues properties because the data determines the number of series and points. You can reset the DesignTimeData by right clicking the property label in the properties dialog and selecting 'Reset' or by removing the configured path from the setting. This will then allow access to the NSeries and NValues properties.



Visual Studio Properties List

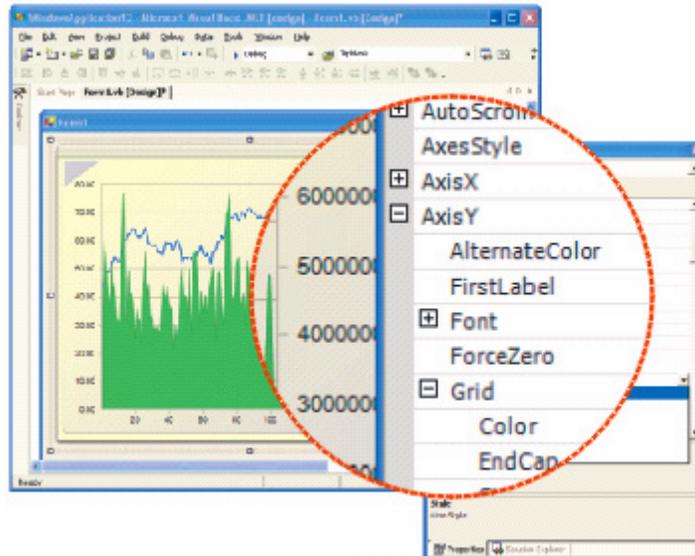
Most IDEs, and more especially Visual Studio, provide a Properties list that is common to all components embedded in a form (buttons, list boxes, charts, etc). This provides an intuitive way of manipulating objects allowing you to quickly view supported API members while boosting productivity and reducing the learning curve. In some cases, the properties list provides the specific property settings and enumerations preventing unnecessary compilation errors.

In Chart FX, a wide range of properties are provided allowing you to tailor the look, feel and behavior of the charts. You can set both visual and data dependant properties using the Properties list. Most of the Chart FX properties available in the Properties list are used for atomic operations (which are simple operations that must be performed entirely or not performed at all).

Note: If you set properties that are data-dependant at design time, at run time, these settings will still apply and may overwrite an automatic Chart FX behavior. For example, Chart FX calculates the Axis min and max automatically according to the data source. If, at design time, you set the Y Axis maximum to 100 using the Properties list, at run time the chart will still display that value regardless of the maximum in the dataset.

Complex Properties and Objects

In order to avoid a large number of properties at the first level in the Properties list and to offer a more intuitive and pleasant design-time experience, Chart FX uses the complex property paradigm in the Properties list. Please note, complex properties can extend beyond a second level. For example, the following picture depicts the AxisY property which supports Grids that offer several properties like Colors and Lines Styles:



All properties available in the Properties list can be accessed via code and Intellisense. Nevertheless, the Properties list DOES NOT provide access to the entire Chart FX API.

For additional information on the entire Chart FX API please refer to the online documentation or the Chart FX Resource Center.

The Chart FX for Java Designer

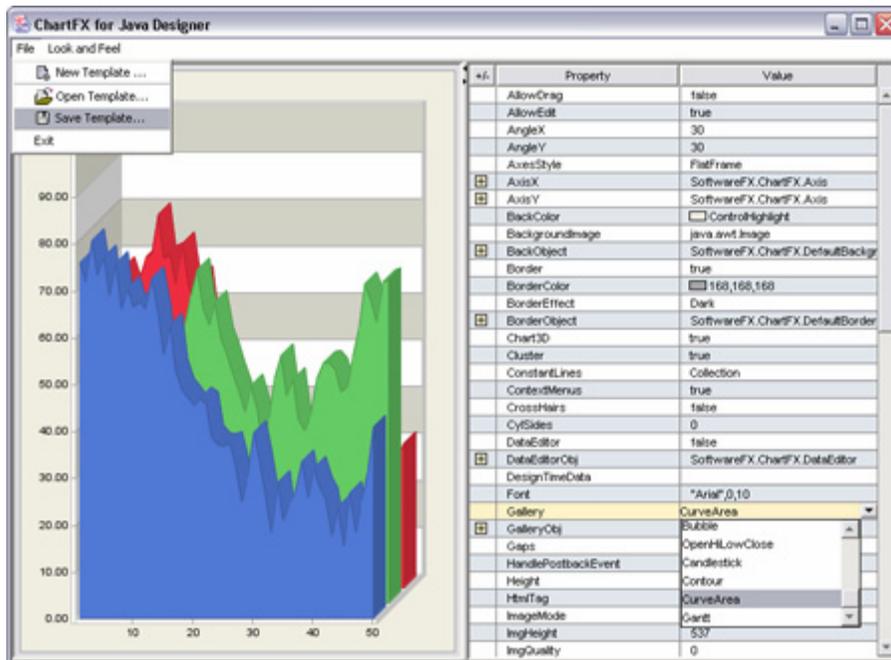
The Chart FX for Java Designer is a stand-alone chart template generator. Using this tool, you can create XML files containing all the visual attributes you desire for your chart. Once you have configured these settings in a familiar design-time environment, you can export the settings as XML and later import them into your charts using the import method.

Installation

The Chart FX for Java designer requires the Java Runtime Engine; however, depending on the Edition (Standard or Enterprise) you may be required to download additional components:

Both **J2EE 1.3 & 1.4** and **J2SE 1.4** include the **Java API for XML Processing (JAXP)**, which the designer is dependent on. Therefore, if you are running one of these environments, you simply need to run the designer Executable Jar File (**chartfx.designer.jar**) located in the Designer directory of the Chart FX for Java installation.

If you are running **J2SE 1.3**, you will need to download and install **Java API for XML Processing (JAXP)**. This may require Downloading the **Java Web Services Developer Pack 1.x** from <http://java.sun.com>.



Running the Designer

1. Systems that support executing .jar files by double clicking on the file may do so to run the Chart FX for Java designer.
2. For users who prefer to start the designer from a command prompt:

Navigate to the Designer directory (containing **chartfx.designer.jar**, **chartfx.jar** and **cfxdesigner.lic**). Execute the following command:

```
C:\Designer> java -jar chartfx.designer.jar
```

Using the Designer

Once you have applied the desired attributes to the chart using the properties dialog window, click the "File" option located in the designer's menu and select "Save Template". This will save an XML file containing the visual attributes applied in the designer. Note that per series attributes as well as data related settings are not saved in the XML. For example NSeries and NValues, which control the number of series and points in the chart, would not be recorded.

Importing the XML

Now that you have saved an XML template with the desired visual attributes, it may be imported into any pre-populated Chart FX chart to produce the desired look and feel with one line of code. To import an XML file, the following code could be used:

```
chart1.importChart(FileFormat.XML, application.getRealPath("/ChartFX")  
+ "/data/chartsettings.xml");
```

Once the data has been passed to the chart and the XML file has been imported, you will see the visual attributes applied to achieve the chart as follows:

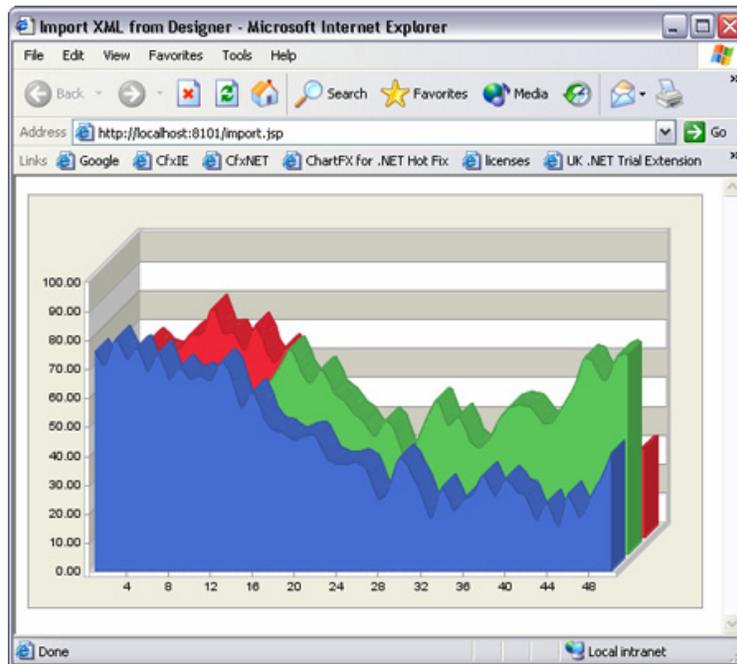
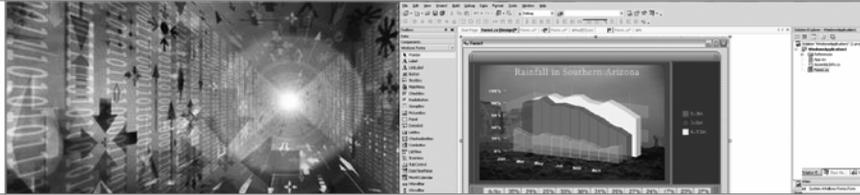


Chart FX for Java Designer Licensing Considerations

The Chart FX for Java Designer can be distributed to as many computers as needed. To distribute the designer to new machines, run the Chart FX for Java installer again and select the Designer option, or simply copy the three files, chartfx.jar, chartfx.designer.jar and cfxdesigner.lic to the machine and run the .jar file as described above.

Please note that this license is only valid for the designer and that calls to the API will still need to be directed to the machine with the properly licensed server component. If you wish to make calls to machines where you have distributed the designer, you will need to obtain new licenses for additional test/development server installations.

Passing Data to Chart FX



Overview

The most basic method of passing data to a chart using Chart FX is to utilize the built-in API. The Chart FX API provides properties and methods to load and manipulate data into a chart. Passing data to your charts using the API may be the most basic method, however, one of the features of Chart FX is the ability to read or pass data from many sources. These sources include databases, text files, XML, arrays, collections, and many others.

Getting Started

Passing data from databases and text files is supported by Chart FX's core objects; however, depending on the platform, all the code required to read data from additional data sources such as XML, array and collection sources is isolated in an additional library (ChartFX.Data.DLL) allowing Software FX to easily build additional "Data Providers" as the need arises without having to modify the core objects.

By providing an additional library exclusively used for data, the programmer can decide when to include or exclude the library, thus giving control of what is to be deployed. Although data functionality requires an additional library to be deployed as part of your application, it allows easy integration of new data sources, i.e. Active Directory, Grids, OLAP, etc.

In Chart FX, there is no need to loop through data points to populate the charts. If you have a previously filled data array, just pass it to Chart FX and have the chart filled with data with as little as one line of code.

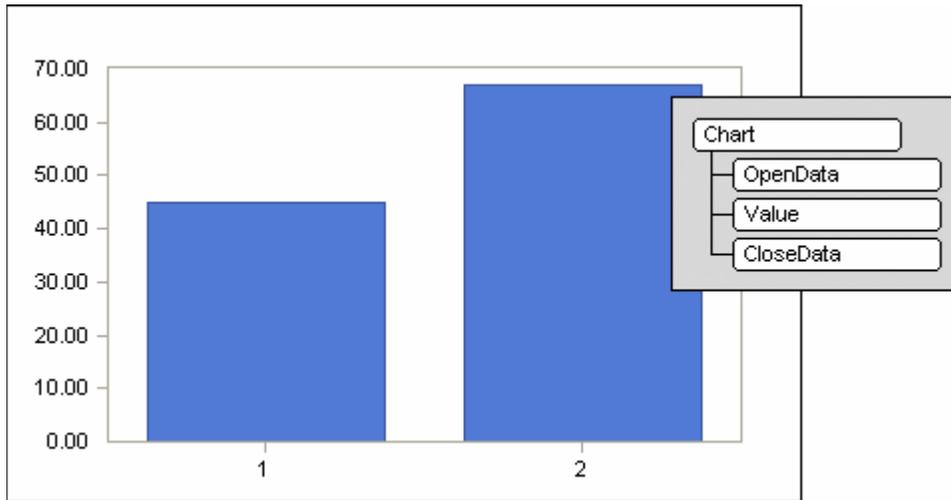
Additionally, Chart FX supports data binding with data controls and data readers. If you drop an ADO.NET DataSet control on your form, you can set a property in Chart FX allowing the chart to take information from the data control directly. Or, if you prefer, load the data to the chart by coding in an event handler using a DataSet or even a Data Command.

This chapter will explain how to populate your charts from the most common data sources. These sources include: API, Database, Text, XML, Arrays and Collections.

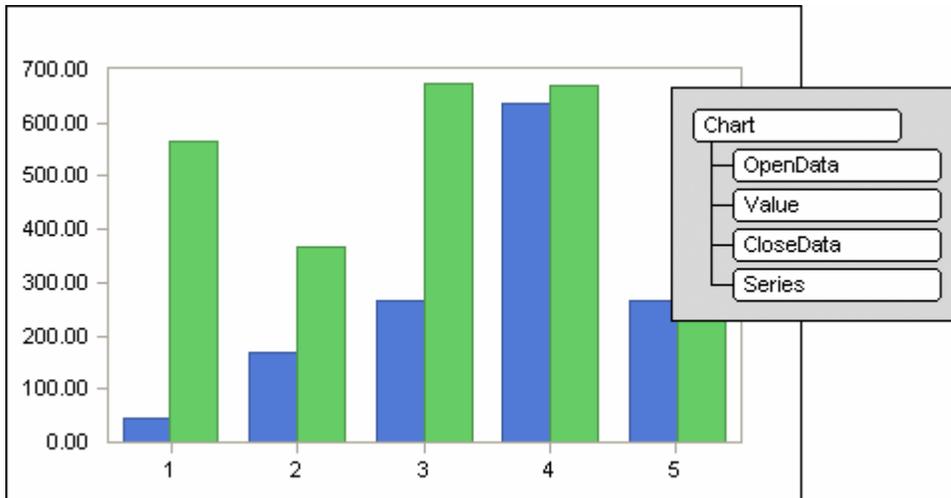
Series and Points

In order to render a chart, Chart FX must have the data organized in a meaningful way.

Providing data for simple charts is as simple as having a simple set of values to plot. For example, a bar chart representing the total yes/no votes of a poll is a single series with two values as shown below.



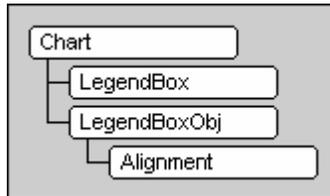
As charts display more complex data, the organization of the data must be structured and organized in a meaningful way – not only for the rendering of the data, but also for importation purposes. For this, Chart FX applies the notion of Series in conjunction with points. By introducing series, Chart FX is able to capture the necessary data to produce many additional charts taking advantage of multidimensional data. For example, we may have a chart to display relative products sales before and after a marketing push. In this scenario, you may want to setup a chart with 2 series and N points where N is the number of products affected by the marketing push. Additional, you may want to modify the properties of the additional series to distinguish it from the first such as color, etc. To do this you would simply reference the Series property indexer of the Chart object in order to manipulate a respective series as appropriate.



Setting Legends

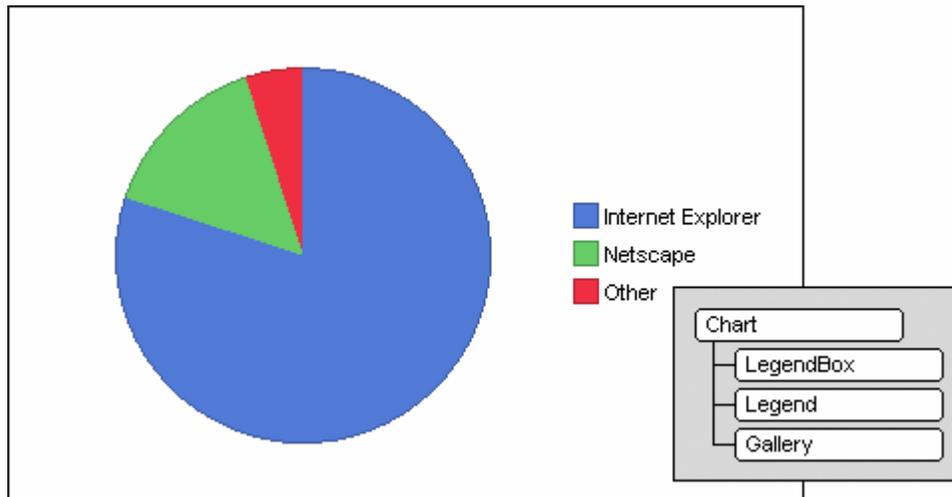
A legend is a visual guide to show the viewer additional information regarding the values being displayed without convoluting the chart itself with additional overlaid data. Additionally, legends are particularly useful to improve chart readability for charts like Pie, which have no axes but point labels are still needed. Depending on the chart, legends typically are associated with the series or the values of the chart. Additionally, there may be instances where neither a Series-based legend nor a Point-based legend applies. In such scenarios, there is also the option for a user defined legend.

The LegendBox for values is always shown docked to a margin (right margin by default). Additionally, the Alignment property can be used to align the LegendBox within the margin. The Chart object LegendBoxObj property is used to set position and style attributes directly to the legend box.



Value Legend

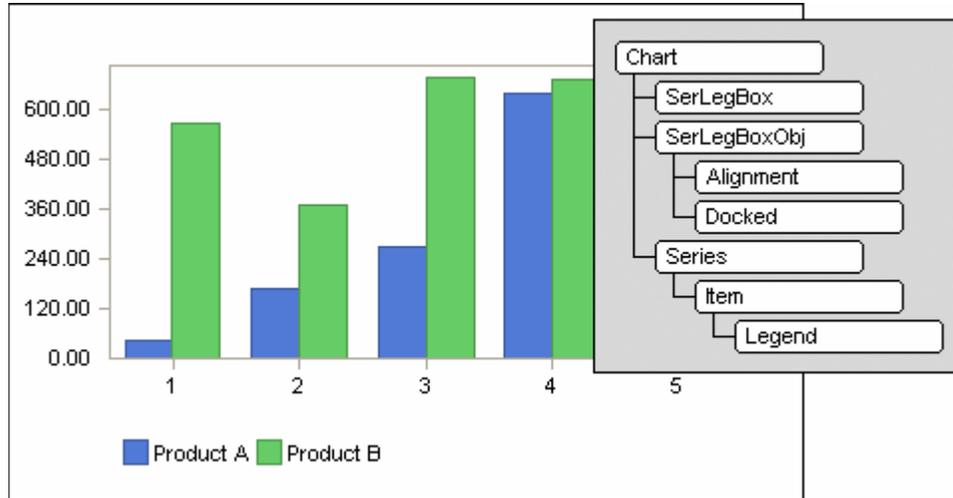
The Chart's LegendBox property is used to display legends associated with data points. In order to give meaning to the points, you need to use the Legend property to associate a string for each value point. For example, the following Pie chart is created with the following API calls:



Series Legend

In a multiple series chart, it is standard practice to add a legend box containing descriptions of the series displayed. Series legend box keys may be set using the Chart object SerLeg property or the Legend property of the SeriesAttributes object. In addition, you may set the position and style attributes using the SerLegBoxObj property.

Using the SerLegBoxObj property you can control the position and other important aspects of this tool in the chart. For example, the following API can be used to show and position the series legend on the bottom and to the left of the chart:



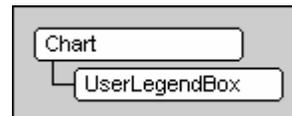
User Defined Legend

In most cases, the legend box and series legend box provide the necessary information users need to successfully understand a chart. Other times, there is a void where the legend box and series legend box cannot provide this to the end user. The user legend box allows you, as the developer, to create legends to provide essential information about a chart that normally cannot be relayed using the series legend box or the value legend box.

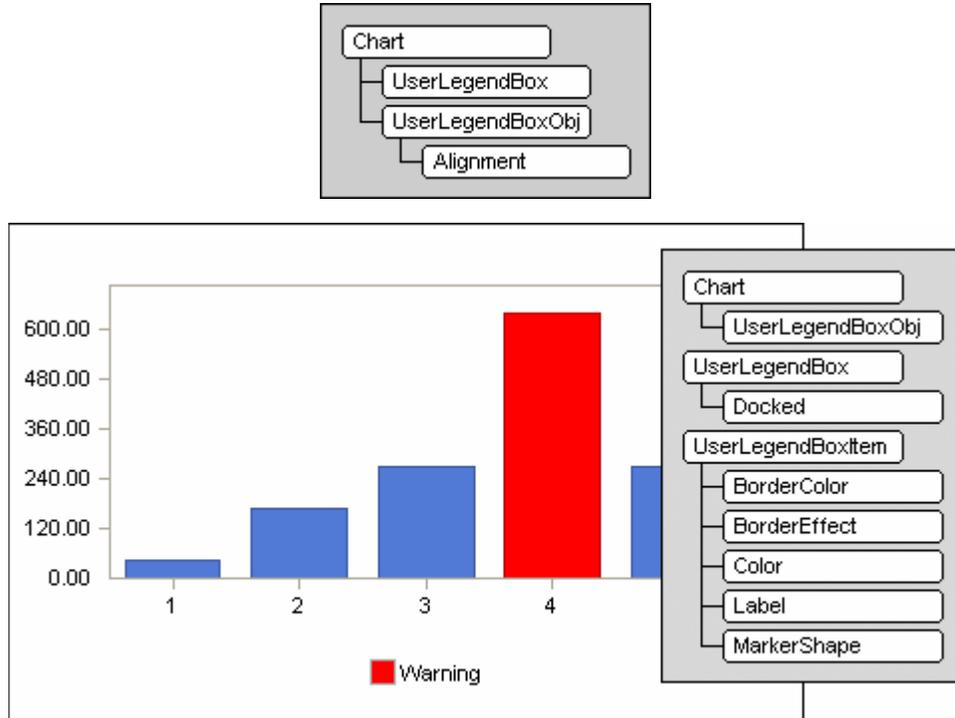
A user legend box is very useful in many situations, one of which is when the number of colors in a chart is not related to the number of series in a chart. For example, when a Contour gallery type chart is used, colors in the chart represent intensity of a particular condition. By enabling the user legend box, you can define legend keys to help an end user decipher exactly what the chart is representing.

Another situation when the user legend could be very useful is when you assign per point attributes to your chart. For example, all red points may represent values over 50, where all blue points represent values under 50. A user legend box could be used to make the user understand this relation more clearly. User legends are also a great way to define or explain abbreviations or symbols included in financial charting applications.

The user legend box is enabled in the same way as both the series legend box and legend box objects. The **UserLegendBox** property is a Boolean property allowing you to show or hide the user legend box with one line of code.



Similar to the Series and Value legend boxes, the user defined legend box allows alignment along with the various other common attributes available for legend boxes.



Passing Data Using the API

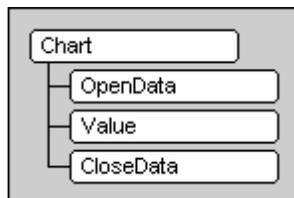
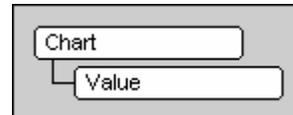
Passing numeric values using the Chart FX API is as simple as opening a communication channel using the **OpenData** method, setting the values and then closing the communication channel using the **CloseData** method. You can pass data using the API through any loop command (while, for, etc.) supported in your development environment.

Passing Data using the Value property

Sometimes the easiest or most convenient way to pass numerical data to Chart FX is to use the **Value** property. The **Value** property allows you to set the numerical value for a point in a particular series in the chart.

The Value property takes the index of the series and the index of the point you want to populate – both zero based indexes.

For example, if you want to set a value of 40 for point 3 of series 2, you must use the **Value** property from the chart object passing both the series index of 1 and point index of 2 to the value property, and then assign the value 40.



The Value property must be used after calling the **OpenData** and before calling the **CloseData** methods. For example, after calling **OpenData**, you can loop through the points and use the Value property in order to set the points for the entire chart as the following API shows:

Unknown Number of Points

The **OpenData** method requires you to specify the number of points expected to be available for each series. There may be times when this number is unknown at compile time. Sometimes, knowing the number of points the chart will hold is not possible.

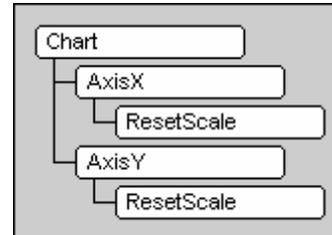
For example, if you're retrieving values from a database according to end users requests, it will be impractical to fetch all records and count them just to know the value Chart FX needs to open the communications channel.

The OpenData method supports a constant called **Unknown**, which will force Chart FX to allocate memory dynamically as you pass numeric values to it. To use the Unknown constant, you must specify it as the third parameter to the **OpenData** method.

Note: Unknown is valid only for an unknown number of points. The number of series in the chart must be preset when calling the OpenData method.

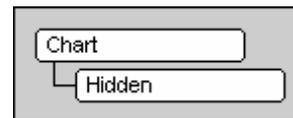
Axes Rescaling

When data is loaded to the chart, by default, Chart FX automatically adjusts the axes **Min** and **Max** to guarantee every value is inside the range. This behavior is controlled by the axes **AutoScale** property. However, if a new set of values is loaded on the same chart, Chart FX will adjust the Min and Max again, only increasing when necessary but never decreasing them, which can affect the chart visualization. To force the Min and Max to be recalculated based on the new set of data, call the axes **ResetScale** method before loading the new data.

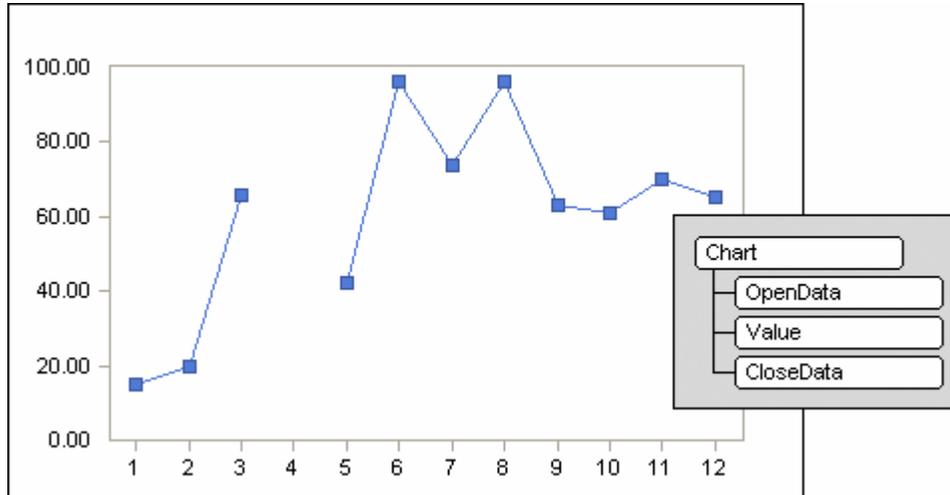


Hidden Points

Chart FX has the ability to create hidden points in the chart. Although Chart FX forces the same amount of points per series, you may specify hidden points to create the illusion of some invisible points in the chart. Assigning the **Hidden** Chart constant:



All you need to do to create a hidden point is to assign the Chart object Hidden constant as a value to a particular point.

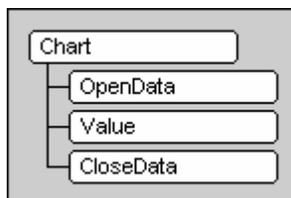


Note: Remember to use the OpenData and CloseData Methods when setting values to the chart.

Changing Existing Values in the Chart

If you've already populated the chart and all you want to do is change an isolated value; you may do so without repopulating all the data again. The `OpenData` method supports a setting called **Unchange**. `Unchange` instructs Chart FX you intend to change data without having to change the size of the data points.

Although changing points outside of `OpenData()` / `CloseData()` may seem to work, there are dependencies such as the automatic resizing of the X or Y axis depend on the channel of data being open. As an example, the API calls necessary to make such a change are as follows:



Passing Data Using Databases

In most cases, you will want to populate charts with data coming from a database table or query. Chart FX supports data binding through various methods through platform specific data access.

.NET Framework

For the .NET Framework, we make use of both the disconnected model through the use of DataSets as well as the connected model through the use of SqlDataReader and related classes. For additional information on how to connect to databases using the Chart FX for .NET refer to the appendix of this manual or visit our online documentation.

Java

For Java, we make use of the classes available through java.sql package such as Connection, ResultSet, Statement, etc. For additional information on how to connect to databases using the Chart FX for Java, visit the appendix of this manual or visit our online documentation.

COM

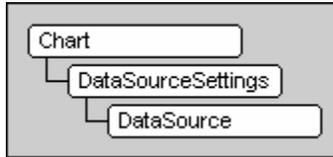
For COM, we make use of the ODBC and OleDb objects. For additional information on how to connect to databases using the Chart FX for COM visit the appendix of this manual or our online documentation.

Note: More detailed online documentation is available at: <http://support.softwarefx.com>

Passing Data Using XML

XML has proven to be a very dynamic and flexible means of data storage. Chart FX has been developed with this in mind and allows you to populate your charts with data directly from XML files.

The Load method is supported by the XmlDataProvider object. To use this feature, you must first create a new XmlDataProvider object, load the XML data either through one of the various Load methods, and finally set it to the DataSource property :



When passing the data using the XML data provider, the data is passed as columns and rows, the same as with a table. The following demonstrates the format of a typical XML used to import and export data to and from Chart FX.

```
<?xml version="1.0"?>
<CHARTFX>
  <COLUMNS>
    <COLUMN NAME="Product" TYPE="String"/>
    <COLUMN NAME="Q1" TYPE="Integer" DESCRIPTION="1ST QTR"/>
    <COLUMN NAME="Q2" TYPE="Integer"/>
    <COLUMN NAME="Q3" TYPE="Integer"/>
    <COLUMN NAME="Q4" TYPE="Integer"/>
  </COLUMNS>
  <ROW Product="ChartFX 98" Q1="9200" Q2="7835" Q3="10245" Q4="8762"/>
  <ROW Product="ChartFX IE 3.5" Q1="14350" Q2="11233" Q3="16754" Q4="987"/>
  <ROW Product="ReportFX" Q1="12398" Q2="7654" Q3="5678" Q4="9087"/>
  <ROW Product="Image Toppings" Q1="8742" Q2="12358" Q3="14321" Q4="8702"/>
</CHARTFX>
```

Here, the definition for each column is enclosed in a COLUMNS node. The columns can optionally contain a DESCRIPTION attribute used to label the legend of the chart. If the attribute is omitted as in Q2 through Q4 above, the value of the NAME attribute is used for the legend.

Configuring Properties via XML

Chart FX makes extensive use of XML. Chart FX not only supports XML for importing data, but XML can also be used to manipulate visual attributes and formatting for your charts. The XML property files can be written to or read from an XML file by using the Import and Export methods.



A formal schema has not yet been defined. Each property however is an element, and properties for most sub objects (series, point, axis, etc.) are represented as sub elements. When customizing series or point attributes, the values are represented as XML properties within the SERIES element instead of as sub elements as shown in the XML file below.

```
<CHARTFX>
  <VOLUME>35</VOLUME>
  <TITLES>
    <TITLE><TEXT>My First Chart</TEXT></TITLE>
  </TITLES>
  <SERLEGBOXOBJ>
    <DOCKED>Left</DOCKED>
    <DRAWINGAREA>true</DRAWINGAREA>
    <FONT><NAME>Verdana</NAME></FONT>
    <ALIGNMENT>Center</ALIGNMENT>
  </SERLEGBOXOBJ>
  <SERLEGBOX>True</SERLEGBOX>
  <SERLEG>
    <LEGEND>Sales</LEGEND>
    <LEGEND>Projected</LEGEND>
  </SERLEG>
  <SERIES>
    <SERIES index="0">
      <BORDER>True</BORDER>
      <GALLERY>LINES</GALLERY>
    </SERIES>
    <SERIES index="1">
      <BORDER>True</BORDER>
      <GALLERY>Lines</GALLERY>
    </SERIES>
  </SERIES>
  <POINT>
    <POINT series="1" point="2">
      <COLOR>Red</COLOR>
      <MARKERSHAPE>Rect</MARKERSHAPE>
    </POINT>
  </POINT>
  <STRIPES>
    <STRIPE>
      <AXIS>0</AXIS>
      <TO>20</TO>
      <FROM>30</FROM>
    </STRIPE>
  </STRIPES>
</CHARTFX>
```

Crosstab Data Provider

Many of the charts require data to be properly formatted for Chart FX to generate the desired chart. With data typically coming from data sources such as databases, the data is usually formatted in tabular format as shown in this table.

ProductName	Quarter	QuarterSales
Chart FX 98	1	\$52,300.00
Chart FX IE 3.5	1	\$41,400.00
Chart FX IE 2000	1	\$37,600.00
ReportFX	1	\$49,130.00
WebBarFX	1	\$75,000.00
Chart FX 98	2	\$36,220.00
Chart FX IE 3.5	2	\$21,400.00
Chart FX IE 2000	2	\$75,600.00
WebTreeFX	2	\$17,300.00
...		

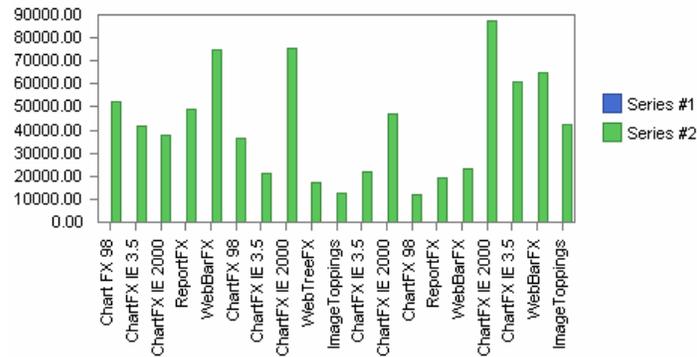
Normally, developers would have to spend countless hours manipulate this format manually to generate the desired format. Chart FX solves this problem by providing the Crosstab data provider. With the Crosstab data provider, we can easily convert the above data to the following format for displaying within Chart FX.

ProductName	Quarter 1	Quarter 2
Chart FX 98	\$52,300.00	\$36,220.00
Chart FX IE 3.5	\$41,400.00	\$21,400.00
Chart FX IE 2000	\$37,600.00	\$75,600.00
ReportFX	\$49,130.00	
WebBarFX	\$75,000.00	
WebTreeFX		\$17,300.00
...		

When you wish to utilize the Crosstab Data Provider in your Chart FX applications, depending on your platform, you may need to add the related data library as a reference in your project. This library is typically located within your Chart FX for .NET installation.

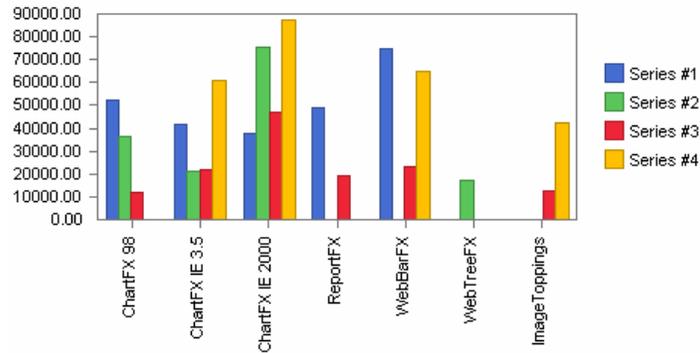
By including this reference as part of your project, you can create Crosstab objects as well as other data providers implemented by the data library. Refer to later topics in this section for more information on how to use the data library object with different data sources.

The tabular formatted table when passed to Chart FX will generate the following chart:



In most cases, this is not the chart you prefer to generate. The ProductName field information is repeated for each row and in order to compare the quarter sales for various products in different quarters, you must match numbers physically distant from one another on the page. Additionally, it could be easily overlooked there are some quarters with no sales for a specific product.

Using the Crosstab data provider, the original table can easily be reorganized to produce the revised table. When read in by Chart FX, the revised table will display the following chart – displaying the data more effectively.



The Crosstab Data Provider Functionality

There are two important aspects to effectively use the Crosstab Data Provider:

- How the Crosstab Data Provider integrates with Chart FX.
- How to configure Crosstab to convert tabular data to columnar data.

Taking advantage of the extensibility of Chart FX for .NET's design, the Crosstab Data Provider is a separate component, which can be used with Chart FX as a data provider. The Crosstab Data Provider's input must be in the form of a supported Chart FX data provider, such as an array, collection, text file, XML or database among others.

Configuring the Crosstab Data Provider to Convert Tabular Data to Columnar Data

Due to the multitude of possible scenarios, the Crosstab Data Provider needs to have rules for the conversion of tabular to columnar data. These rules define how each column in the tabular format is used. A column may be defined as any of the following options using the **Data Type** property: Column Heading, Row Heading, Value, IniValue or Not Used.

Column Heading: When data in a column of the tabular data is used as a column heading, each unique item appearing in a column will form a column in the resulting columnar format. There can only be one Column Heading defined; if more than one column heading is defined the first one defined will be used.

Row Heading: When a column or set of columns of the tabular data is used as a Row Heading, each unique item appears in the columns specified will form a row in the resulting columnar format. The number of row headings can be defined is unlimited.

If more than one column is specified as a row heading, the Crosstab Data Provider will concatenate the values into a single string in left-to-right order. The **Separator** property can be used to define a string separating each part of the row heading. By default, there is no separator.

IniValue: When a column of the tabular data is defined as IniValue, the Crosstab Data Provider will use its content as the initial value for data points to achieve floating bars and area charts.

Value: When a column of the tabular data is defined as Value, the Crosstab Data Provider will use its content as the value of the element in the matrix, at the intersection of a specific row and column. Only one value column can be defined; if more than one value is defined, the first one defined will be used.

Not Used: When a column of the tabular format is defined as Not Used, the Crosstab Data Provider will ignore the column. By default, all the columns are not used.

For further explanation please visit the online documentation.

Note: The RowHeading records passed to the CrosstabDataProvider need to be ordered consecutively. You must ensure the RowHeading values are consecutive, otherwise the chart will not understand which values to group together, displaying undesirable results.

Other Crosstab Data Provider Properties

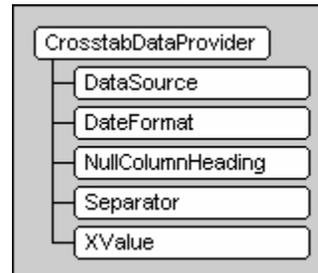
DataSource: The DataSource property is used to instruct the Crosstab provider to retrieve the information from an external data provider. The configured external data provider must be configured to a valid data provider such as a TextProvider, XmlDataProvider, ListProvider, etc.

DateFormat: The DateFormat property is used to instruct Chart FX how to format the dates when a column specified as a Row Heading is a date. If no DateFormat is specified, the format will be set from the Culture property. By default, the Culture property is set to en-US.

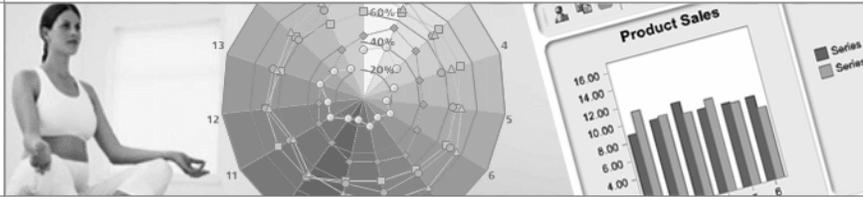
NullColumnHeading: The NullColumnHeading property is used to instruct Chart FX to use the configured string value when a null column heading is encountered.

Separator: The Separator property sets the separator between two or more fields which may form the point label when more than one Row Heading is configured as a DataType.

XValue: The XValue property instructs Chart FX to use a single row heading as XValues instead of X Axis labels. Setting this property to True will instruct Chart FX to plot the row heading values as Xvalues instead of X Axis labels. This does not apply when you have multiple row headings configured.



Visual Attributes

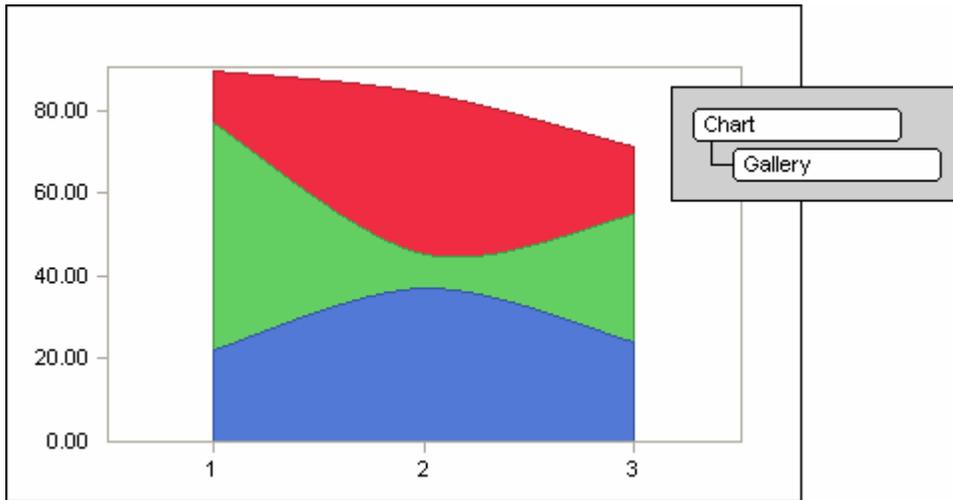


Overview

With over 20 different chart types, customizable legends, ready-to-use color palettes, multiple and customizable axes, annotation objects, per-marker attributes, gridlines, background images and border objects, Chart FX allows you to create some of the most visually appealing charts in just minutes. In addition, Chart FX provides advanced visual features, including anti-aliasing, smoothness, alpha-transparencies, gradients, and many more.

General Attributes

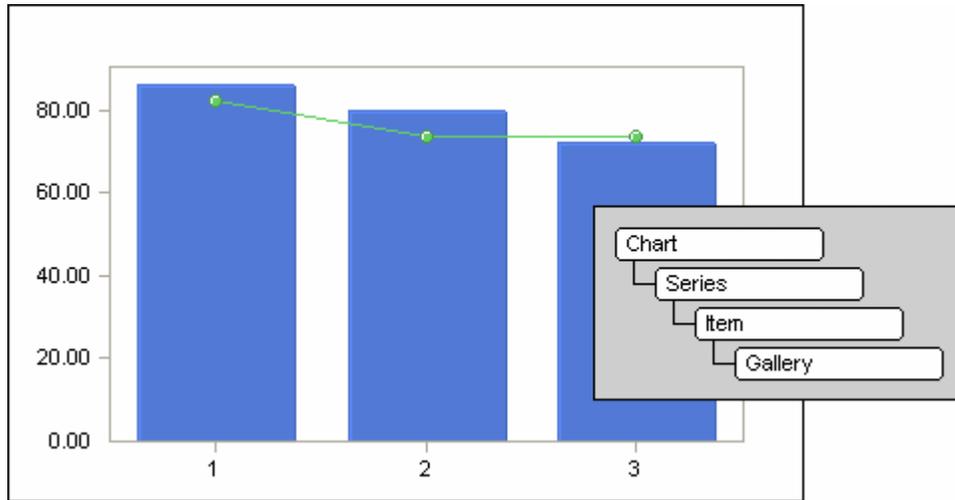
General visual attributes are exposed by the Chart object and when configured affect the chart as a whole. An example of setting a general visual attribute is setting the chart's **Gallery**. Setting the gallery for the chart using the Chart object Gallery property will set a chart type globally; in other words, only one chart type will be displayed at a time (Bar, Lines, etc.).



Some other supported members of the Chart class include the Border, BorderColor, BorderEffect, CylSides, MarkerShape, MarkerSize, PointLabels, Scheme, TypeMask, Volume, MarkerStep, etc. These are just a few of the general properties exposed by the Chart object, for more supported members please reference the Chart FX API Reference.

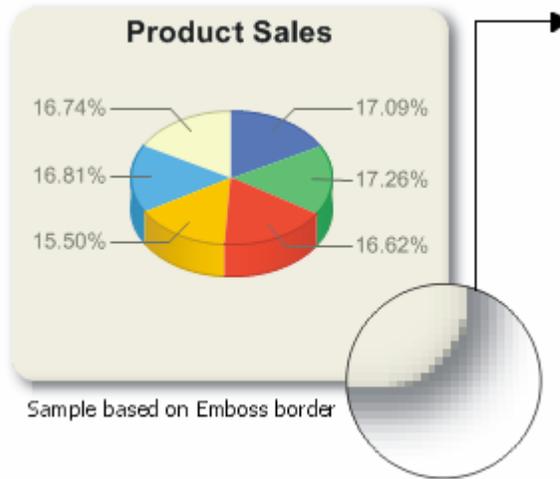
Per-Series Attributes

The SeriesAttributes object exposes many of the same properties described as General attributes, however, when configured only affect the selected series in the chart. By allowing customization on a per-series basis, Chart FX provides the developer with more control over the supported chart elements. Setting different chart galleries for selected series in a chart permits you to create combination charts to convey more information to the end users at one time. The Series object also allows you to set visual attributes to a series that can bring attention to a selected series.



Borders

Of all the features built into Chart FX geared to enhance the chart's presentation layer, none stand out more than Image Borders.



Alpha blending capability is used to display Chart FX image borders. These borders consist of a series of bitmaps that have transparent or semi-transparent pixels. In addition to a red, green, and blue color channel, each pixel in a border bitmap has a transparency component known as its alpha channel.

The alpha channel lets you control the color of the chart background as well as the color of the HTML page or form while keeping the effect of the border intact; resulting in an outstanding presentation effect for your charts without the restrictions imposed by static bitmaps that can't be changed programmatically.

Image Borders are very simple to integrate and require little or no programming efforts on your part. Also, image borders are supported in a separate library. This allows you to decide if you want to use and deploy them as part of your application or server.

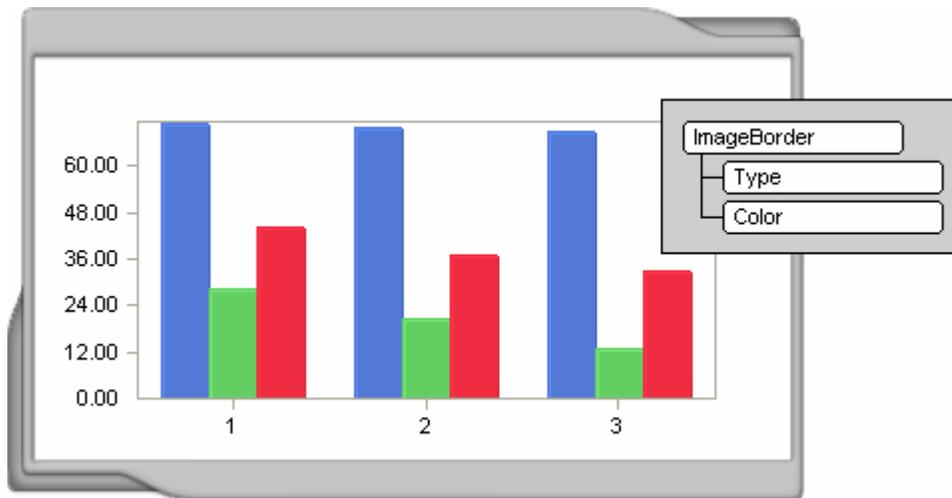
Borders and Background Color

The Chart FX **ImageBorders** use shading and great edge detail to give a multidimensional effect when they are placed in your forms. When using image borders, you want to seamlessly integrate them to your page or form design. If you have set the background color of your form to a color other than white, an undesirable white block will be visible around the chart.



The **PageColor** property can be used to modify the background color of the shaded area around image borders. Setting the PageColor property to match the background color of the chart container will achieve a seamless visual effect. The Border Object exposes some properties to customize the appearance of the border; for example, a property Color can be used to change the border color.

The following figure shows a border object "Pulsar" assigned to the chart, with its Color property set to "Silver":



Note: You can also integrate and display image borders, when available, from the Chart FX Wizard, the Chart FX Designer and Properties List (if supported, through the BorderObject property). The advantage by adding these borders from these tools is that you can quickly select among the different supported borders visually, instead of adding the code, compiling and running the application.

Borders and other chart tools

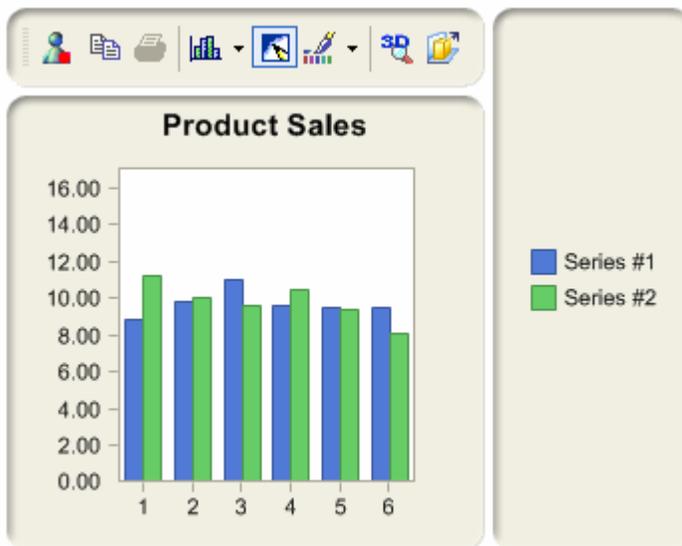


Chart FX image borders are prepared to handle additional tools and windows that can be displayed in the chart area, including Toolbar, Legend windows and Data Editor.

The chart displayed uses an Embed image border, and notice how the different elements are properly rendered with the same border type.

Transparency

With transparency, you can make chart elements completely invisible in the chart area, such as background pictures and gradients that apply to the entire chart area. Also note that this setting can be applied to any chart element that receives a color. For example, if you have a scatter chart and you want to hide certain points, you can use the **Point** property and assign a transparent color to only certain points in the chart.

Essentially, transparency is achieved by using the Transparent color. For example, if you want to set a background picture, you will want to make the chart backgrounds by using the following API:



Semi-Transparency

In Chart FX, the alpha blending feature is used to display a chart element that has semi-transparent pixels. In addition to a red, green and blue color channel, each pixel in an alpha bitmap has a transparency component known as its alpha channel. The alpha channel typically contains as many bits as a color channel. For example, an 8-bit alpha channel can represent 256 levels of transparency, from 0 (the entire pixel is transparent) to 255 (the entire pixel is opaque). For example, if you want to make the third series of your chart a semi-transparent green, you will set the Color.FromArgb property to (120, 0, 255, 0).

Changing the transparency of an object evenly reveals the image elements that lie beneath the object and because semi-transparency can be applied to any chart element that supports a color, you can create a myriad of effects. For example, in a 3D oblique chart, you can make the first series semi-transparent so other series are revealed in the 3D view; or you can use the Point object to have particular marker(s) display semi-transparency. Similarly, you can make the chart background semi-transparent so a background image is blended into the chart, which creates a nice watermark effect.

Color Palettes

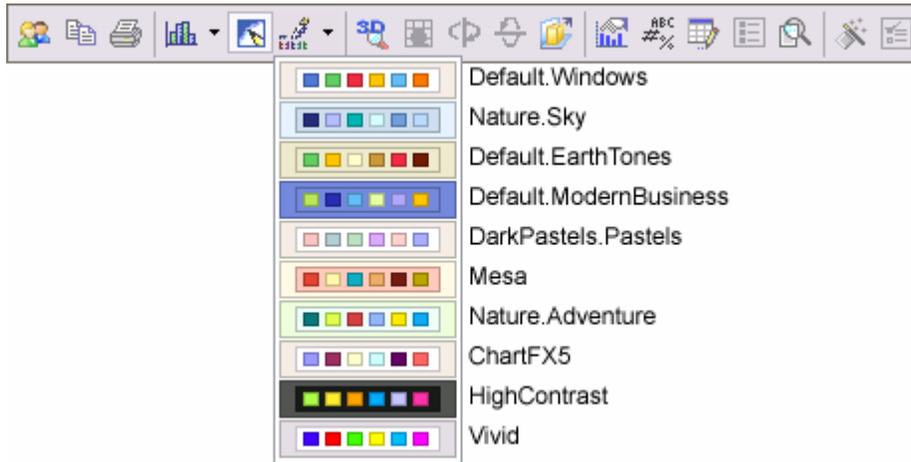
In most situations, you will want to assign colors to chart elements like markers, backgrounds and labels. However, our design team invested a lot of time creating default color palettes for Chart FX. This way, when you create a chart and populate it with data, everything looks nice and consistent (even if you have not taken the time to assign default colors to chart elements).

If you look into the Chart FX way of coloring your chart, you'll find out Chart FX provides other palettes that you, or your end users, can use to quickly change the chart.

Essentially, Palettes are a group of colors that apply to the different elements in the chart. When a palette is changed, all the elements that have not been forced to have a specific color will adapt to the new palette.

Using palettes from the Chart FX interface

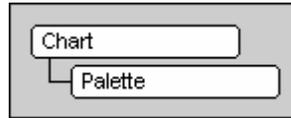
Palettes provide a special functionality since they allow end users to customize the chart colors with a single click. All default and extended palettes are listed in the Chart FX toolbar as a combo box that can be used to preview or apply a desired palette to the current chart. The image below shows the supported palettes available from the toolbar:



Please refer to the "Tool Customization" section in Chart FX the Programmer's Guide for more information on displaying and customizing the Toolbar in the Chart.

Using Palettes programmatically

Palettes are supported through your IDE properties list, or you can load and apply them through the chart using the Palette property. Please note, the palette property receives a string parameter with the name of the palette you want to load.



Since Chart FX palettes are extensible, you can create and integrate your own palettes programmatically, or let your users use them through the Chart FX Toolbar. We have used this functionality to create ten predefined palettes within Chart FX:

- Default.Windows
- Natural.Sky
- Default.EarthTones
- Default.ModernBusiness
- DarkPastels.Pastels
- Mesa
- Natural.Adventure
- ChartFX5
- HighContrast
- Vivid

Note: To properly load these palettes, you will simply use their string names (i.e. "DarkPastels.Pastels").

Palette Naming Convention

Notice how many of the palettes use a name consisting of two words separated by a period (DarkPastels.Pastels). Each of the Chart FX palettes is derived from a base color scheme, which provides the "first word". The unique combination of colors used to create a selected palette from a base color scheme is the "second word" in the name. Therefore, if a base color scheme only has one combination, the second word is not required. This can be observed in the Vivid palette for example. Because there is only one palette existing in Vivid, you only need to specify the base color scheme to complete the configuration (i.e. "Vivid").

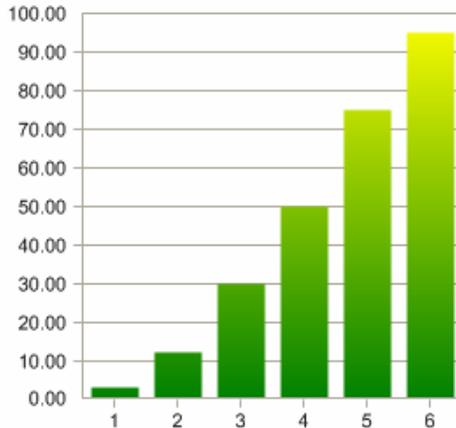
Gradients

Gradient fills let you create a gradual blend between colors in chart elements like markers and backgrounds. For example, gradients in chart markers (i.e. bars) can improve the chart's readability by giving a meaningful color representation to the data contained in the chart. Also, when applied to the chart background, gradients can help you achieve a superior visual effect for your Windows Forms applications and web sites.

By default, the color in a linear gradient changes uniformly. However, in this chapter you will learn how to use the Chart FX API so you can create and customize linear gradients so that the color may change in a non-uniform fashion.

Gradients in chart markers

In most situations, gradients are commonly used to achieve a cosmetic fill effect on a particular shape. However, when used in chart markers, Chart FX gradients serve a different purpose as they allow the user to immediately recognize data trends and call out important data in the chart.



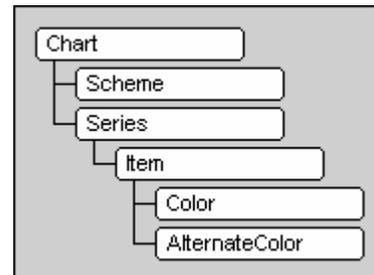
To illustrate this, the chart displayed uses a gradient fill that allows the user to immediately recognize markers that fall within a range with just a quick look at the chart.

To achieve this effect, you must set a special color scheme that activates the markers gradient fill. Then, Chart FX defines a virtual rectangle that starts at the top of the chart (defined by the Y Axis Max) and uses the marker color as a starting point (top) and an alternate color to blend uniformly to the bottom (X Axis).

Setting the **Scheme** property to "Gradient" and setting the **Color** and **AlternateColor** on the Series object will produce the Chart shown.

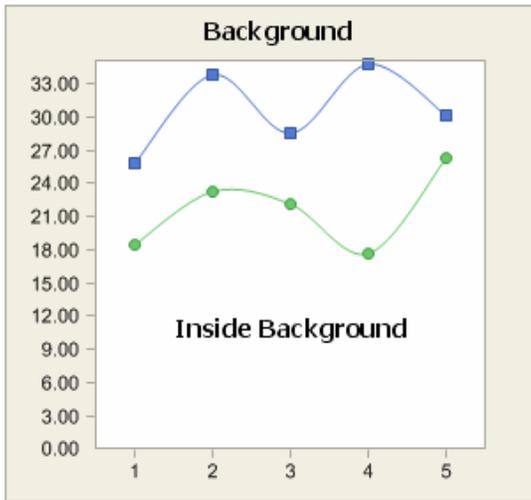
It is important to note that marker gradients are used for data analysis purposes in Chart FX. Therefore, marker gradients accept only two colors and are limited to a horizontal effect.

Finally, please remember the Y Axis Maximum will significantly affect the virtual rectangle on which the gradient is going to be painted. For example, in the chart shown above if the Y Axis Maximum is changed from 100 to 200, all chart markers will have intensity close to the alternate color (green) as they will be farther away from the top of the chart where the marker color (yellow) is used.



Gradients in the chart's background

As opposed to marker gradients, background gradients are used for cosmetic purposes only. In other words, background gradients are only used to enhance the chart's visual appearance and have no direct relation to the data contained in the chart.



In Chart FX, gradients can be applied to 2 rectangular areas in the chart area: The chart's inside background and the main background.

Chart FX allows horizontal, vertical, diagonal and radial gradients when applying gradient fills to either of these areas of the chart.

In order to create background gradients, you can use the Chart FX Wizard, Chart FX Designer, the Property List, or you can use Chart FX API to set the objects and its properties to create and display gradient fills.

By default, the color in a linear gradient changes uniformly. However, in this chapter you will learn how to use the Chart FX API so you can create and customize linear gradients so that the color may change in a non-uniform fashion.

For more information about how to create a Gradient Background and how to assign it to any of the Chart's backgrounds please refer to the Chart FX online Programmer's Guide and the API Reference.

Manipulating gradient colors, positions.

You can manipulate the GradientBackground object's properties to set visual attributes like colors, positions and color intensity. The following section will show you how to create multicolor and bi-color gradients.

Essentially, a gradient object provides a Color array property that you can use to set the colors you want for your gradient. These colors will then be uniformly blended in the chart's background by dividing the bounding rectangle into equal sections and blending from one color to another in the Color array.

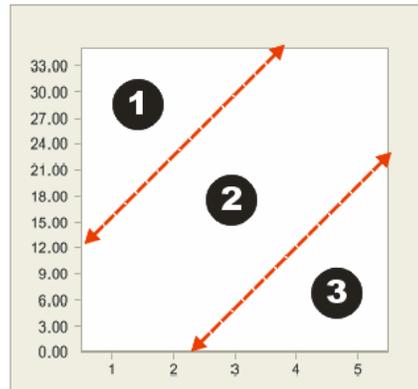
If for example, you pass four values to the color array, Blue, Red, Yellow and Green, and set the Chart's Inside Background as "Forward Diagonal" gradient, Chart FX will create 3 regions that will blend uniformly:

Here region 1 will go from Blue to Red, section 2 from Red to Yellow and section 3 from Yellow to Green.

If this arrangement is not convenient, you may alter the positions of each blending area by using the Position array supported by the GradientBackground object.

To understand how positions work in the bounding rectangle, imagine the first position (where the gradient starts) as position 0; the ending position would then be 1 and then intermediate positions would be fractions between 0 and 1.

In our previous example if we wanted the gradient from Blue to Red to occupy only 10% of the bounding rectangle and the gradient from Red to Yellow to fill 70% of the area, we could have used the Position array as follows: 0, 0.1, 0.8 and 1.



Bi-Color gradients

In some cases, you may want to work with bi-color gradients and simply repeat a gradient pattern across the chart's background. If this is the case, instead of defining a Color array with the 2 same colors over and over, you can simply create a bi-color gradient.

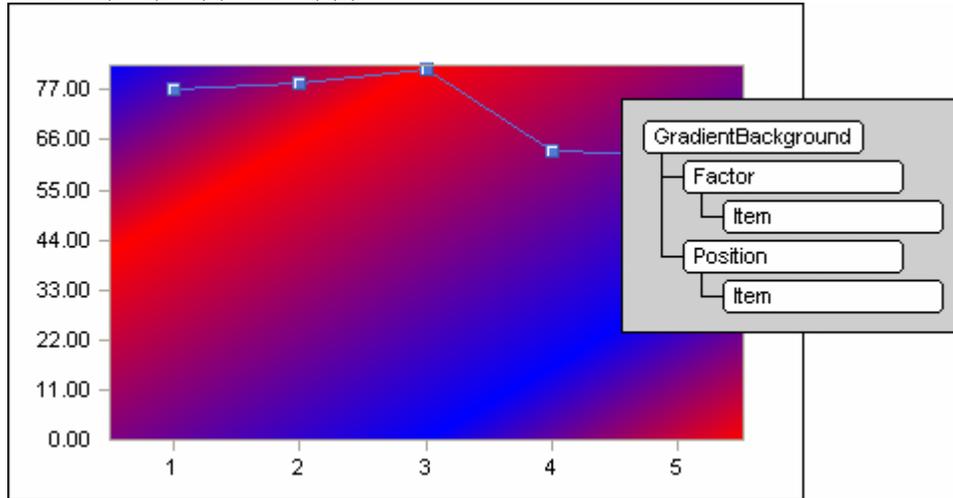
A bi-color gradient requires **Position** and **Factor** (color intensity) arrays to display a repeating pattern across the chart's background. Previously we described the effect the Position array had on the gradient. In bi-color gradients a new Factor array must be defined to instruct Chart FX on the color intensity on each position.



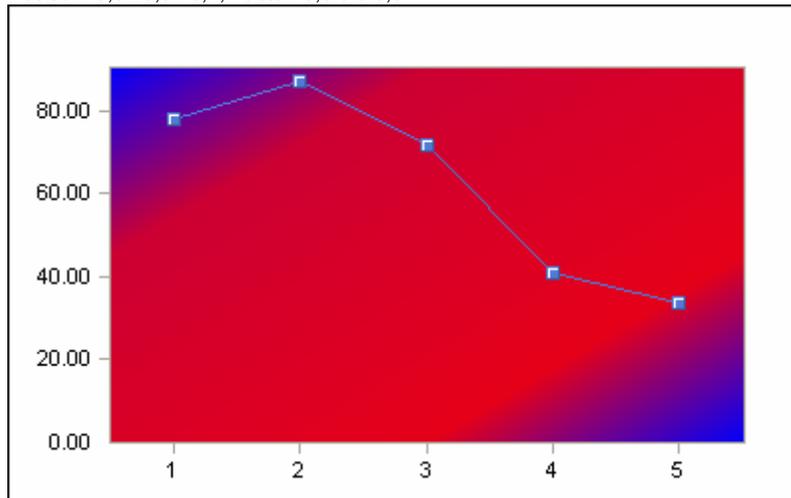
To understand the Factor array, consider that the values it takes are a fraction between the 2 colors in the color array. If set to 0, then the resulting color will be the color available in the position 0 of the Color array and similarly with a value of 1.

For example take a closer look at the Factor array and the effect it creates on the gradient in the following 2 samples:

Position: 0,0.25,0.75,1, Factor: 0,1,0,1



Position: 0,0.25,0.75,1, Factor: 0,0.8,0.9,0



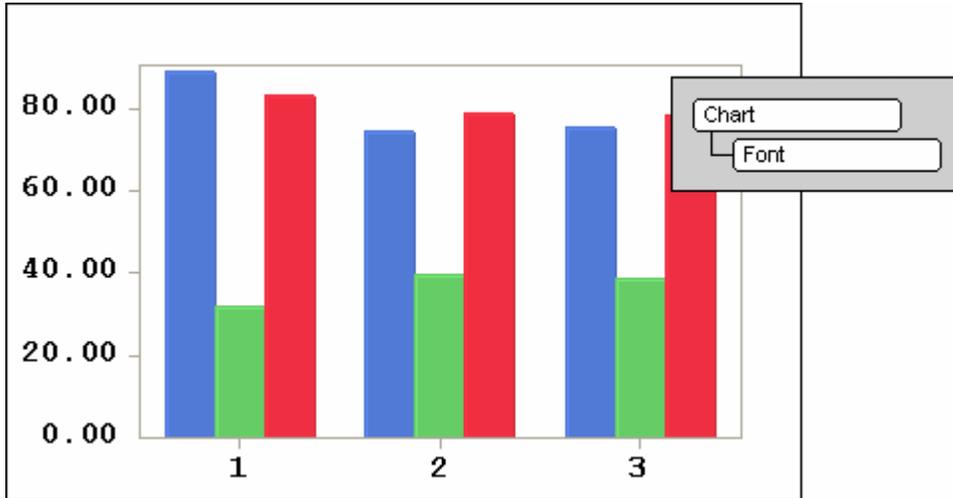
Note: The Factor array does not apply to multicolor gradients (3 or more colors).

Handling Fonts and Titles

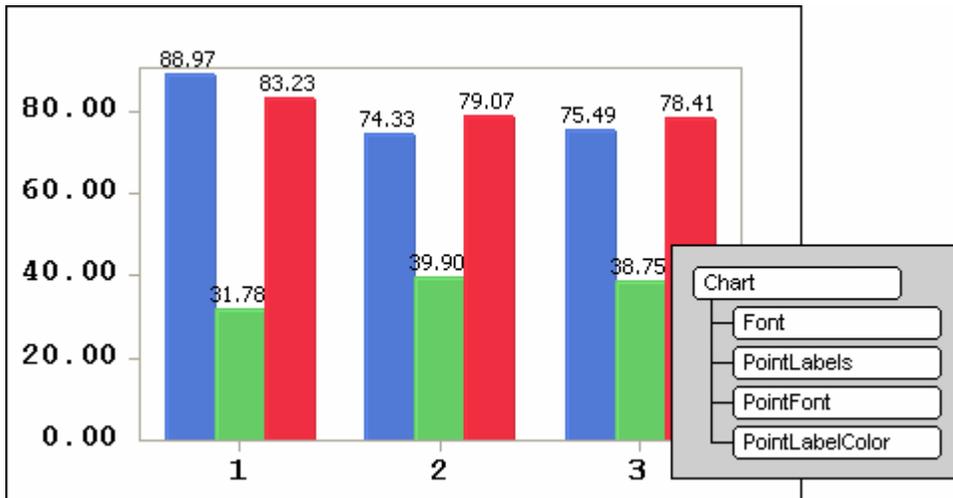
Fonts

As you have seen from the beginning of this section, visual attributes can be configured at a general level in the Chart object, or at a more granular level. Similarly, Fonts can be applied in general at the Chart level, or on a granular level for various sections of the chart such as: Axis Labels, Points Labels, Legends, Series Legends, Data Editor, etc.

If the Chart Object is assigned a font such as Bold Arial size 12, that font setting will be used for all chart objects that make use of fonts for rendering.



Now if you want to show the Point Labels in a different font, size and color, you can custom configure them using the **PointLabelsFont** and **PointLabelColor** properties.

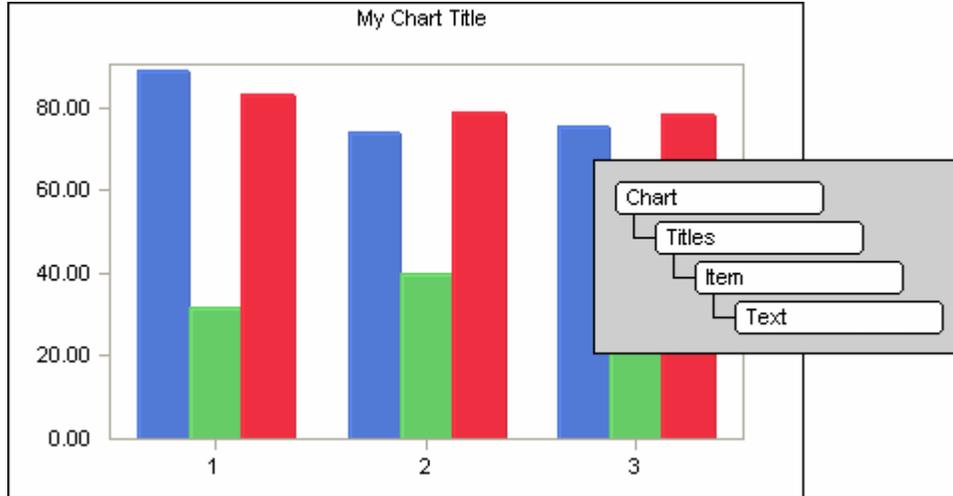


Titles

To improve the chart's readability, Chart FX supports the creation of titles that you can use to set the chart main title. A title is a string that can be assigned to any portion outside the main chart area.

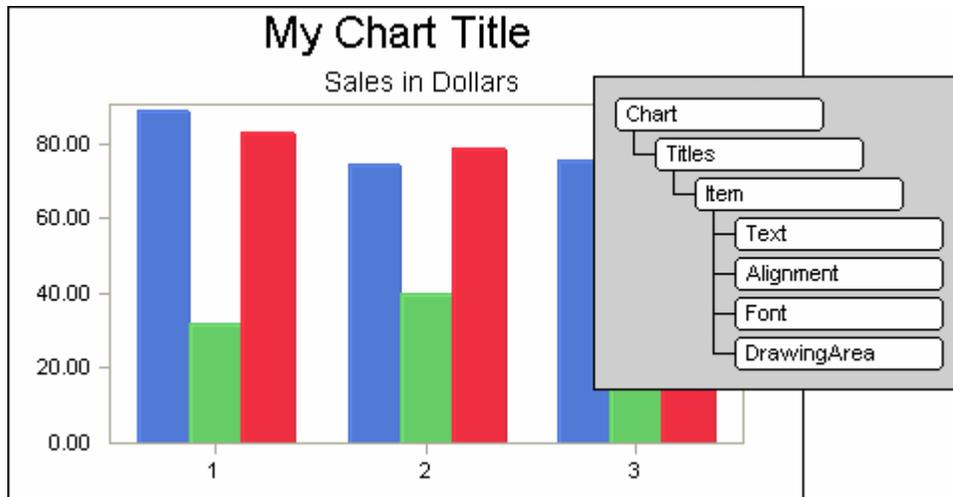
This string can be positioned, aligned and formatted according to the different properties supported by the **TitleDockable** object which allows the creation of an unlimited amount of titles in the chart area. Titles can be assigned using the Property List or you can create and control them programmatically.

By default, all the titles you create will be positioned at the top of the chart, or where the main chart title is displayed. For example, if you use the following API, it will create a title on the top portion of the chart.



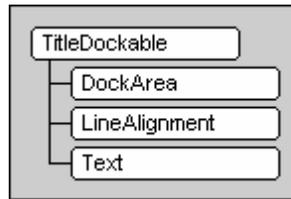
You can create multi-line titles by simply adding an additional title in the same position. These lines can then be independently aligned and formatted.

The **Title** object supports a series of properties that you can use to align position and format the different titles in the chart.



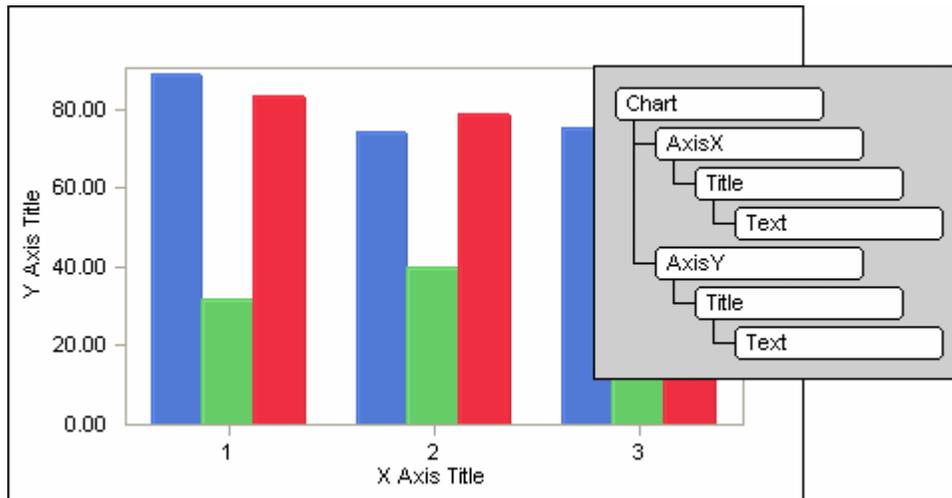
TitleDockable Class

The **TitleDockable** class supports the additional **DockArea** and **LineAlignment** properties that allow a title to be docked in the chart area. In order to utilize this functionality, you must create your title objects as a TitleDockable object.



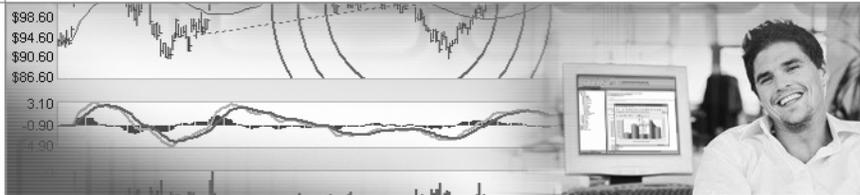
Axes Titles

Titles can be created on the chart's Axes. They can be assigned, aligned and formatted according to the different properties supported by **Title** object. Axes Titles can be assigned via the Properties List or you can create and control them programmatically.



Axes Titles are attached to the Axis to which they belong. If the Y Axis gets moved to the right side of the chart (Far), the Y Axis Title will get moved there as well. If the X Axis is hidden, the X Axis Title will be hidden also.

Axes

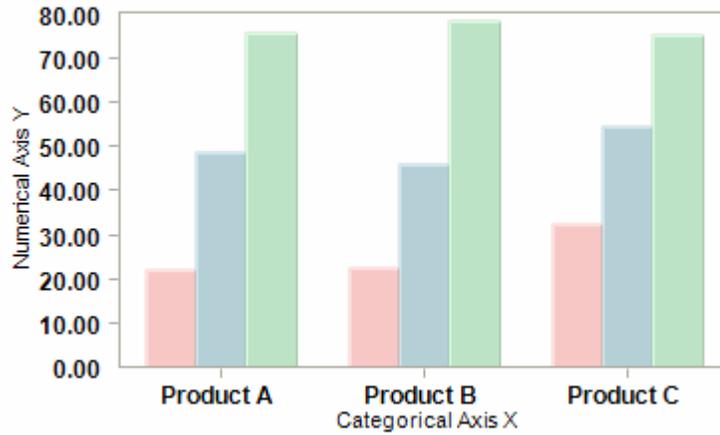


Overview

The Axis object provides a myriad of properties and methods that allow control over many axis settings, including labeling, scaling and formatting attributes like colors, gridlines and tickmarks.

Introduction

Axes are one of the most important elements since they contain information that help the user interpret the chart. In most instances, charts are drawn in an XY-coordinate system. The Value or Y axis displays a range of values represented by the numeric data, and the Category or X axis displays how the data is broken down in the chart. The following picture highlights the Value (Y) Axis and the Category (X) Axis:



When you first create and populate a chart, Chart FX uses a “best-fit” algorithm that accommodates axis settings automatically. This behavior minimizes the initial programming effort of customizing the chart’s axes. However, in most cases, programmers will face complex axis configuration scenarios like labeling, scrolling, scaling and formatting that must be used in order to help users follow and analyze data.

This section describes the most common axis manipulation scenarios and advanced features and techniques that are used to enhance the chart’s readability like creation of logarithmic scales, multiple axes as well as axis sections and panes. It is important to note that end users can also manipulate the most common and recognizable axis settings via a user interface (toolbar and context sensitive dialogs) provided by Chart FX. For additional information, please refer to the “Chart FX User Interface” chapter of this manual.

Recognizing Numerical and Categorical Axes

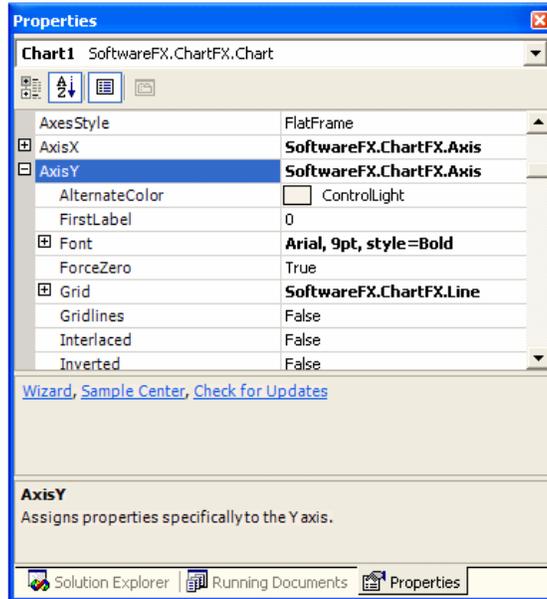
The first step to properly configure an axis is to recognize whether you are dealing with a numerical or categorical axis. Sometimes this is not trivial. For example, in an XY or scatter, both axes are numerical. Similarly, for some chart types, you can tilt the x axis to be vertical and while still remaining a categorical axis. Also, some chart types don't even have X or Y axes; for example a pie chart will not respond to any of the properties or methods provided in the Axis object.

In essence, a numerical axis will contain numbers associated with each tick mark and, therefore, will respond to any property or method that is used to scale or format such numbers. When dealing with numerical axis you don't have to worry about labeling each tick mark as Chart FX will automatically select the label (or number) for each tickmark in the axis. On the contrary, a categorical axis displays labels (not numbers). In most cases, you will need to label each tick mark to improve the chart's readability. If no labels are specified for a categorical axis, Chart FX will assign numerical tags (1,2,3,...) to each tickmark in the axis.

The Axis Object

The **Axis** object provides a myriad of properties and methods that allow control over many axis settings, including labeling, scaling and formatting attributes like colors, gridlines and tickmarks.

For accessibility purposes, Chart FX exposes two complex properties (**AxisX** and **AxisY**) of the **Axis** type. In many development environments (e.g. Visual Studio .NET) these properties can be accessed directly from the properties list so these axes can be configured at design time easily.

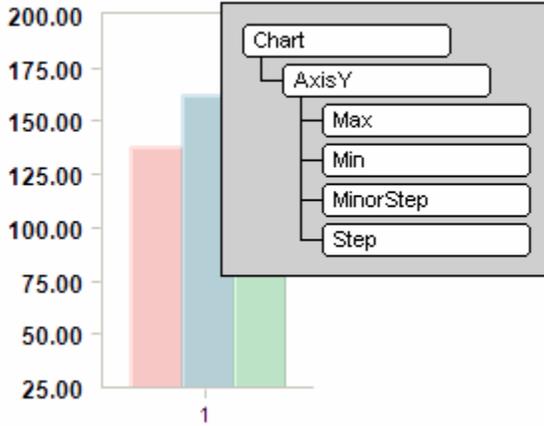


Note: For design time support in other development environments and tools, please refer to the Chart FX Resource Center and Electronic documentation. Additionally, the Axis object is used to create an unlimited number of axes (categorical or numerical). Multiple axes support (including Secondary Y Axis) is described later in this section.

Axes Scaling

Automatic Axis Scaling

As previously mentioned, when you first create and populate a chart, Chart FX will use a “best-fit” algorithm that accommodates the numerical axis settings automatically. This behavior minimizes the initial programming effort of customizing the chart's numerical axes. These values are chosen according to the chart type, data and the chart's size and then adjusted so the axis shows “nice” values in the chart. Finally, the chart markers will be plotted in the chart area according to their value and the position in the chosen scale.

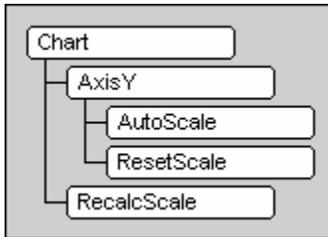


You can reset any of these scale values using several properties and methods associated with a numerical axis. Some scale values include: **Minimum** (or where the axis starts), **Maximum** (or where the axis ends) and **Step** (or major unit).

The **Step** property controls where labels and tickmarks are shown in a numerical axis. This fraction is usually called the major unit.

The **MinorStep** property allows you to set a minor unit on which you can only show and configure tickmarks and gridlines.

Recalculating Axis Scale Information



When new data is set to the chart, Chart FX will then again recalculate the scale values according to the new data set. This behavior can be controlled by the **AutoScale** property of the axis. This property is enabled by default. If you choose to disable this property you must set the Min and Max values manually.

If you preset any of the scale values (Min, Max, Step or MinorStep) Chart FX will not recalculate the scale when new data is sent to the chart. This behavior can be altered by invoking the AxisY **ResetScale** method. This method recalculates scale values associated to that specific

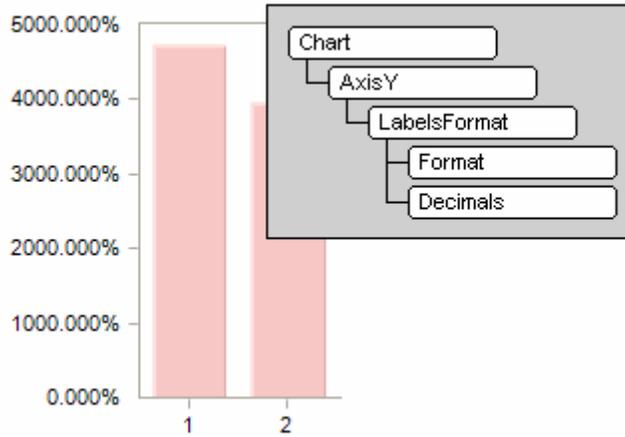
numerical axis. This method is particularly useful if you know the specific axis that needs to be reset.

If you want to recalculate ALL numerical axes settings, the Chart exposes a method called **RecalcScale** (as depicted in the Object Model diagram above). Please keep in mind that RecalcScale method applies to ALL numerical axes and reads the entire data array. In other words, you should avoid calling this method repeatedly as it will adversely affect the performance of your application.

Formatting a Numerical Axis

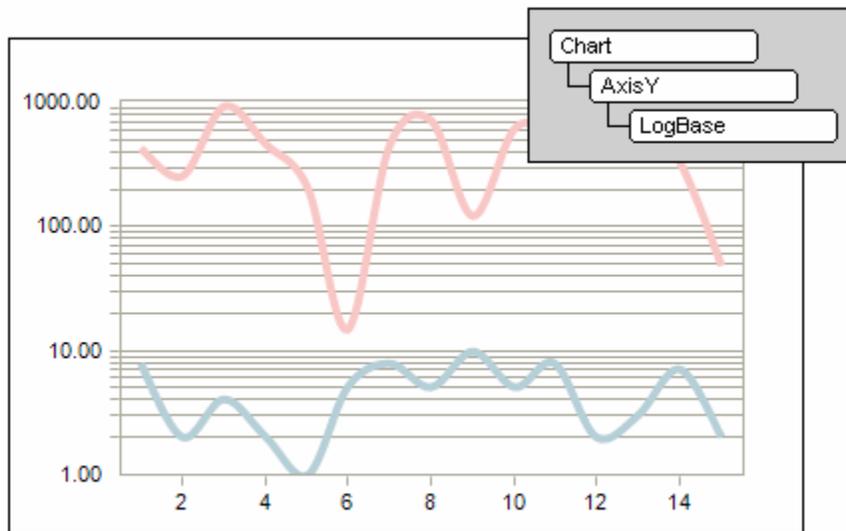
Formatting a numerical axis is as simple as setting any of the pre-defined axis formats in Chart FX using the **Format** property supported by the **ValueFormat** class. For example, you can set scientific notation in the primary Y Axis using the **LabelsFormat** property.

Please make sure you check the **Format** property on the API Reference for more information on any pre-defined formats or constants supported by Chart FX.



Logarithmic Scales

When plotting large numbers, Chart FX supports independent logarithmic scales on any of the axes (Primary, Secondary Y and X when used in a numerical sense). You may have Log-Log, Linear-Log or Log-Linear scales.

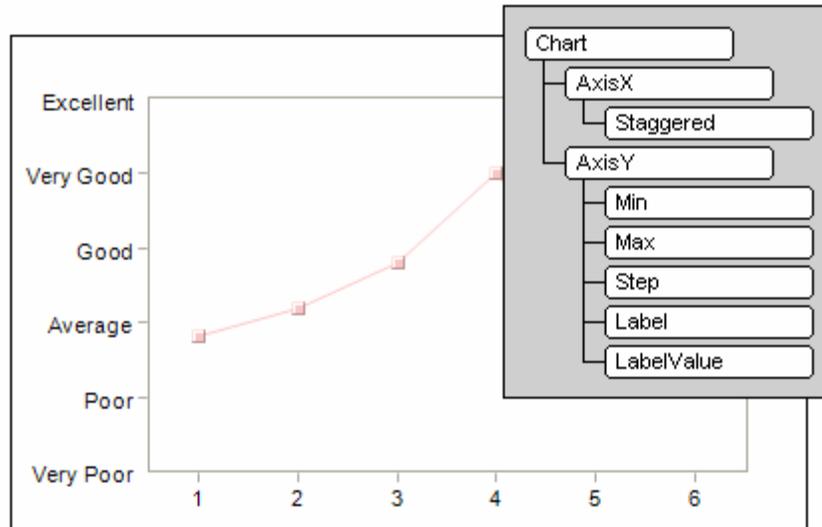


To set a Logarithmic scale to any of the Chart FX numerical axes you can use the **LogBase** property. This property will set a logarithmic scale for a numerical axis and recalculate the values as powers equal to the setting of this property. For example, you can set the Y Axis to be logarithmic with base 10.

Axis Labeling

Custom Labeling a Numerical Axis

Although it is not common to assign custom labels on a numerical axis, there may be situations in which a numerical axis may need to display or map a series of labels that describe the specific value of a tick mark. For example, consider the case of ratings. Ideally you would want the chart to display labels on the axis itself, as depicted in the following picture:



To accomplish this, you must:

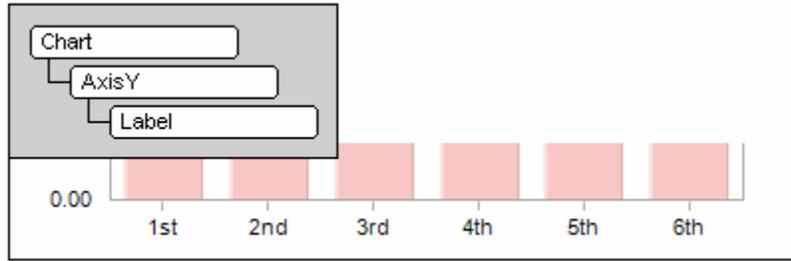
Make sure you manipulate the scale settings (**Min**, **Max**, **Step** properties) so you end up with six tickmarks in the axis (i.e. 0, 20, 40, 60, 80 and 100).

Use the **Label** property and add six labels in the correct order corresponding to ticks allocated with **Min** and **Max** above. For example start with "Very Poor" (index 0) and end with "Excellent" rating (index 5).

Use the **LabelValue** to specify the interval where those labels should be placed on a numerical axis. If you specify a **LabelValue** of 20, each label will be placed next to its corresponding tick mark.

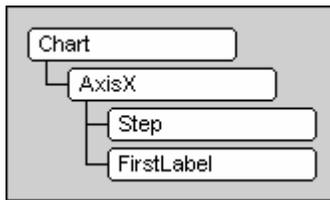
Labeling a Categorical Axis

By default, a categorical axis (X Axis) is labeled with tags (1, 2, 3...) on every tick mark. You can improve the chart's readability by assigning meaningful information to these tickmarks. The **Label** property can be used for this purpose. When used, the chart will display these labels in their appropriate tickmarks as depicted in the following figure:



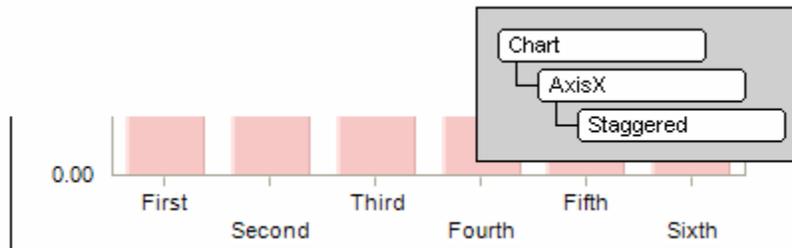
Note: If you populated the chart from a database, these labels are automatically assigned from text fields in the query. For more information on populating charts with data coming from a database please refer to the "Passing Data" section earlier in this manual.

By default, Chart FX will calculate the amount of labels that can be displayed according to the chart size and will hide some labels if all cannot be shown. There are some properties and techniques that will help you fit more labels on a categorical axis.



Just like a numerical axis, a categorical axis supports the **Step** property to control the interval labels are shown along the axis. For example, when a value of 3 is applied to the Step property, labels will be placed on positions 3, 6, 9 and so on. The **FirstLabel** property works in a similar way, only that it forces the first label to be placed on a specific tick mark. For example, if a FirstLabel is set to 1, then X Axis labels will be shown at positions 1, 4, 7 and so on.

You can also make room for more labels if you rotate them 90 degrees using the **LabelAngle** property or you can make the labels staggered as depicted in the following figure:



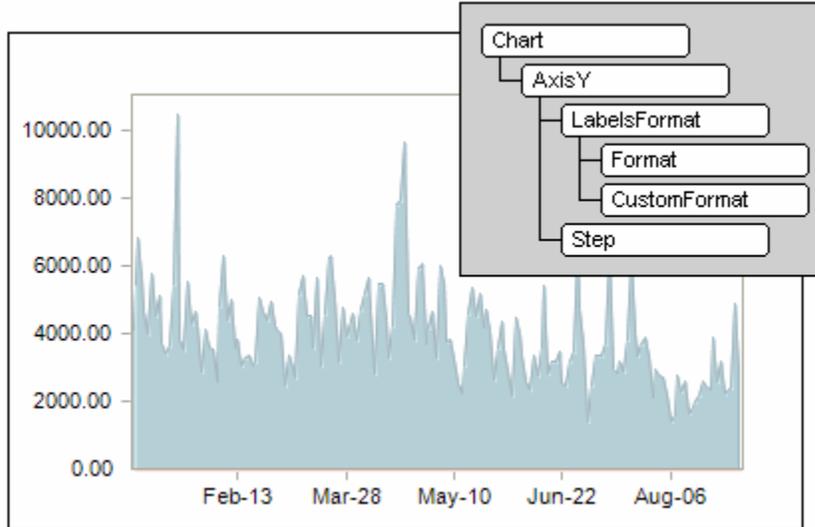
Note: The **Axis.Style** property uses the bitwise operator (OR) to add or remove settings from the property. For more information, please refer to the **AxisX.Style** property in the Chart FX API Reference.

Additionally, the **AxisX** provides a myriad of properties that can be used to customize labels like font, gridlines, tickmarks, colors, axis styles, label rotation and other special effects. For more information, please refer to the **Axis** object in the Chart FX API reference or the samples repository in the Chart FX Resource Center.

Manipulating Dates and Times in a Categorical Axis

Chart FX provides means of formatting the axis labels to display date/time labels through the **Format** property. This property supports either **Date** or **Time** constants specifically for displaying dates in a categorical axis. For example, if you have passed data that ranges in minutes, you would select the Time constant; or if you have passed data that ranges in days you can setup the **Format** property to **Date** and control how frequent labels are displayed using the **Step** Property.

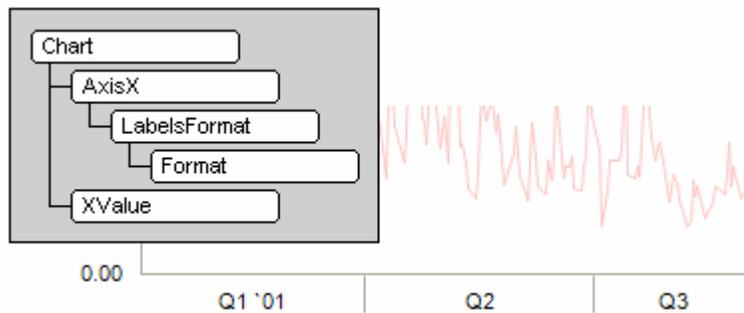
For example, you can setup a categorical axis and format the tick mark labels using a **CustomFormat** date mask that formats the date with the month name and the day as depicted in the following figure:



Smart Date/Time axis

Although the ability of formatting dates in a categorical axis provides a certain freedom for customization, these labels are still fixed and do not change dynamically as the chart is resized in the browser or in a form. The Smart Date Axis involves passing dates as XValues to the chart and setting the Format property to Date or Time constants. A fully functional "Smart Date/Time Axis" sample is provided in the Resource Center.

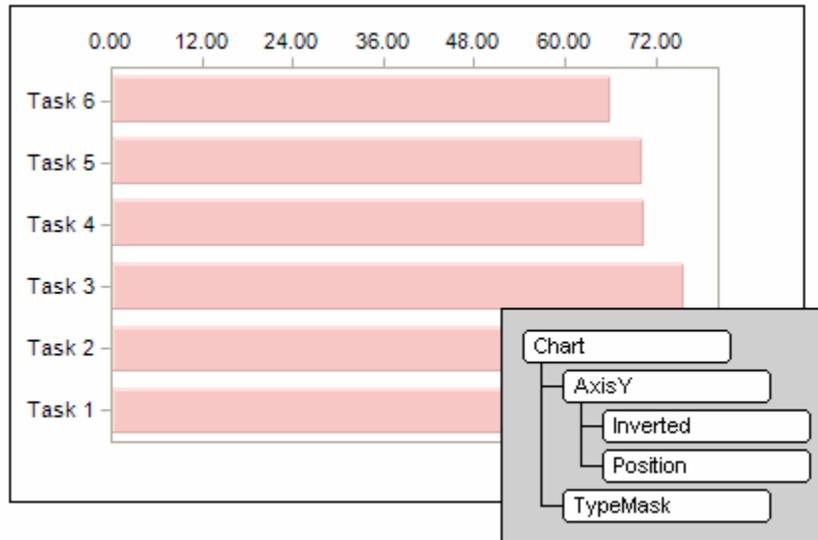
Because the Smart Date Axis IS NOT a categorical axis but a numerical one, it treats dates as numbers and it is capable of adapting dynamically to the chart size. For example, if the chart contains 3 years worth of data in days, you WILL NOT need to label each tick mark as Chart FX will assign labels accordingly making the process a lot easier to code as well as improving the chart's readability. In the picture displayed below, Chart FX selected the Quarter view as the most appropriate label for the chart size.



Tilting, Inverting and Positioning Axes

The most common position for labels in numerical and categorical axes is at the left and bottom portion of the chart, respectively. Nevertheless, you could run into situations where axes must be tilted, inverted or positioned in different areas of the chart.

For example a **Gantt** chart not only tilts the categorical axis to the left side but usually displays the numerical axis (or dates) at the top of the chart. The following image depicts a chart with the categorical or x axis tilted with the `TypeMask` property (`Horz` constant) and the Y axis is repositioned to top portion of the chart:

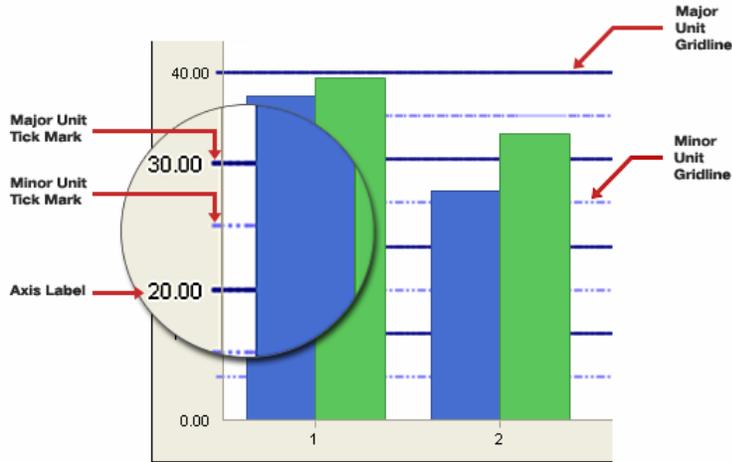


Note: Not all chart types support axis tilting. For additional information please check the online documentation. In addition, if you have a need for creating Gantt charts we strongly recommend the Chart FX Gantt Extension as it is specifically designed to handle Gantt related features. For more information on Chart FX Extension please visit <http://www.softwarefx.com/extensions>.

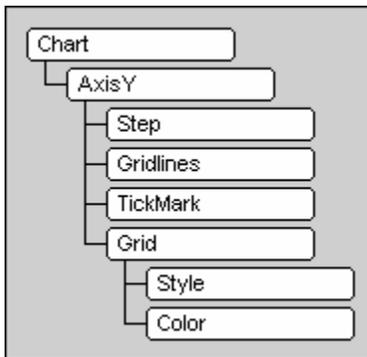
Axis Gridlines and Tickmarks

In some situations, you may want to make use of gridlines and tickmarks to improve the chart's readability. We have already demonstrated how you can control scaling with the **Step** and **MinorStep** properties. It is important to note that you can create and control gridlines as well as tickmarks around major and minor units within the chart. These elements allow the user to easily detect when a particular value plots in a certain value or range.

Both the **AxisX** and **AxisY** have properties that allow you to set parameters that configure gridlines and tickmarks on any axes of the chart. However, it is important to understand where gridlines and a tickmarks are positioned in the chart, they can be easily identified in the following picture:



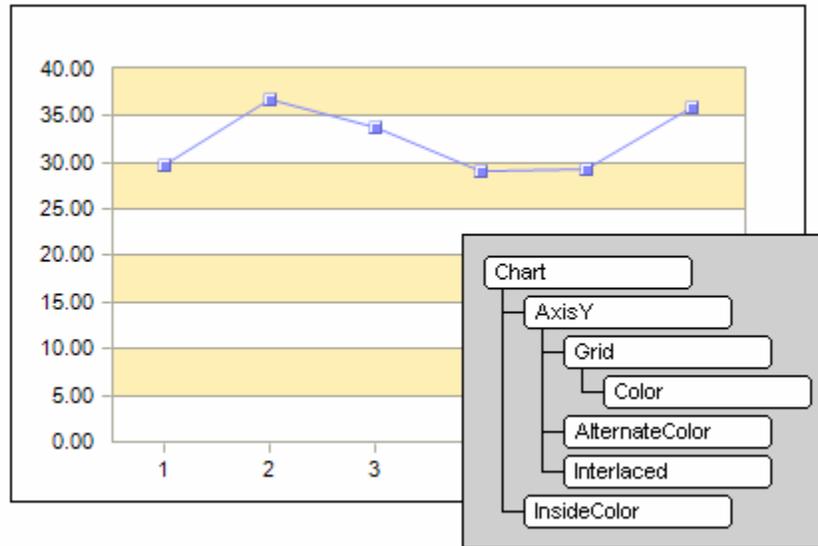
You must remember axis labels are placed in the major unit controlled by the **Step** property. Visual line attributes may be applied to gridlines using the **Grid** property in conjunction with the supported Line class members (**Color**, **Style**, **Width**, **EndCap** and **StartCap**). The **Gridlines** property is used to show or hide the gridlines in the chart area, and the **TickMark** property is likewise used to control tickmarks. In the figure shown above, it is easy to identify a step of 10, with tickmarks placed outside the axis and Dash-Dotted gridlines. This can be achieved using the following API calls:



Similarly, when you set a minor unit using the **MinorStep** property, labels will not be shown in this location but gridlines and tickmarks can be set. For example, in the figure shown above, the **MinorStep** property has been set to 5 with minor tickmarks shown as a "cross" and a solid gridline associated with the minor unit.

Interlaced Grids

In some cases interlaced gridlines are very useful to improve the chart's readability; they are depicted in the following figure:



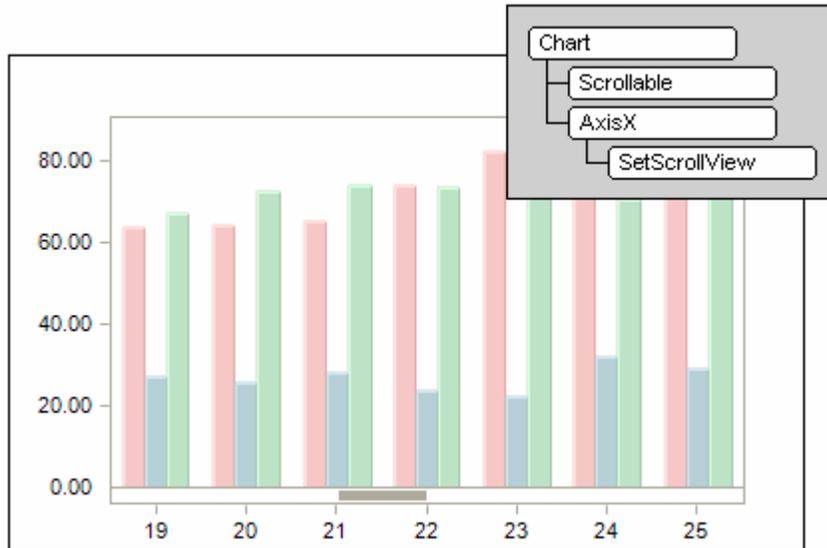
To achieve this effect, you need to set the chart's **InsideColor** property, the AxisY grid **Color**, and the Axis Y **AlternateColor** properties with the color you want for the interlaced stripes, plus set the **Interlaced** property supported in the Axis object.

Axis Scrolling

Although Chart FX provides a wealth of properties to control how labels are displayed on an axis, the number of data points and the chart size become important factors influencing how many labels you could fit within a chart or how much data is actually visible. In other words, some charts simply have too much data for the desired chart size and there's little you can do to improve the chart's readability.

Chart FX provides an elegantly designed scroll bar that attaches to an axis and allows end users to zoom in and view portions of an entire chart. All axes in Chart FX support scrolling.

To enable scrolling you must use the **Scrollable** property of the axis you wish to allow scrolling. The scroll bar will appear docked right next to the axis labels as depicted in the following figure:



Once scrolling is enabled the user can drag the scroll bar thumb to a desired position in the chart. By using the scrolling feature data points will not be crunched inside the chart area and all points will show their respective labels. This enhances the chart's readability and allows the user to view portions of the chart at a time.

Note: Scrollbars will appear automatically when the end user uses the Zoom tool available in the Chart FX toolbar.

Controlling the Scroll position programmatically

Although the scroll bar is an end user tool, you may want to control the number of points that are visible in a specific scroll area. This can easily be done with the **SetScrollView** method supported by the Axis object.

For a categorical axis, this method receives the index of the points you want to see in the chart. For a numerical axis, the SetScrollView method receives the values you want to show in a specific scroll view.

Chart FX will fire an event called **UserScroll** every time the user interacts with the axis scroll bar. This event may be useful if you need to allow the user scroll two charts simultaneously as the user interacts with a single scroll bar.

Please refer to the Chart FX API Reference for more information on the **UserScroll** event supported by Chart FX.

Creating Additional Axes

Creating and displaying additional axes in the chart is very similar to assigning a secondary Y Axis.

For example, if you want to add a third series to the previous chart, which values range between 0 and 1; both existing scales (0-100 and 0-2000000) are still inadequate to represent fractions between 0 and 1. Therefore, we need a fourth axis to represent these values. Please remember that three axes are already displayed in the chart: Y Axis, Secondary Y Axis and the X Axis.

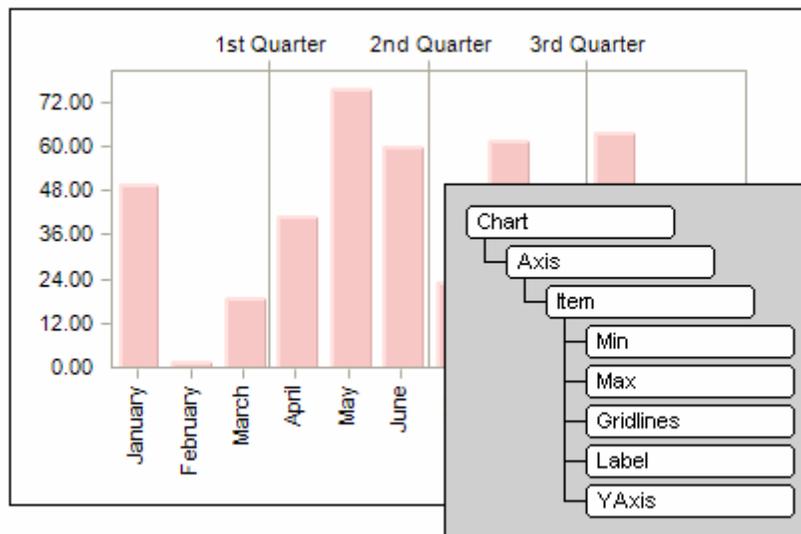
You can create as many as there are series in the chart. To add the fourth axis we will assign the third series to a new axis index, in this case 3.

Please remember Chart FX indexes are zero based. In addition, this assignment must be applied before the chart is populated so Chart FX can initialize the fourth axis properly. If this can't be done before data is applied, then you must use the **RecalcScale** method after populating the chart.

After you have assigned the series to its appropriate axis, you can use properties exposed by the Axis(n) object to configure its visual and scaling attributes.

Additional Category Axes

Additional category axes can significantly improve the chart's readability by providing another categorical scale to the already existent x axis. For example, the following chart features a double categorical axis highlighting months as well as quarters in the year:



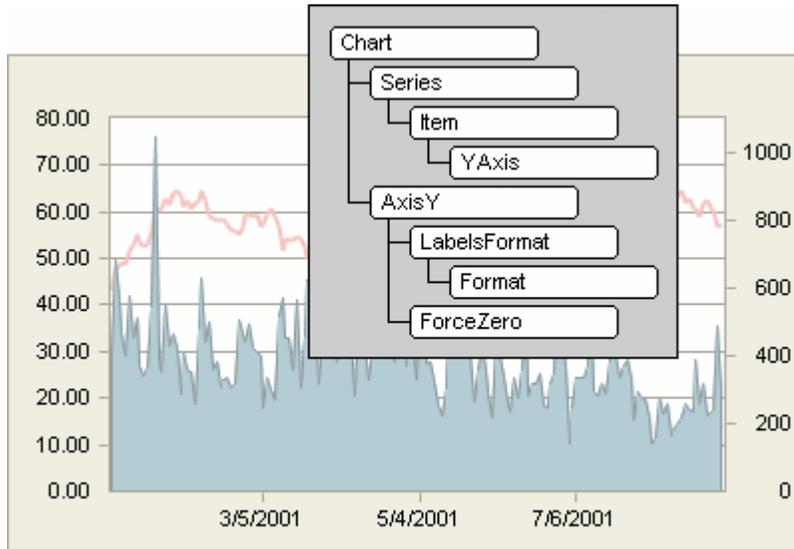
The trick to adding an additional category axis is simply creating a new axis using the axis object and using a new index for the additional categorical axis. Make sure you keep track of the indexes that you have already used when creating a new numerical or categorical axes. Furthermore, the YAxis property of the newly created axis should be set to false. In the image shown above we have also manipulated the Min, Max and Step so it shows gridlines only every three months in order to highlight the respective quarters.

Creating a Secondary Y Axis

When the range of values for different data series varies widely, or when you have mixed types of data such as price and volume, you can plot one or more data series on a secondary value (y) axis. The scale of the secondary axis is independent and reflects the values for the associated series.

The first step in creating a secondary Y axis is to assign a series to the secondary Y Axis using the **YAxis** property in the Series object. After you have assigned the desired series to the secondary Y axis, you can use the **AxisY2** similarly to the **AxisY** to handle scales, gridlines and different settings for the secondary Y Axis.

For example, the following chart contains price and volume information. We have assigned the volume series to a secondary Y axis that ranges in the millions while the price remained in the primary Y axis with values ranging between 40 and 80. We are also displaying the primary Y axis using the Currency format.



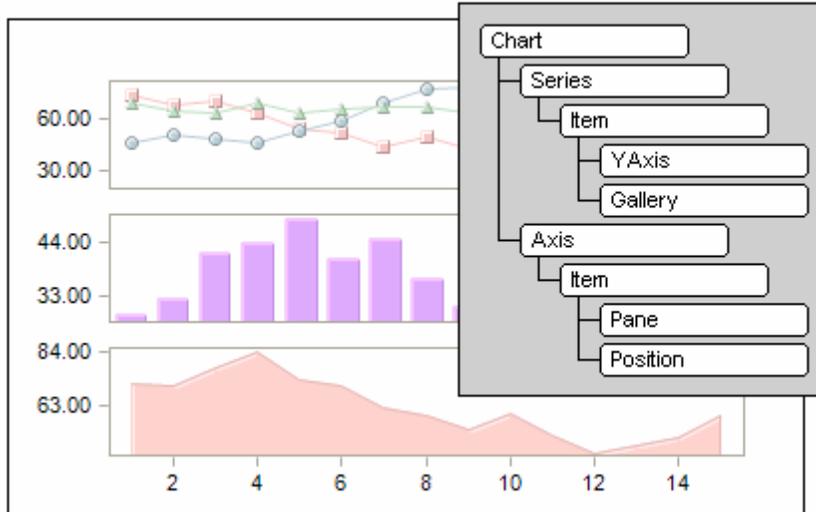
Note: A sample project with code is available in the Chart FX Resource Center.

Axis Panes

Sometimes, creating additional axes is not enough to highlight and properly separate data. The Chart FX Axis Panes allow the developer to display multiple sections of data in a single chart window. This feature allows developers to separate series allowing for improved analytical experience as each pane displays different scales and different chart types as required.

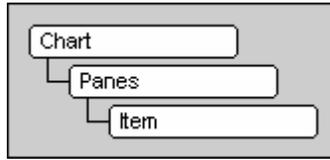
To create panes, you must first assign series to a new numerical axis as mentioned earlier, and then use the Pane property exposed by each axis. Please note that multiple series may be configured to a single pane as well.

In the following example, we have a 5 series chart in which the first three series have been applied in the first pane and additional panes have been created for the other two series which are displayed as a bar and an area chart, respectively.



Axis Panes Attributes

The Pane object exposes a collection of members that allow the developer to control basic visual attributes for a selected pane item. Please see the short description of each below, for further details please review the API reference.



InsideBackObject - This member is used to configure a background image or GradientBackground object to a chart pane. For more information regarding creating GradientBackgrounds, please see the Visual Attributes of the Programmer's Guide/Interactive samples in the Chart FX Resource Center.

InsideColor - This member is used to get or set an inside color for a selected Pane object.

Proportion - This member is used to get or set a value indicating the proportional size of the selected pane. For example, if there are three panes in a chart. Pane1's proportion is configured to the value "1". Pane2's proportion is configured to the value "2". Pane3's proportion is set to the value of "1" as well. Assuming the entire charting area represents 100%, Pane1 will use 25% of the chart area, Pane2 will use 50%, and Pane3 will use 25%. Other chart components visible in the chart area may affect the final size of the panes.

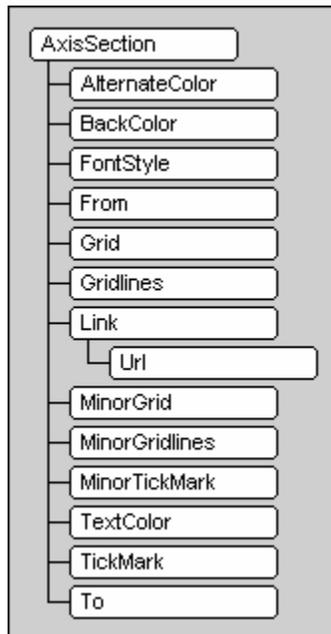
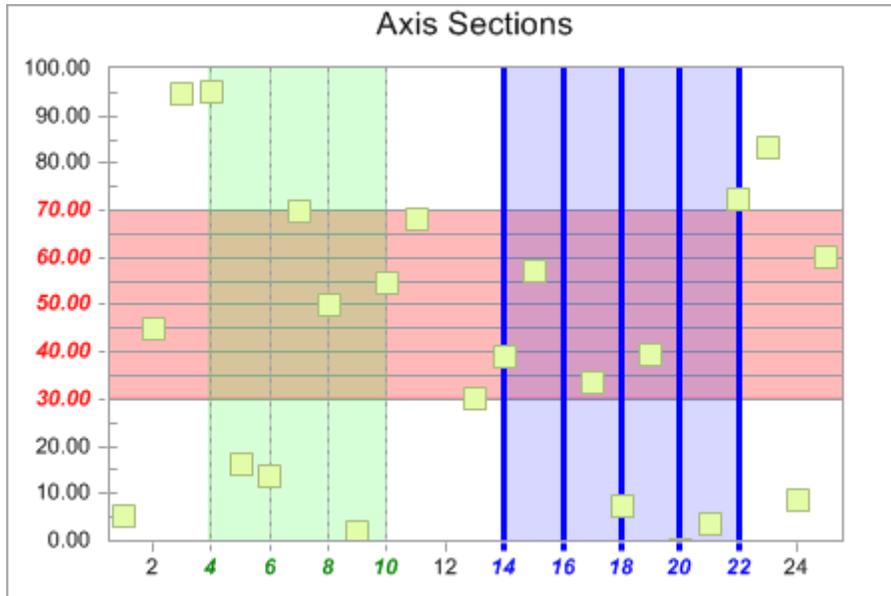
Separation - This member is used to get or set a value indicating the space (in pixels) in between panes.

Title - This member is used to obtain the Title object used for a selected Pane object. All the supported members of the Title object are exposed for the pane's title.

Axis Sections

The Axis Section implementation is another analytical feature offered by Chart FX. When using Axis sections in your applications, end users will quickly identify focus areas of a chart and will be able to easily determine whether values fall into a particular range. This analytic enhancement makes data driven charts more pleasing to the eye and easily deciphered by users.

Take a closer look at the following chart. The values between 30 and 70 are easily identifiable.

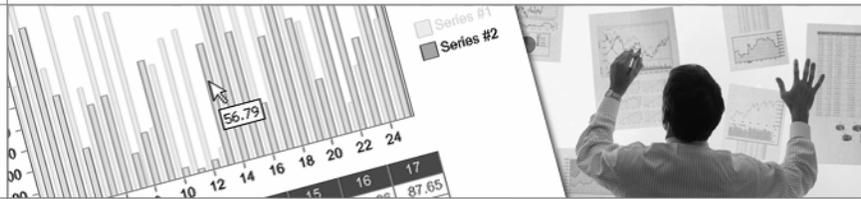


The **AxisSection** object allows the creation and customization of the configured chart axes. This provides the developer with a way to emphasize axis sections in order to improve readability of charts. To create an axis section, first you must specify which Axis object in the chart you want to create a section in. This can be done using the AxisX or AxisY properties of the Chart object.

Once the AxisSection object has been created, you may specify the range (**To** and **From** properties), the **BackColor**, grids and grid attributes, tickmarks, and much more. Many of the properties supported by the Axis object itself are customizable for the AxisSection object. Please see the API reference for a thorough explanation of the AxisSection and AxisSectionBase classes.

For implementation examples for AxisSection objects, please refer to the Interactive Samples section in the Chart FX Resource Center.

Data Analysis



Overview

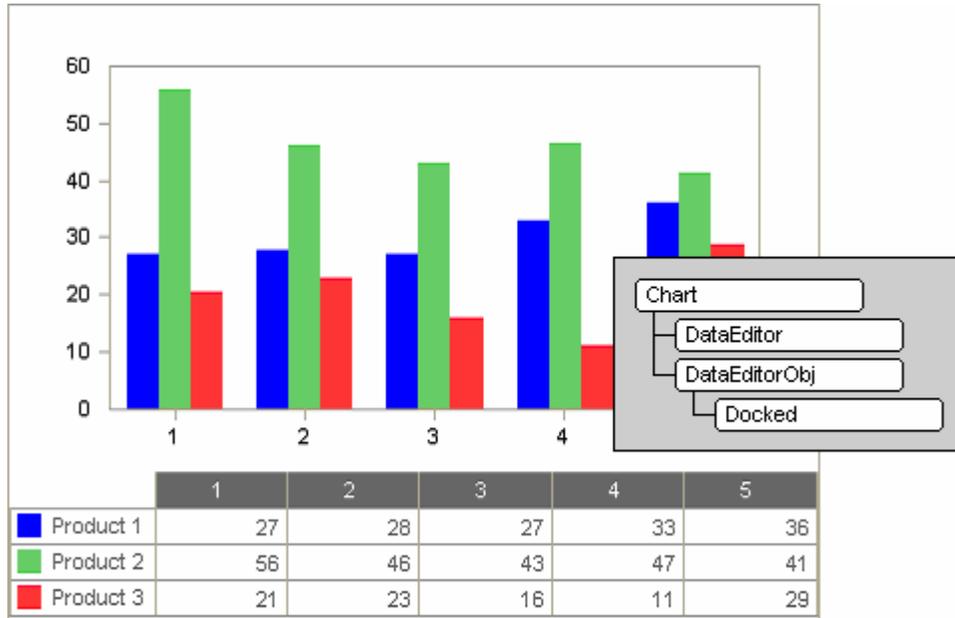
Chart FX provides a spreadsheet called Data Editor that allows users to see the data contained in the chart in tabular format. It also allows the user to modify any legend or value contained in the chart. Just like other Chart FX tools, the end user can control position and style attributes for the Data in the chart area.

Data Editor

Chart FX provides a spreadsheet called Data Editor that allows users to see the data contained in the chart in tabular format. It also allows the user to modify any legend or value contained in the chart (if allowed by the programmer).

Just like other Chart FX tools, the end user can control position and style attributes for the Data in the chart area.

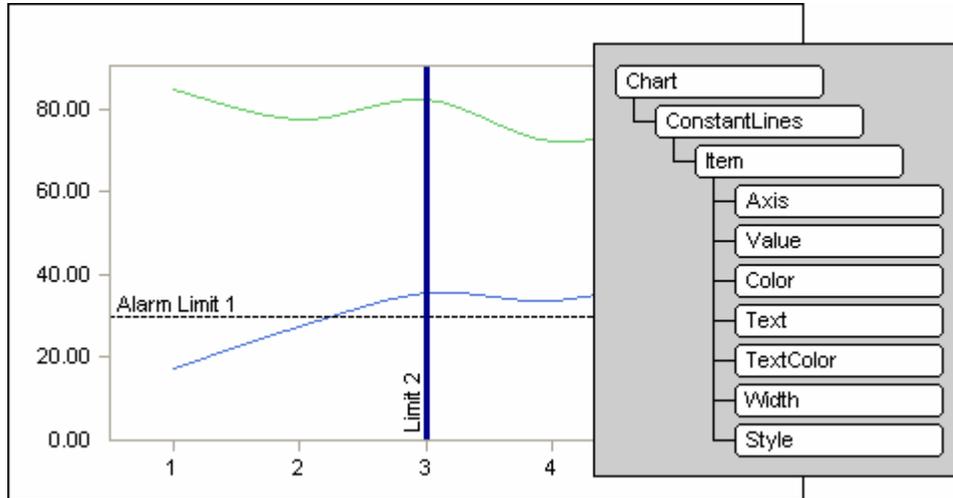
The **DataEditorObj** object provides the developer with control over the Data Editor. With this member you can control the position, color and other important aspects of this tool in the chart. For example, to show and position the Data Editor at the bottom of the chart, the following set of API calls is required:



Constant Lines

Constant lines are some of the most useful objects when it comes to highlighting information in the chart area. You may want to create constant lines to highlight limits or specific points of interest in the chart. For example, in a scientific application, you may want to use the constant line object to highlight an alarm limit; or in a financial application you may want to highlight a target price or date.

Constant lines are lines that you can draw anywhere in the chart area and associate them with a particular value in the axis that they're assigned to. The following figure includes the set of API calls required to create it:

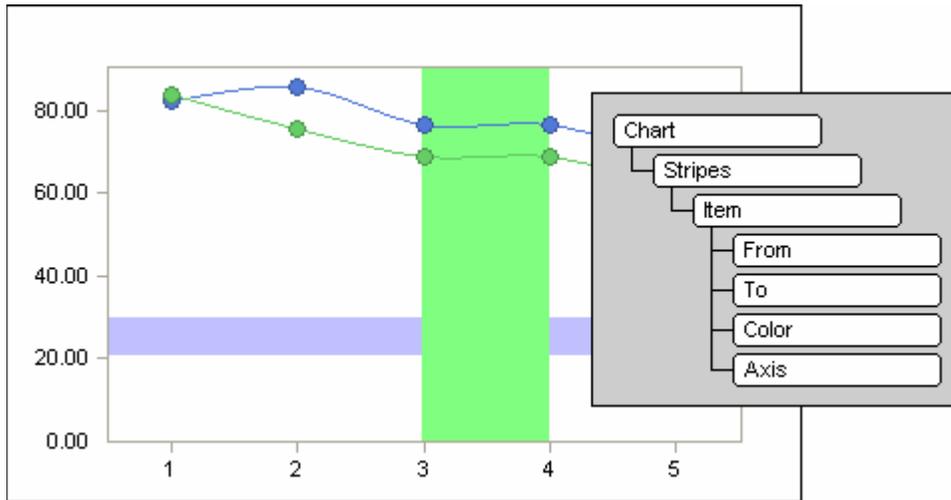


All constant lines are handled by the **Line** and **ConstantLine** objects and their supported members. Using the Line object, you can control the line color, styles and width. The ConstantLine object members also allow control over other attributes such as the associated axis.

Creating a constant line does not require a new data series and you can also configure labels and line styles, colors and widths. If you wish to highlight a range of values instead of a specific value, please refer to the Stripe feature next in this section.

Stripes

Stripes can also be very useful objects to highlight information in the chart area. They allow you to highlight a range of values associated with any of the axes by drawing a color frame in the chart background. For example, in a scientific application you may want the user to recognize points that plot between 20 and 80 with a blue stripe object as depicted in the following figure:



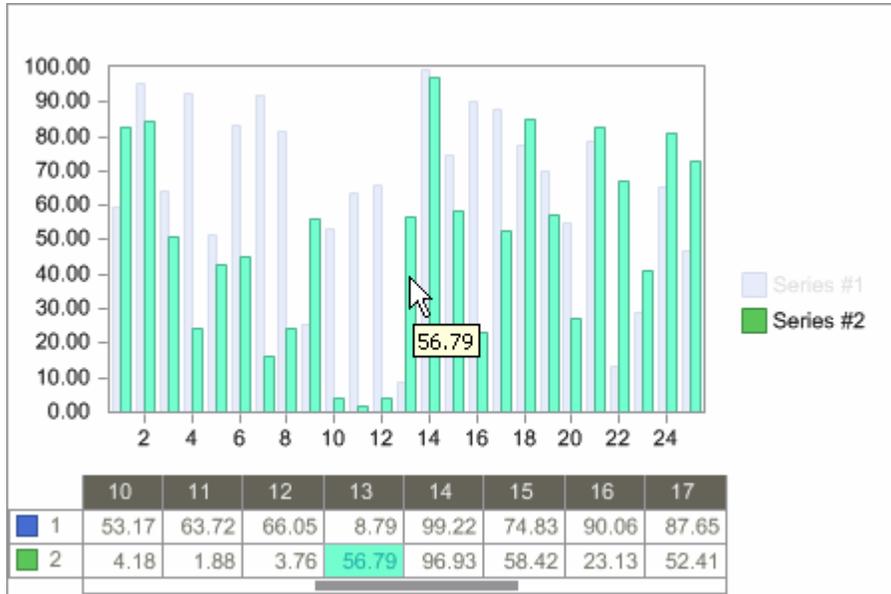
The Stripe object handles stripes and their properties, where you can set color, range and which axis the stripe is associated with.

In some cases, you may also want to highlight the corresponding range of values in the axis object. This can be achieved through the use of a Chart FX feature called Axis Sections, which includes the capabilities provided by the Stripe object. For further details, please refer to the Axis Section area of the Axes chapter in this book.

If you want to highlight a specific value instead of a range, please refer to the Constant Lines sample in the previous section.

Highlighting

Chart FX supports Highlighting features that improve the end users readability by allowing them to select chart elements in which to focus attention. This analytic tool allows users to easily discover values in the chart. There are several scenarios in which this is very useful, for example when a large number of points are visible in the chart area. Providing this feature in the chart allows end users to highlight and discover points. When an item is highlighted in a chart, the other Chart FX tools visible also reflect the action. In the following image, you can see how hovering over a particular point highlights the point's series, along with the respective entries in the Data Editor and the Legend:



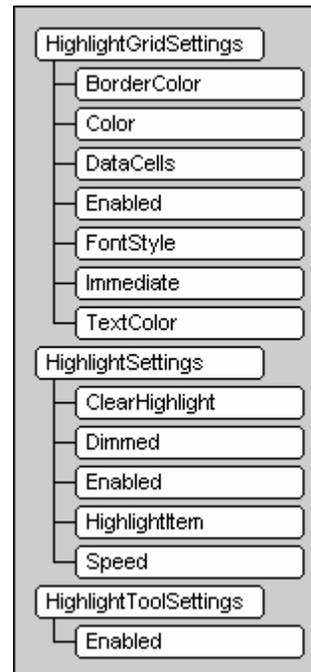
By default, when only a single series is plotted in a chart and a point is highlighted, that point will retain the "highlight" status. All other series will be dimmed while the highlighted point is slightly accentuated. When there are multiple series in a chart and a user highlights a series, the entire series is highlighted, while all other series are dimmed.

This default behavior may be easily customized; for example, instead of the default dimming action of non-selected points, developers could programmatically configure non-selected points not to change and only the selected or highlighted element display highlighted attributes.

Setting colors, styles, highlight speed, grid attributes, if enabled, and much more are all possible using the API shown in the image:

Note: The Highlight feature is ONLY functional when generating Client Control charts. When rendering images in your web applications, this feature is not available.

Using the chart API, you configure attributes for the chart when the **OnHighlight** event is raised. Please refer to the Chart FX electronic documentation for specific details and samples about the use of Chart FX's Highlight feature.



HighlightSettings object

This is the core object for the highlighting feature. Using the supported members, you can turn the highlight feature on and off (Enabled), highlight chart items programmatically, add additional code to highlight processing (using OnHighlight event), clear highlights, control how points will be painted when highlighted and much more. The **PointAttributes** property returns a PointAttributes object. Using this object, you can control how points will be displayed in the chart when highlighted.

HighlightGridSettings object

This object is used for customization of the Data Editor when a highlight event occurs. You can set colors for data cells, border colors, text color, etc.

HighlightToolSettings object

This object is used to instruct how the other chart tools will interpret a highlight event; whether the tool will be affected by a highlight event or not.

HighlightEventArgs object

This object supports the properties available to the developer during the OnHighlight Event.

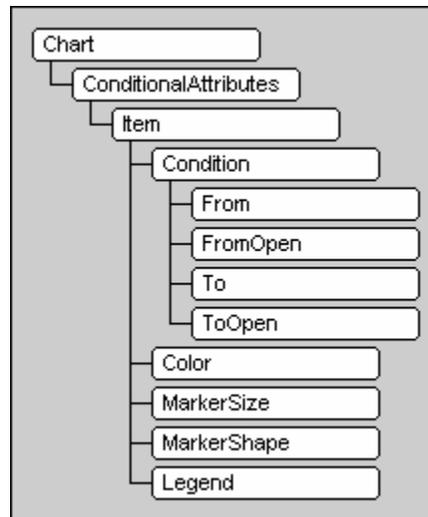
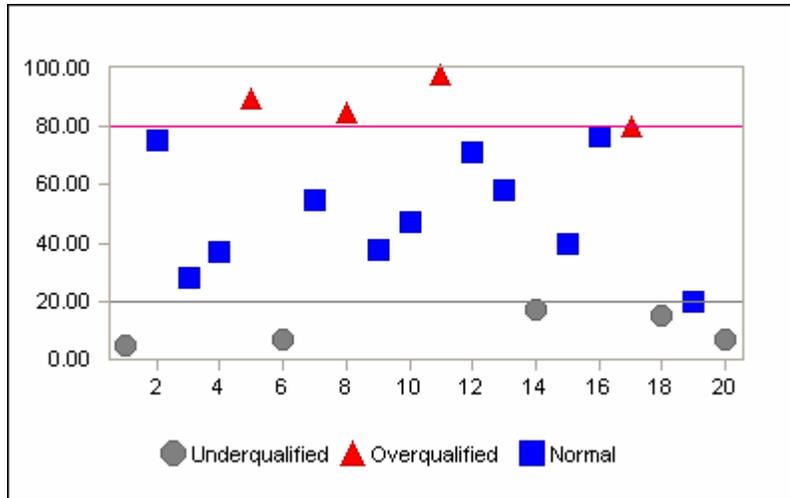
Note: For more information regarding these objects, please refer to the Chart FX API Reference.

Conditional Attributes

The Conditional Attributes feature of Chart FX provides a built in solution to configure point attributes for chart points that meet a configured condition. The value of each point is evaluated against the items in the **ConditionalAttributes** collection. When the point's value meets the conditions configured, that Conditional Attributes object would be assigned to the point; retaining the configured attributes.

The conditional attributes allow coloring points and setting marker size and shape independently without the need for creating complex algorithms to evaluate chart values. The ConditionalAttributes object derives from the PointAttributes object. For more information regarding the supported members of these objects, please consult the Chart FX API Reference.

In the following chart, different marker shapes and colors were configured to show the different ranges in a chart. The ranges are 0-20 (gray circle), 20-80 (blue square) and 80-100 (red triangle). Using 3 conditional attributes, one for each range, the points of the scatter plot were customized depending on their value. Below the chart appears the set of API calls required to obtain it:



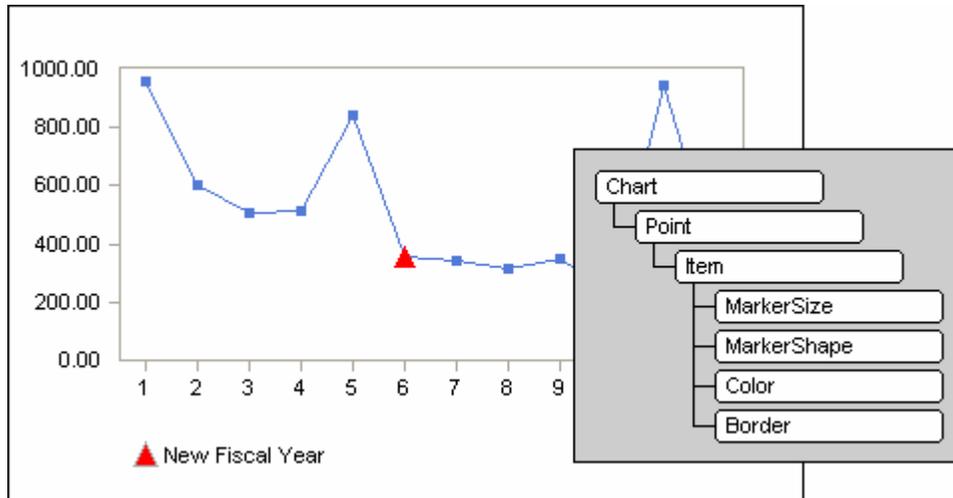
Per Point Attributes

In some cases, you can't use the ConditionalAttributes object to change the attributes of a particular point or set of points. For example, if you want to make the odd and even points look different, the ConditionalAttributes objects can't be used. For these situations exists the PointAttributes object.

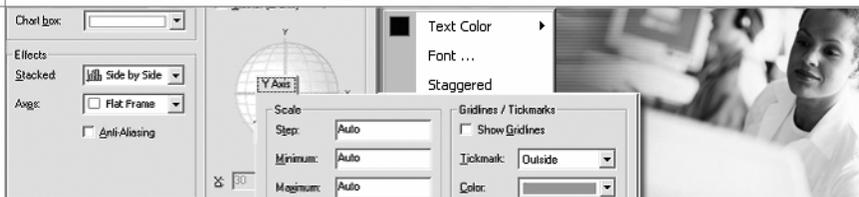
The PointAttributes object exposes many of the same properties available in the Chart object. Once again, giving the developer access to the point object increases the control over charts generated using Chart FX for .NET. The supported PointAttributes object members allow you to set border, color, marker, point label and scheme attributes, as well as many others, directly to a selected point included in the chart.

Setting per-point attributes is most useful for highlighting special points or groups of points within charts. By highlighting selected points in a chart, you can focus the attention of end users to specific information conveyed in the chart. Another useful tool when setting per-point attributes is the user legend box. The user legend box is configured exclusively by the developer and can act as a legend for these special points.

The following set of API calls sets the color to red and the marker shape to triangle, for a particular point:



End User Experience



Overview

Chart FX, unlike traditional charting applications, provides a rich set of features on the client side from a very basic image to a full featured active client with controllable menubars, context sensitive menus throughout, gallery choice, etc. This chapter focuses on highlighting these end user features in detail.

Introduction

Chart FX allows developers to create visually rich, highly interactive charts with data analysis capabilities that go far beyond simple tooltips and dialogs. This User Interface not only reveals important trends in the chart's underlying data but will also allow developers to build effective data visualization applications quickly and effortlessly with no complex code to write or maintain. For example, with a simple property setting developers can make charts scrollable, zoom in portions of the charts or even display a full data grid with the chart's underlying data. Before you commit to allow end users to interact with charts and enable any of the end user tools provided by Chart FX, it is important to take into consideration the platform you are developing for:

Chart FX and Client/Server applications

During the 90s, Client/server emerged as the most appropriate computing architecture in the business world. Client/Server usually refers to executables that run on Clients (PCs or workstations) providing local processing power. By running locally on a client computer, applications can take advantage of local resources at run time. Because they run locally, Client/Server applications target a specific platform or OS. This means conforming to User Interface guidelines and standards is an important aspect and challenge when developing applications that use third-party components. In this regard, Chart FX toolbars, menus and other UI elements and end user actions have been designed to conform to the Windows application UI and interaction design principles and guidelines. For this reason, you can enable or use any of the Chart FX end user tools in your Client/Server apps.

Chart FX and Web based applications

As opposed to Client/Server apps, Web based applications are restricted to run in a web browser and request HTML output from a remote web server. The problem with providing a rich UI for charts embedded in a browser lies in the fact that web applications are platform agnostic. This means, they normally use images to convey information. Unfortunately, end users will not be able to interact much with chart images (resulting in a very poor end user experience). To solve this problem, Chart FX provides platform specific (Windows) client side controls that allow full interactivity in the browser. When a browser accesses a page containing a chart, the Chart FX server is capable of producing a binary file (OLE) that can be read exclusively by our ActiveX or .NET client controls.

If you are developing a public web site that requires universal access you may be limited to work with chart images. However, if you are building an intranet or mission critical web based application that targets Microsoft's Internet Explorer, these client controls provide the following significant advantages:

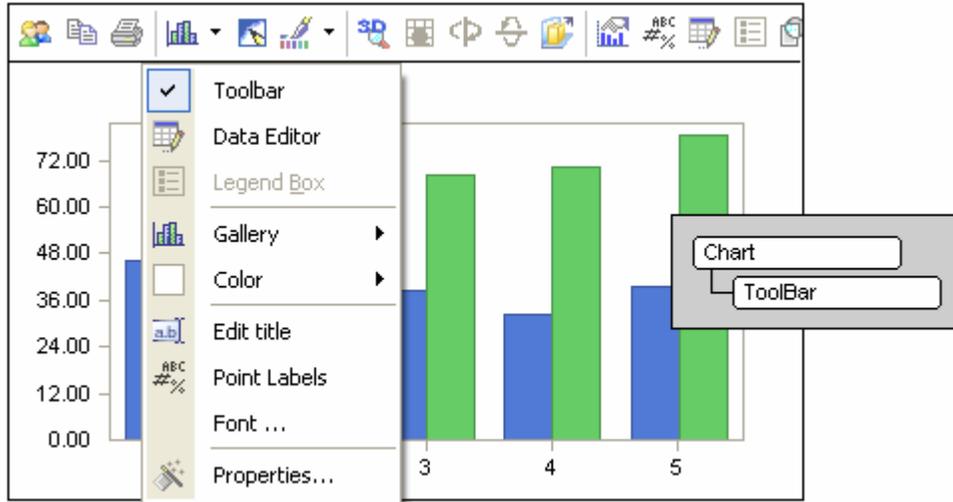
1. The use of client controls that provide a complex UI allow end users to customize the chart and perform additional data analysis without additional intervention from the developer or server.
2. Client controls allow load distribution on many of the complex calculations and paint routines that are required on the server, making the web application more suitable for performance and scalability. In other words, producing server-side images will consume processing power on the server thus possibly affecting performance and scalability.
3. In many situations, controls are mandatory since they use system related resources on the client machine. For example, repainting and scrolling a real-time chart requires the use of the client machine's GDI resources.
4. The only UI permissible in a browser are html forms which makes it impossible or impractical for certain features that client controls handle very easily through natural and intuitive interfaces like toolbars, context menus, tooltips and mouse clicks.
5. Client controls allow selective refreshing. In other words, when producing a server-side image, the whole page must be refreshed before the new image can be displayed on the browser. This is a major inconvenience. As the rest of the page elements that were not affected by the change must also be downloaded, making certain applications (real-time charts) impossible to achieve without the help of client controls.

Note: Updated versions of Chart FX support zero-footprint and fully interactive charts on the browser through the use of DHTML images. For additional information on version, availability and pricing please visit our web site at www.softwarefx.com or contact our sales department at sales@softwarefx.com.

Toolbar & Menubar

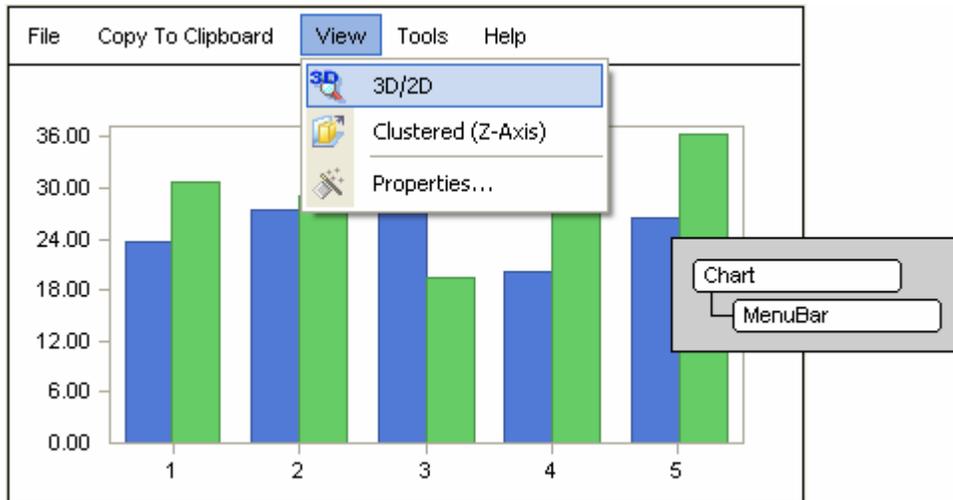
One of the most useful tools in Chart FX is the Toolbar; many developers appreciate this UI element because they are not required to add additional lines of code in their web based applications to give end users more freedom to customize their charts.

By default, the Chart FX Toolbar provides the most common options used by an end-user, such as switching from 2D/3D, color change, zooming & rotation and show/hide tools like data editor and legends. However, it can be completely customized by the developer according to application and user needs, by adding or removing buttons from the toolbar and even changing the tooltips displayed by the buttons.



Note: Please refer to the Chart FX Resource Center or online documentation for more information on how to properly add/remove buttons or otherwise customize the Chart FX default toolbar. By doing this, you will avoid adding unnecessary UI elements in your forms or HTML pages to manipulate chart specific settings.

The Chart FX Menu Bar provides the same options as those found in the toolbar. The Menu Bar is often utilized when the application already has a toolbar displayed. The following picture depicts a chart with a Menu Bar

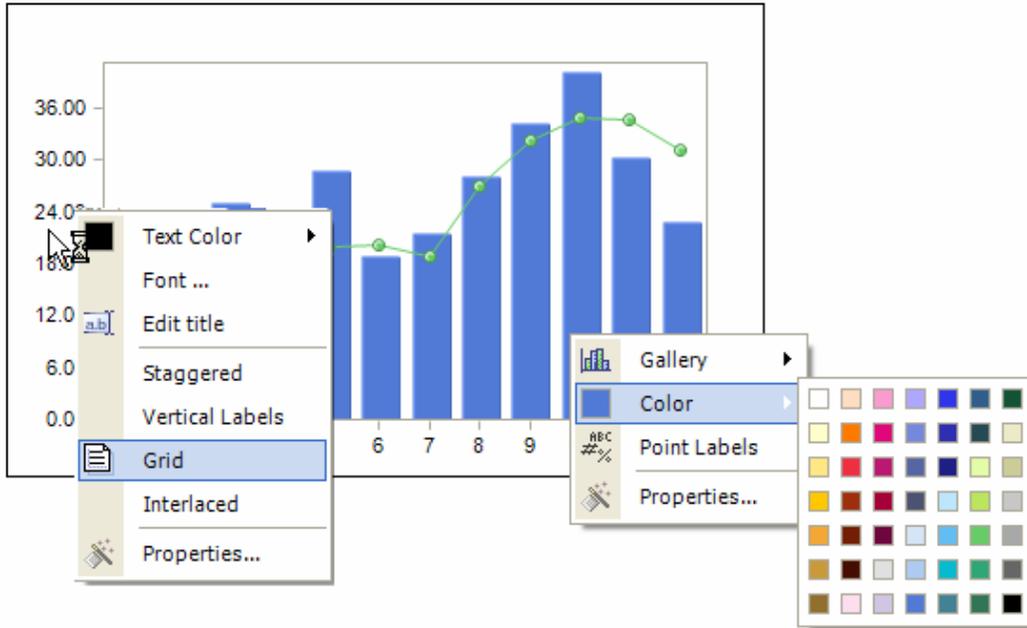


Context Menus

While the Toolbar and Menu Bar provides one-click access to many chart options, the user may find it difficult to access specific features such as axis and point customization. For that purpose, the Chart FX client components provide a comprehensive set of context sensitive UI menus and dialog boxes that allow the end user to customize many of the options of the chart by right-clicking on specific elements and using the Context Sensitive User Interface.

Depending on where on the chart the end user presses the right-button, a menu will show up with the features inherent to that specific element. Chart FX provides UI menus for customizing general chart features, titles, axis, each particular series, legends & data editor, constant lines and stripes.

The following picture depicts some of the elements that support context sensitive menus:

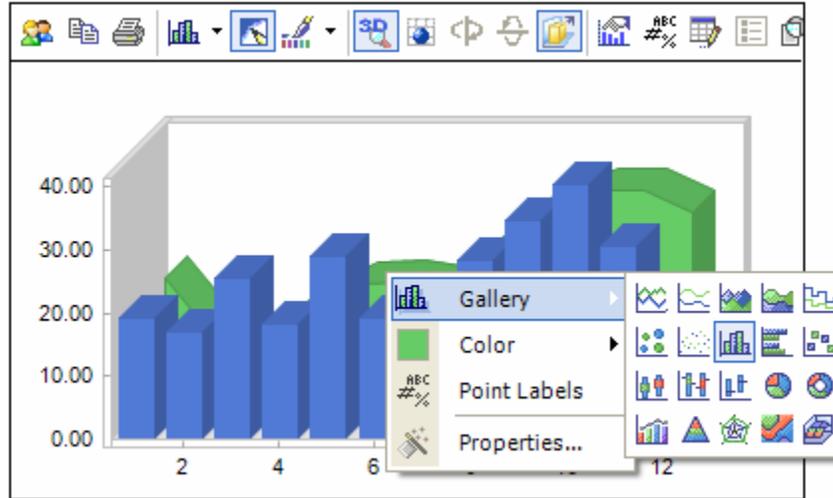


Some of the UI menus provide a Properties option for further customization. This option will bring up a properties page dialog box that allows customizing of general chart features, series and axes. Depending on whether the end user selected a particular series or axis, the dialog box will change accordingly.

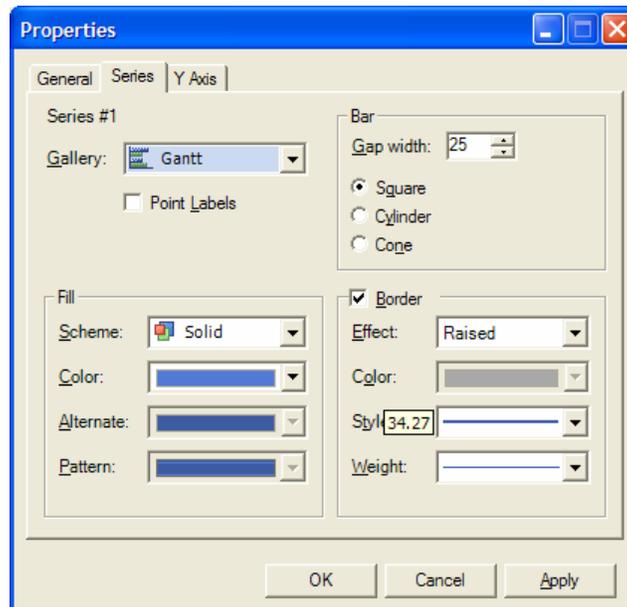
Changing Series and Point Visual Attributes

While in the process of analyzing data, end users may want to change series colors, add labels or even change the gallery setting for a particular data series. The Chart FX UI, including the Toolbar, Context Sensitive menus and dialogs provide an easy way to change visual attributes of any chart element.

In the following picture, a chart data series has been right clicked and a context sensitive menu has been displayed allowing the user to change gallery, colors and other important visual attributes of that particular series.

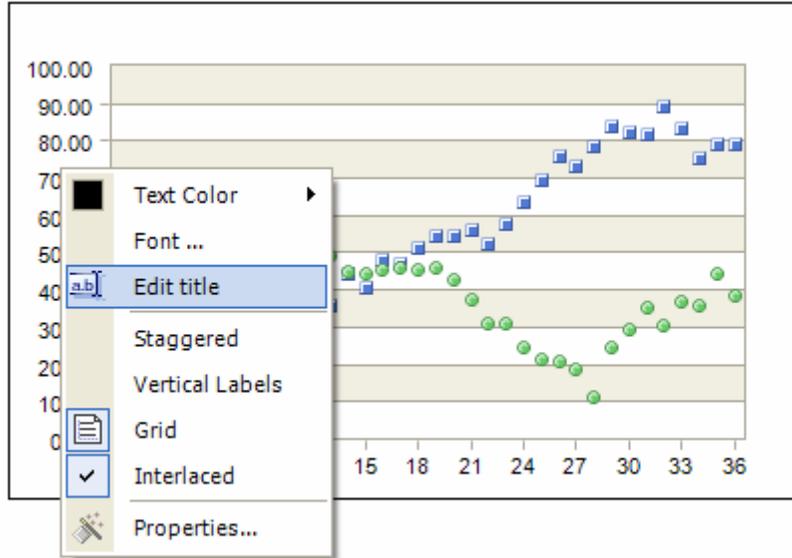


Additionally, end users may select the Properties option to access a dialog that provides additional customization capabilities

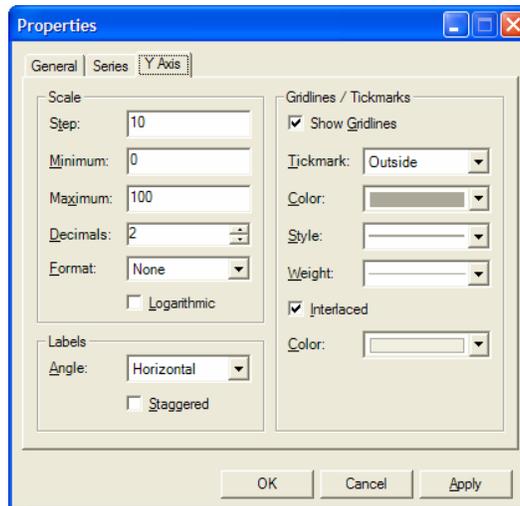


Changing Axis Settings

Although axes settings are automatically chosen by Chart FX or manipulated in code by developers, end users may want to change axis administrative settings such as maximum, minimum, gridlines, tickmarks, label formats, decimals, labels, fonts and titles, among others. For example, for analysis purposes end users may want to change a linear scale for a logarithmic scale when handling large values in a chart. Chart FX allows end users to access most axis settings by right clicking the labels on a particular axis and using a Context sensitive menu or a property dialog. In the following picture, the chart's main y axis has been right clicked and a context sensitive menu has been displayed allowing the user to change basic settings such as gridlines and colors.

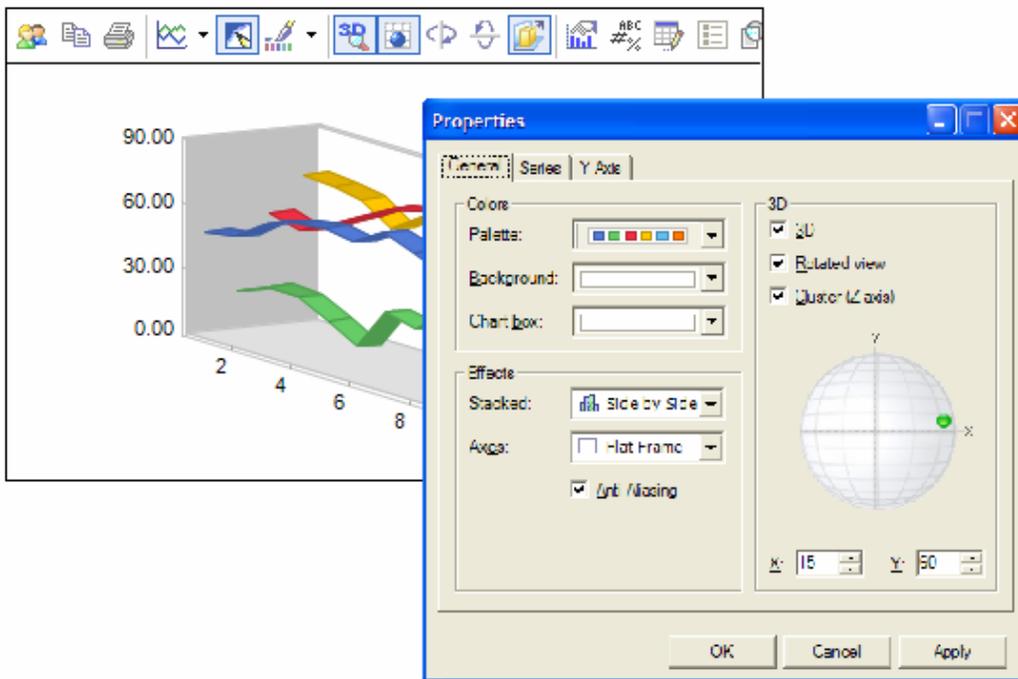


Additionally, end users may select the Properties option to access a dialog that provides additional customization capabilities:



3D Rotation and Perspective

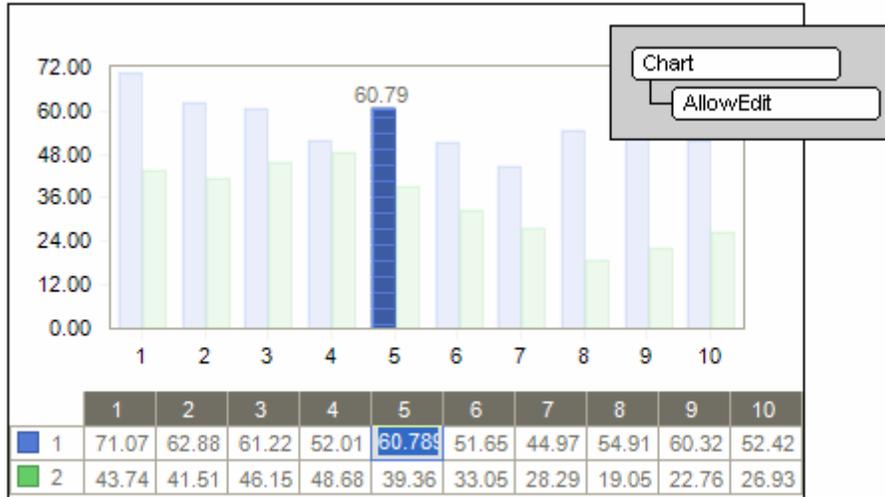
Another nice feature of the Chart FX Client components is the ability to change the rotation angle and perspective of a 3D chart. It can be done directly from the toolbar or from a dialog box for more accuracy.



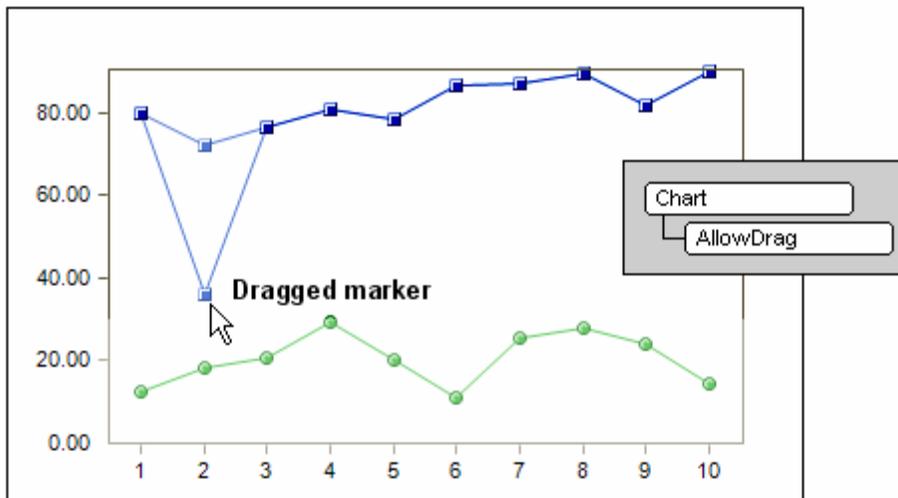
Changing Underlying Data Using the Data Editor or Marker Dragging

Although Chart FX is mostly used as data visualization (read only) solution, there are ways end users can edit or change the underlying data in the chart. As a developer you must decide whether these changes should be reflected in the data source as Chart FX will simply trigger an event (**DataChangebyUser** event) that you can capture in your application to further process these changes in the data (For example, update the data source).

The first technique involves enabling the Data Editor to accept changes in the data. This can be done by enabling the **AllowEdit** property. The end user will simply double click a value in its corresponding cell and the Data Editor will allow editing. After the value is changed, the chart will immediately repaint reflecting the change made by the end user.



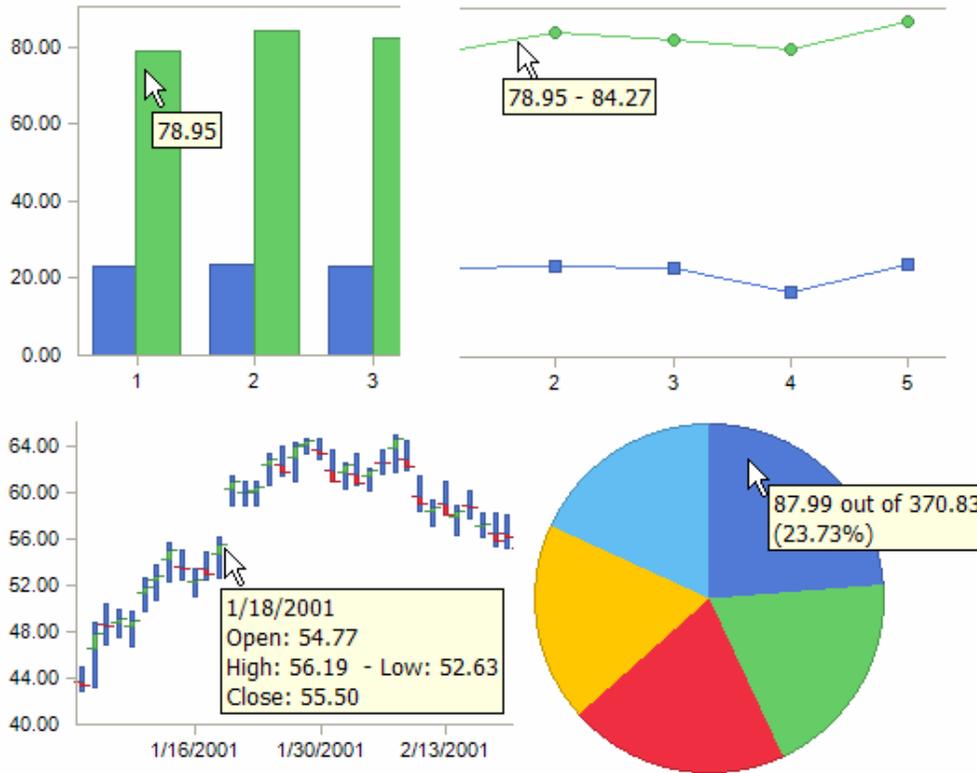
Another option to allow the end user to change the underlying data in the chart is by enabling the **AllowDrag** property. When set, the end user is now capable of changing a value by selecting a chart marker with the mouse and while pressing the mouse left button drag the marker to a desired value. This technique is visually appealing and in some cases (like an XY chart) allows end users to change two values at the same time. Similarly, as the end user drags the point to its desired value, the **DataChangebyUser** event will trigger so you can process these changes programmatically. This process is depicted in the following figure:



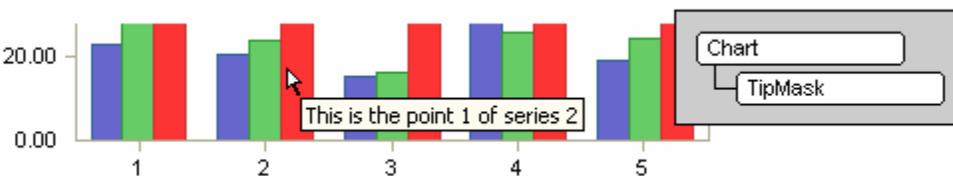
Tooltips

Tooltips appear when the user hovers the mouse over a chart element and are used to expose additional information about it. This is especially important in charts with many points, as the tooltips provide a means for discerning values in the chart.

Chart FX provides smart tooltips; which change according to the gallery type being displayed. For example, in a simple bar chart, the tooltip will show the bar value, while in a financial chart the tooltip will show the High, Low and Close values. Similarly, in a pie chart, Chart FX's tooltip will show not only the value for a particular slice, but also its percentage of the total pie, as depicted in the following figure:



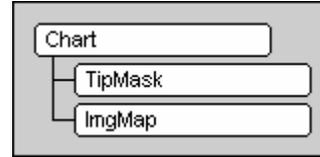
Tooltips can also be customized through the use of a mask. The following image shows an example of a tooltip customization:



Note: In web based applications, chart images will also show tooltips. For further details about the customization of the tooltips in image charts, refer to the Drilldown section later in this chapter.

Showing Tooltips in Chart Images

In web based applications that generate chart images, it is also possible to display tooltips. Firstly, the **TipMask** property allows you to configure the text to be shown when the user positions the mouse over a particular element in the chart which belongs to the image map. Finally you must set the **ImgMap** property which allows you to control how the tooltip will appear; it takes one of the following settings:



TitleTip

If you are displaying your charts on Internet Explorer 3.0 and above, the tooltip will be displayed with the contents of the TITLE attribute. Therefore, when you set the **ImgMap** property to this setting, it will show the tooltip in Internet Explorer. This feature is supported by modern browsers as well.

MouseTip

If you want to achieve browser independence for displaying tooltips, you can set the **ImgMap** property to this setting which allows you to invoke a JavaScript to show the Tooltip on any browser. It will add the following function calls to the AREA tag:

```
onmouseover="popupon('<ToolTip>',event)"
```

This function will be called when the mouse passes over the Image Map area.

```
onmouseout="popufoff()"
```

This function will be called when the mouse abandons the Image Map area.

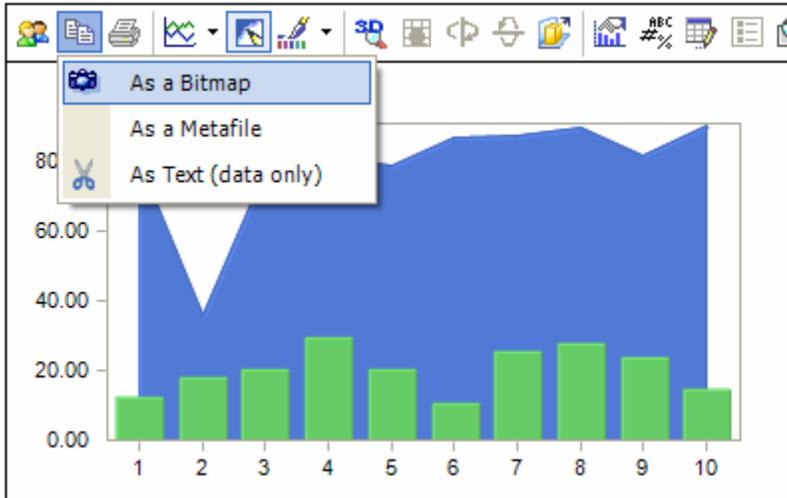
You will be responsible for the implementation of these functions. For further details on how to implement browser independent tooltips, refer to the electronic documentation.

Note: Tooltips in chart images only applies to Web based applications.

Exporting Charts to Other Productivity Applications

Perhaps one of the most functional aspects of a data visualization solution is when end users are capable of detecting important trends in the data and use mechanisms to convey that information to other people. For example, corporate executive may want to create a document, presentation or email with a chart containing important information he discovered while using a mission application.

The Chart FX UI allows end users to quickly and effectively export images and data available in chart; this functionality can be easily accessed through the Chart FX toolbar, as depicted in the following figure:



The Chart FX MS Office Add-in

Unfortunately, when chart images are copied and pasted in a Word document or a PowerPoint presentation, the Chart FX UI is lost and no additional analysis capabilities are available within the productivity application. The Chart FX MS Office Add-in is a FREE software that uses an OLE Server capable of producing interactive charts in any MS Office application.

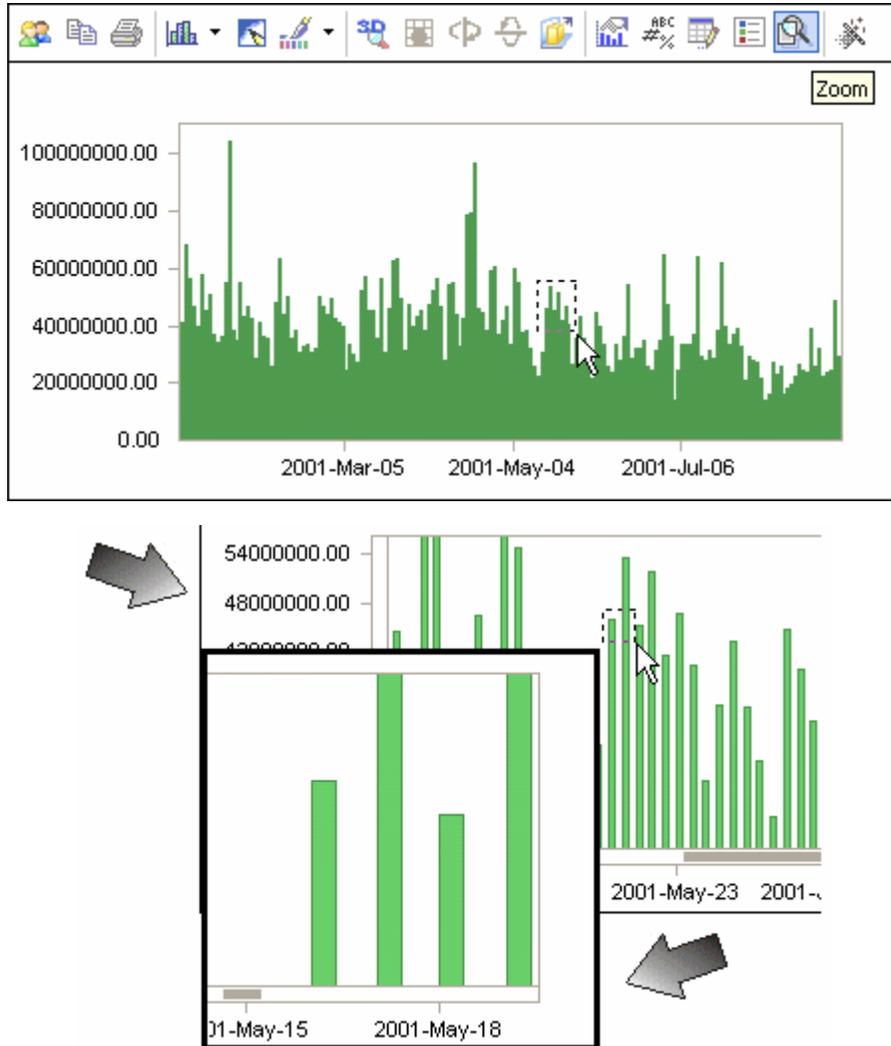
When the Chart FX MS Office Add-in is installed an additional option ("Copy as Object") will appear in the Chart FX export button in the toolbar. The "Copy as Object" option will allow your end users to paste charts as interactive charts that display the Chart FX toolbar, and other UI described in this manual, even outside the context of your application.

For additional information or to download the Chart FX MS Office Add-in please visit our web site at www.softwarefx.com/extensions.

Zooming

The Chart FX toolbar provides a button for zooming which is useful when details are hidden by large amounts of data. By simply selecting the area of interest, the end-user will be able to zoom in to view details. When zooming is applied, the boundaries of the chart will change and scrolling will be enabled for access to other areas of the chart. The zoom tool can be used repeatedly in succession until the user reaches the desired zoom level.

This process is depicted in the following figure:



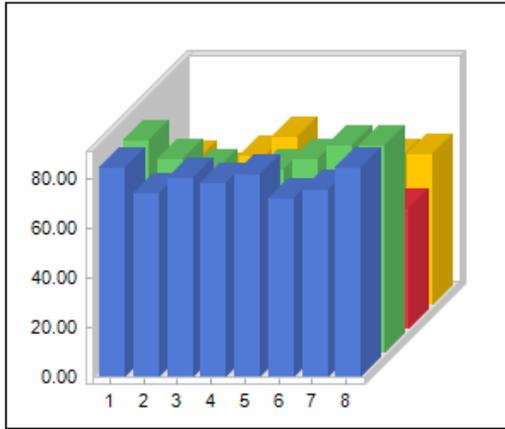
Note: Zooming is an end user action that can be only accessed from the chart's toolbar.

Highlighting

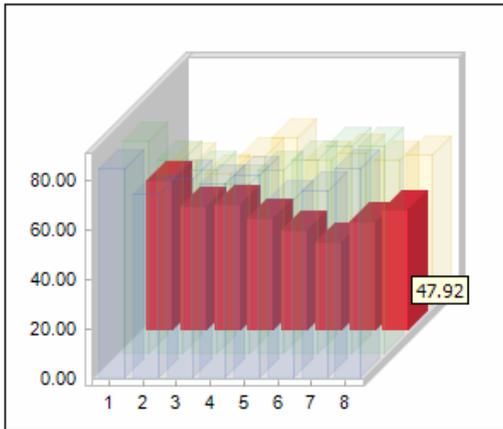
Chart FX supports Highlighting features that improve end users readability by allowing them to select chart elements in which to focus attention. This analytic tool allows users to easily discover values in the chart by simply hovering the mouse over existing data series or points. When the highlighting activates other data series will be dimmed allowing end users to clearly see data related to that particular point. This effect is particularly useful when the chart has many data series or when some data series overlap others.

The following pictures depict a 3D clustered bar chart where some series overlap others, with highlighting it easier to discover the overlapped series as others become transparent.

Normal Chart

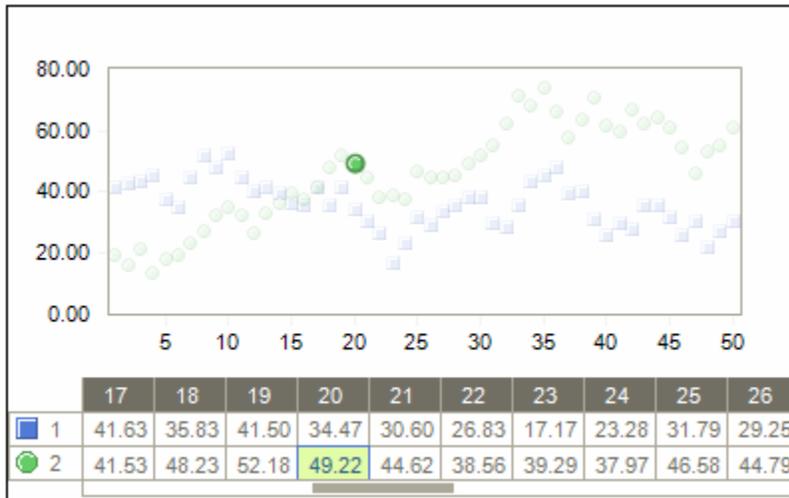


Highlighted data series



Using the DataEditor and Legend boxes for Highlighting purposes

Besides interacting with data series in the chart area, the Highlighting effect also works when hovering the mouse over legend boxes and cells in the Chart FX Data Editor. Highlighting over these elements is particularly useful to locate data points in a chart that contains a large number of points. For example, the following picture depicts Highlighting over a particular cell in the data editor:

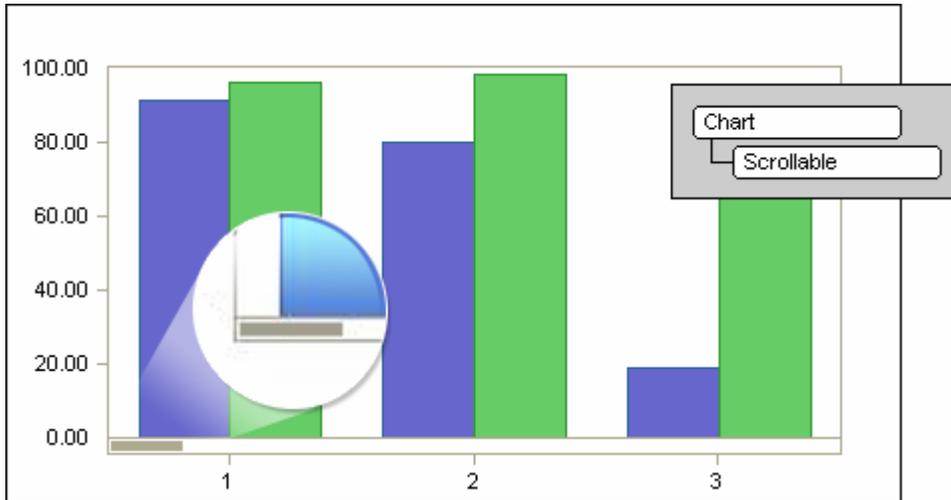


Scrolling

Although Chart FX provides a wealth of properties to control how labels are displayed on axes, the number of data points and the chart size become important factors as to how many labels could fit along an axis or how much data is actually visible. In other words, some charts simply have too much data for the desired chart size and little can be done to improve the chart's readability.

Chart FX provides an elegantly designed scroll bar, which attached to an axis, allows end users to zoom in and view portions of an entire chart. All axes in Chart FX support scrolling.

To enable scrolling you must enable the Chart object scrollable property using the following API call:



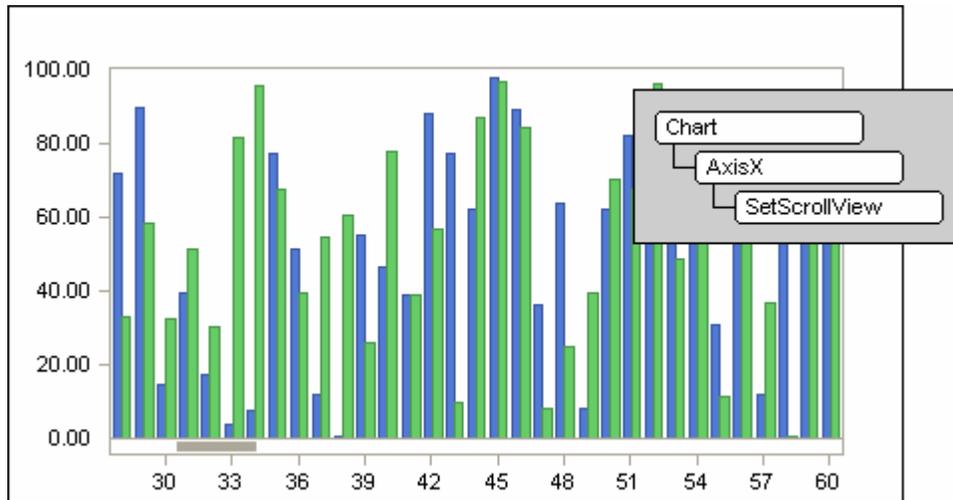
The scroll bar will appear docked right next to the axis labels as depicted in the figure. The user can now drag the scroll bar to a desired position in the chart.

By using the scroll bar, the data points will not be crunched inside the chart area and most points can show their respective labels. This enhances the chart's readability and allows the end user to view portions of the chart at a time.

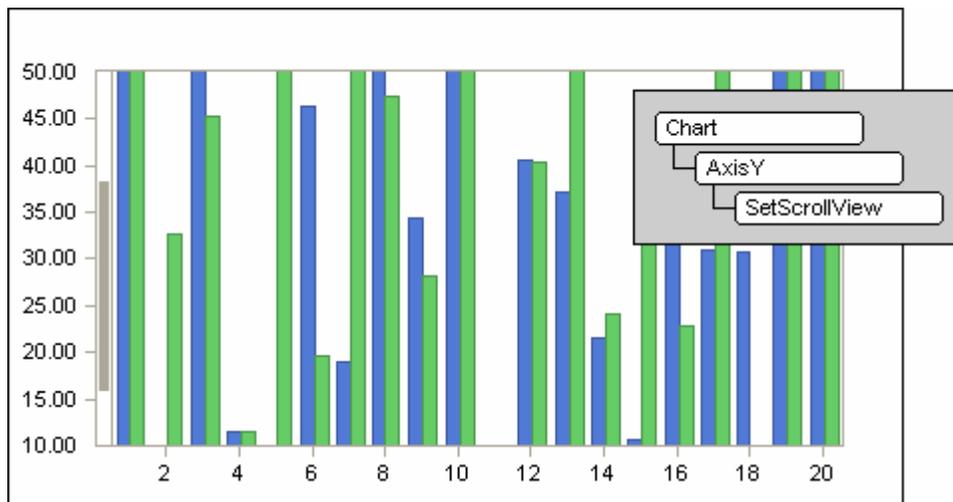
Controlling the Scroll Position Programmatically

Although the scroll bar is an end user tool, you may want to control the number of points visible in a specific chart. This can easily be done with the **SetScrollView** method supported by the Axis object.

For a categorical axis, this method receives the index of the points you want to see in the chart. For example, if you have 300 points and you want to force Chart FX to show points 30 to 60 in the categorical axis, you can set the **SetScrollView** method as follows:



For a numerical axis, the **SetScrollView** method receives the values to be shown in a specific scroll view. For example, if your numerical axis is scaled between 0 and 300 and you want to set a scroll view between 10 and 50, you can set the **SetScrollView** method as follows:



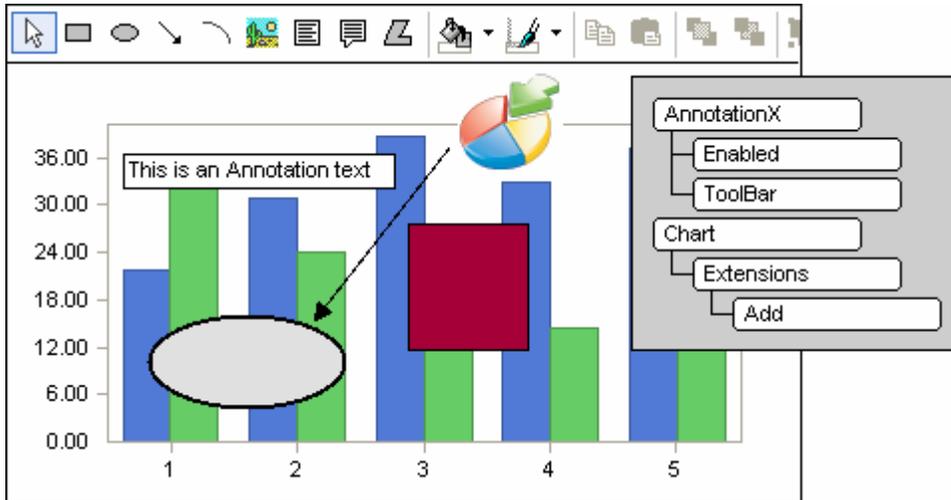
Similarly, Chart FX will dispatch an event called **UserScroll** every time the user interacts with the axis scroll bar. This event is useful if you need to synchronize several charts as the user interacts with a scroll bar.

Refer to the Chart FX API Reference for more information on the **UserScroll** event supported by Chart FX.

Annotation

Chart FX provides an Annotation Extension to allow the inclusion of floating objects in the chart. The Annotation extension supports a toolbar so end users can create objects and highlight different areas in the chart without additional programming efforts. This means end users will be able to add text, arrows, geometrical shapes, and even images they can freely move around to highlights specific data points or sections in the chart area. End users can also rotate, group and even flip annotation objects.

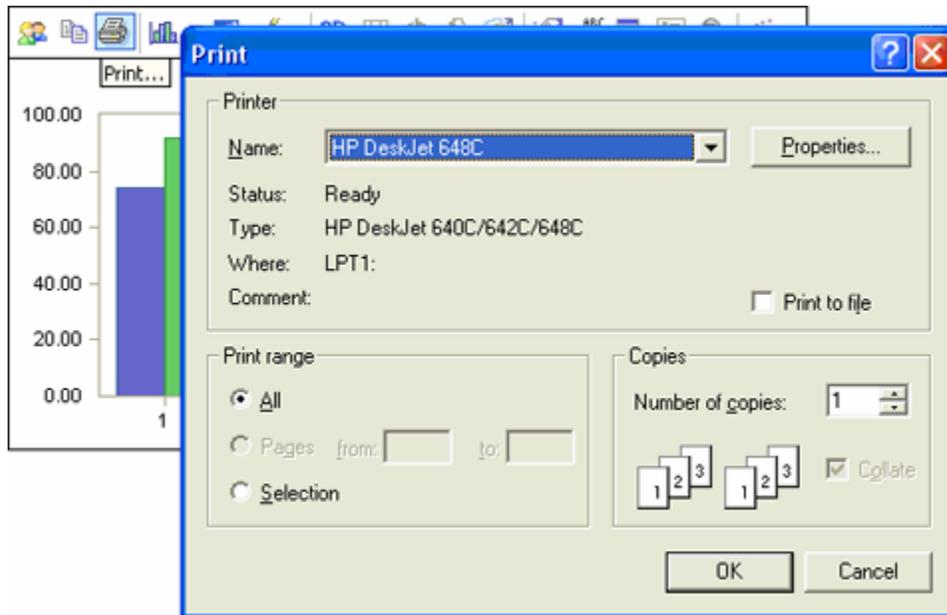
In order to enable to the annotation toolbar, you must create an annotation extension object, set the **Enabled** property to "True", add the newly created object to the chart and set the **ToolBar** to True. Below is an example of the annotation toolbar and some objects, as well as the required set of API calls to enable it:



The Annotation extension also exposes an Object Model that you can use to add and manipulate any annotation object programmatically. This is particularly useful when you want to draw anything on top of a chart and that is not supported using the API exposed by Chart FX. For example, if you want to add a company logo to any section in the chart you can easily use a picture object available in the Annotation extension. For additional information on examples using the Annotation extension programmatically, please refer to the Chart FX Resource Center.

Printing

The end user is able to print a hardcopy of the chart within the application. Chart FX provides an integrated printing mechanism available to the end user through the Toolbar.



The MenuBar also exposes Print Preview option as seen in the following images:

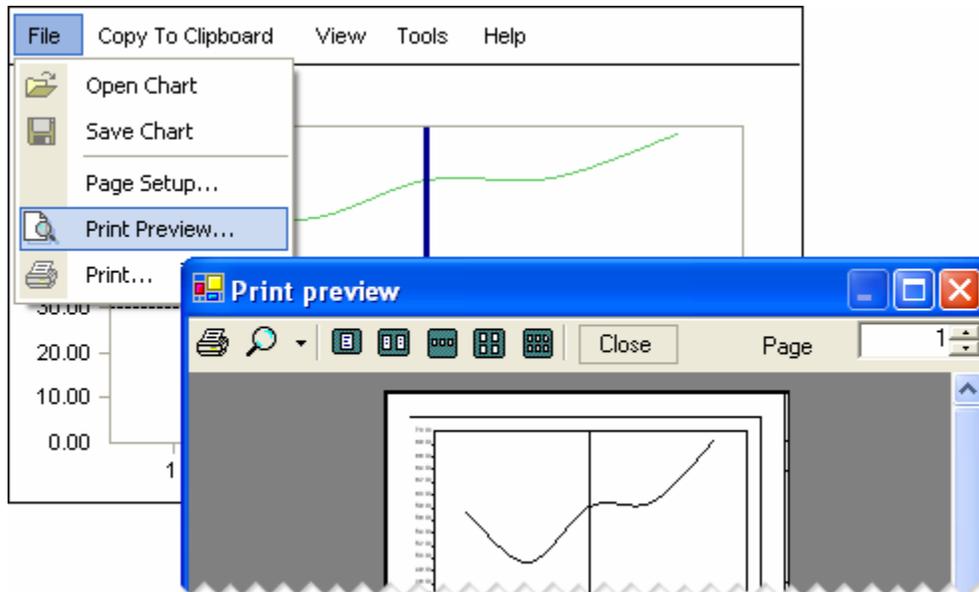


Chart FX also exposes the **Printer** object for a high level of customization. For details about the use of the Printer object, refer to the online documentation.

Personalization

One of the inherent problems when allowing end users to change visual attributes like colors, 3D effects, chart galleries and other non-data dependent attribute is that when charts are reloaded, scripts or code associated with the chart will apply default settings reversing any changes end users may have done at run time. This could be annoying, especially if end users invested sufficient time and effort customizing charts to their preferences.

Although saving end user preferences on the server is an option, this usually posts major maintenance and coding efforts as developers will need to identify and load user chart preferences as they access your application or web site.

Please note that saving charts, including its underlying data, may not be a viable option since end user simply want to reapply non-data dependent attributes such as colors chart types but they want to see up-to-date and dynamic data when they use the application or access the web page in future sessions.

Chart FX elegantly solves this problem through Personalization. A process by which Chart FX automatically saves ALL non-data dependant attributes in an end user local machine. Personalization requires each chart in the application to be uniquely identified so when the chart is reloaded, it will look for local Personalization settings after the new data has been loaded. Because this process takes place after the chart has been fully serialized, end user preferences persist over any setting in the code.

Additionally, because Personalization does not save any data dependant attribute, the resulting chart contains new data with end user's preferred visual attributes.

Personalization can be implemented automatically by the developer or it can be left to end users to decide whether they want to save chart's personalized preferences.

Personalization from the end user's perspective

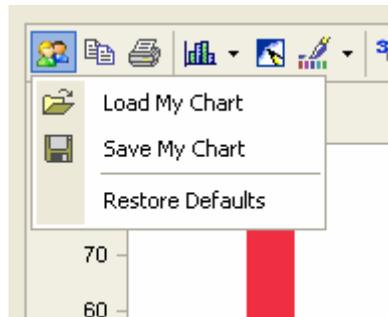
One of the options of the Chart FX toolbar is the **Personalized Charts** icon. It provides end users with 3 options:

Load My Chart: when the user clicks on this option, any previously saved transformations will be applied to the new data. These transformations have been saved through either UI or API. If there are no transformations saved, no changes will be applied to the chart.

Save My Chart: when it's clicked, all the transformations made by the end-user to the original chart will be saved in a small file (approx. 1K) in the user's local hard drive. For further details on where the chart is saved, refer to the electronic documentation.

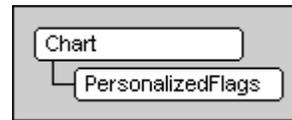
Reset Defaults: this option clears all the transformations made by the end-user and resets the chart to the original look established in the application. By default this option is disabled but it will be enabled as soon as the end-user modifies any attribute of the chart or a saved chart is loaded.

Important note: In .NET, Chart FX uses Isolated Storage to achieve Personalization. This means, there are no security issues associated with this technique.



Personalizing a Chart using the API

When chart personalization is applied through the UI, user intervention is required to save and apply the transformations. In some cases, the developer may want to provide an automatic mechanism to save and apply those changes. Chart FX provides this through the **PersonalizedFlags** property. With this property, the developer has the ability to decide whether the changes will be saved and/or loaded automatically.



The supported flags are:

PersonalizedFlags.AutoLoad: Before the chart is rendered, Chart FX Client control will look for a saved transformation file in the end user's Isolated Storage, and will apply it to the new data if found. This functions as if the end user selected the "Load My Chart" option. By default this option is disabled.

PersonalizedFlags.AutoSave: Any transformations made to the chart will be automatically saved to the end user's Isolated Storage when the chart is destroyed (the browser is refreshed or closed, or another page is invoked). By default this option is disabled.

PersonalizedFlags.EnableRestore: As soon as the end user makes the first change to the chart through the UI, the original chart is saved in memory to allow the Reset Defaults option. If you don't want to provide such option to the end-user, you may avoid the process of saving the chart in memory. This can be accomplished by turning off the **EnableRestore** flag (enabled by default). You must use the bitwise combination of the flags and property, using the following API call:

For further details about how Chart FX prevents interference among Personalized Chart settings saved from different applications or web sites, refer to the Chart FX electronic documentation.

Drilldown

Charts are an excellent visual navigational tool. Imagine delivering a system or application that puts summary level information into a chart and lets end users click on chart elements to drill down. End users can dive into the details, instantly create other graphs, or view news stories or abstracts that pertain to selected chart elements or markers.

Unfortunately, all this can't happen without the developer's intervention. In other words, you must instruct Chart FX what to do or where to go when an end user clicks certain chart elements. For this reason, there's basic programming you must do in order to achieve drilldown. Also, because of architectural reasons, drilldown is implemented differently in Web applications and Client/Server or desktop applications.

Implementing Drilldown in Client/Server applications

Implement drilldown features in your Client/Server or Windows forms application, involves capturing mouse events, such as click, double click, etc. Chart FX provides ample control over mouse events. For more details, refer to the Capturing Mouse Events section of the Chart FX Advanced Features chapter of this book.

Implementing Drilldown in Web based applications

The fact that charts are embedded in a browser posts limitations as to the amount of client code you can write in web page to achieve drilldown. Additionally, charts are usually displayed as images in the browser which presents limitations when interacting with charts. Chart FX not only supports drilldown capabilities but it also allows you to:

- Generate Image Maps when PNG or JPEG Images are being generated. This allows users to drill down even when charts are generated as images.

- Pass parameters from the referrer URL, for example, when a user clicks on a particular series you can decide to pass parameters like value, series and other related chart information. This takes only a property setting and allows you to customize the new page using information coming directly from the chart.

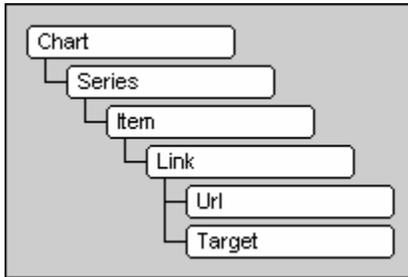
- Customize the drill down capabilities. If you want to redirect the request when the user clicks on different chart objects, such as points, series, title, stripes and constant lines, you can easily customize this setting.

- Reload charts without reloading the entire page. This creates a seamless storyboard effect as the page will not be reloaded when a chart is invoked in the same bounding rectangle.

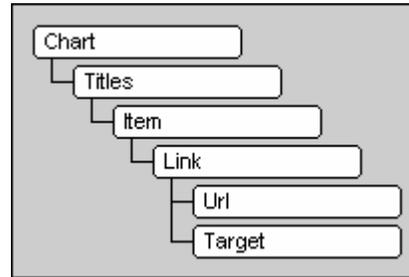
The Drilldown API for web based applications

Essentially, all Chart FX elements expose a **Links** property that allows the developer to enable drill down capabilities on Series, Points, Axes, Stripes, Titles, Constant Lines and any other chart element.

The Links is a complex object containing not only the string of the URL to be invoked when a user clicks on the objects, but also the target. For example, the object models below present Link object for different chart elements:



For a Series object:



For a Titles object:

When specifying a link value for an inner object, it will override the link value from the chart main object. For example, if the **Chart.Link** value was set to `http://www.softwarefx.com`, then all markers will be linkable to our web site. If you want, you can also specify the value, series or index of a particular point as a parameter of the URL property in the Link object. For further details, refer to the API documentation or samples provided in the Chart FX Resource Center.

Image Maps Generation

Chart FX is able to generate Image Maps for charts generated as an image. Usually, Image Maps are used for providing drill down capabilities from image charts. Also, image maps are useful for showing tooltips in image charts, even though you may not be using the drill down capabilities.

When the Chart FX Server component is producing an image and the drill down capabilities are enabled in the code, Chart FX will generate a client-side Map with the location of the different markers in the resulting image.

These maps may be quite large depending on the number of points and series in the chart. Therefore, generating Image Maps could represent a work overload on your server. The number of lines in the image map depends on the number of markers in your chart, e.g. a 3D bar chart with 2 series and 10 points will generate an image map with $2 \times 10 \times 3$ objects. (3 polygons are required to paint a 3D bar chart)

The resulting image map has the following characteristics:

The resulting image map is embedded in the resulting page. External image maps are not supported. They are client-side image map. This means the browser is the one responsible for processing the mouse position. You can control how to generate the Title attribute in the AREA tags with the `ImgMap` property.

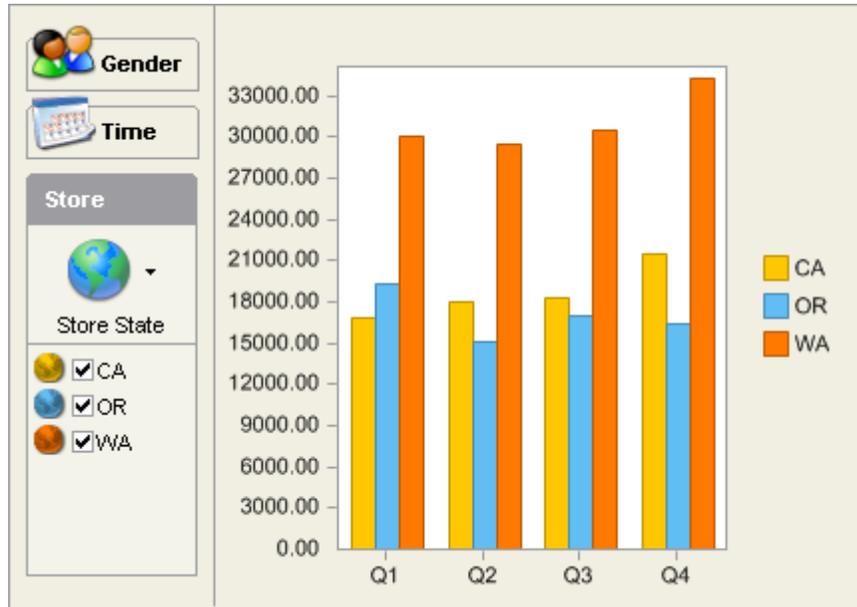
Note: Image Maps are not supported for Curve charts or when you are using the Bit Stream chart generation capability of the Chart FX server-side component.

BI Capabilities, Slicing & Dicing Data (Chart FX OLAP)

Perhaps one of the most significant UI enhancements Chart FX provides is when plotting data from Multidimensional or OLAP data sources. In Business Intelligent applications, much of the application is made up of a User Interface that enables end users to pivot, slice and dice data to uncover crucial information.

Chart FX OLAP picks up the information and presents the hierarchical structure of an OLAP cube in a comprehensive, easy to use Interface. All this end user functionality is provided with little or no intervention from the developer, thus reducing development time and increasing customer satisfaction.

The following picture depicts a sample bar chart with the Chart FX OLAP UI:



Note: Chart FX OLAP is sold separately. For additional information on this product, including pricing and availability, please visit the Appendix section of this manual or visit our web site at www.softwarefx.com/extensions.

Additional Extensions and End User Experience

Just like Chart FX OLAP, other Chart FX Extension provide their own user experience. For example:

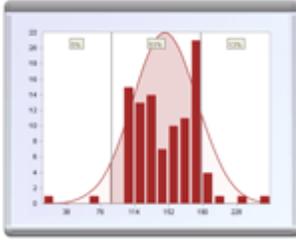


Chart FX Statistical allows end users to apply different formulas, view properties and observe relationships between studies. Opacity and color-coding features have also been included to improve readability of the statistical results graphically.

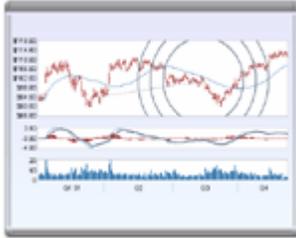


Chart FX Financial provide end users a fully functional User Interface to add/remove a wealth of Technical Analysis indicators for stock market data.

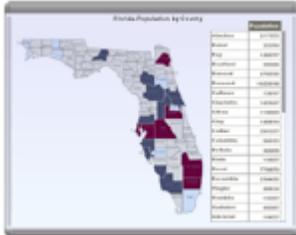
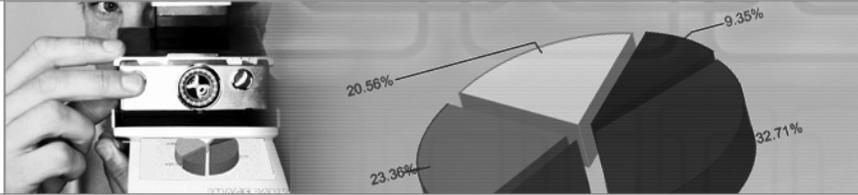


Chart FX Maps allows end users to zoom in and drill down to more detailed maps.

Controlling the Chart FX Output



Overview

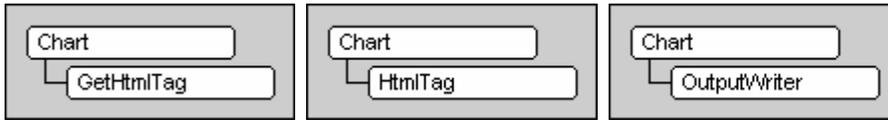
Although Chart FX was designed to take full advantage of the browser's capability to display active content such as a .NET object, many developers want to stay away from client-side components and prefer to force image generation regardless of the browser or OS supported by the client. Chart FX is even more powerful as it provides the the ability to generate Image Maps allowing users to interact by clicking on particular chart elements that have been defined as links within code. Therefore, chart images are no longer static data interpretations, but actually may be used to create entire applications based on the graphs created by the Chart FX Server component.

Selecting the Chart's Output

There are several different ways to control the chart's output from the Chart FX server control. One, is to simply allow the browser detection feature generate the appropriate format suitable for the requesting client.

You may also instruct Chart FX to render graphs in a variety of formats. Configuring the chart to do so may also be accomplished several different ways.

As a server control, Chart FX supports a method called **GetHtmlTag** that allows the developer to select, among other things, the type of desired output for your charts. By default, Chart FX will generate a PNG image of the chart if you do not specify the type, and call the GetHtmlTag method. The **HtmlTag** and **OutputWriter** properties may also be used to render charts in the desired format:



Note: The OutputWriter property is normally reserved to configure external output writers for a chart.

What Image Formats can Chart FX Generate?

Examining the GetHtmlTag definition carefully, you'll notice that PNG, JPEG, SVG and Flash are all possible parameters (instead of simply "Image"). This means Chart FX is capable of generating any of these file formats. The SVG and Flash formats require adding a reference to appropriate image writer library in your project. Once a reference has been added, you may then configure the **HtmlTag**, **GetHtmlTag** or **OutputWriter** member accordingly. With extensibility in mind, future output writers may always be added as future requirements demand.

.NET Client Controls

Format: .NET format.

Comments: Component is required on the browser.

Full managed code, no signatures are required.

Interactivity: Full interactivity (including user toolbar).

Accessibility: Windows .NET clients only.

Generating charts as .NET Client Controls allows users to interact with the chart through a variety of tools and context menus. This chart format allows the end users to customize the visual attributes as well as modify data values directly in the chart. In order for client computers to view these types of charts, the .NET Framework must be installed as well as the appropriate .NET security settings applied for the zone of the chart that is being accessed.

When setting the GetHtmlTag or HtmlTag members to "Auto", Chart FX will use environment variables received from the client to determine if this format is supported. If so, the .NET Client control will be used to view the chart requested.

PNG Images

Format: Raster format.

Comments: Best image format for producing charts.

Interactivity: Provides limited interactivity through image maps.

Accessibility: Supported in most current browsers.

The PNG algorithm is particularly effective for drawing vector images (such as charts), as images will experience no quality degradation or color dithering. This format came as a response to Unisys prohibiting the public distribution of the GIF generation algorithm. This format is even better in terms of quality and compression than GIF and supports many more features, such as transparency, interlaced, 24-bit palettes and many other advantages over GIF images.

This means, PNG is the preferred image format for Chart FX. However, the drawback is that many older browsers do not support this image format (As a matter of fact, only browsers 4.x and above can display PNG images). Therefore, the JPEG format is also available for these older browsers.

JPEG Images

Format: Raster format.

Comments: Slightly faster than PNG, however charts are not as crisp.

Interactivity: No interactivity.

Accessibility: Universal access.

This format was created to compress and display photo quality images (not vector images such as charts). As a result, displaying charts using this algorithm is not advisable as you will obtain quality degradation and color dithering when creating JPEG images.

When you set the third parameter of the GetHtmlTag method to "Image" you are basically instructing Chart FX to determine the most appropriate image format. This is particularly important as PNG images will be generated whenever possible achieving best results in the majority of browsers. Whenever a PNG image can't be generated, Chart FX will generate a JPEG image.

Additional Output Writers

The SVG and Flash writers are additional writers supported by Chart FX. With this extensible methodology, other output writers may also be added in the future. Please see the related sections later in this chapter.

SVG

Format: Vector format.

Pros: Enhances performance significantly; files are small and painted on the client.

Cons: Limited interactivity.

Accessibility: Accessible from many platforms.

Flash

Format: Flash Macromedia format.

Pros: Charts are downloaded and viewed using a widely available 3rd party Flash viewer at the client.

Cons: Limited interactivity.

Accessibility: Accessible from many platforms.

Accessibility

Format: Renders chart as text.

Pros: Comply with the Section 508 of the Rehabilitation Act. Many accessibility browsers can read chart information to a user.

Cons: Limited chart interpretation.

Accessibility: Requires supported Accessibility tools (Browser).

Note: Due to Microsoft's decision to change the security settings for .NET components (Search for "Trustworthy Computing" at the Microsoft site), Software FX, Inc. has decided Chart FX should generate an image by default instead of a .NET component. You can easily adopt the .NET component generation behavior by changing the third parameter of the GetHtmlTag call to ".NET" or "Auto" or change the HtmlTag property at design time on the Web Forms project.

Forcing the Chart FX Server to Generate Images

In order to force Chart FX to always generate images, you must configure the **GetHtmlTag** method and prevent the automatic Browser Detection capability. By modifying the third parameter of the GetHtmlTag method to "Image", the Chart FX Server component will always generate images no matter the browser being used to access the page. Once again, this is also possible using the **HtmlTag** property:

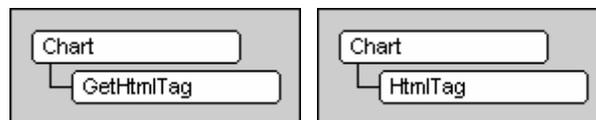


Image Map Generation

Chart FX supports Image Map generation when PNG or JPEG files types are rendered. When configured to generate chart objects, image maps are not required. Usually, Image Maps are used for providing drill down capabilities; however, they can also be useful for showing tooltips in chart images as well.

By extending this functionality, Chart FX does not compromise your original site functionality even when you are restricted to providing users sites free of active client-side component features.

Before you read this topic, please make sure you understand how to setup drill down capabilities in web based charts. For more information please refer to the **Drilldown Capabilities** section later in this chapter.

Client Side Map (Area Tags)

When the Chart FX Server component is producing an image and the drill down capabilities are enabled in the code, Chart FX will generate a client-side Map with the location of the different chart elements in the resulting image.

These maps may be quite large depending on the number of points and series present in a chart. Therefore, you must be aware that generating Image Maps could represent a heavier load on your server. Please note that the number of lines in the image map depends on the number of elements in your chart, e.g. a 3D bar chart with 2 series and 10 points will generate an image map with $2 \times 10 \times 3$ objects. (3 polygons are required to paint a 3D bar chart).

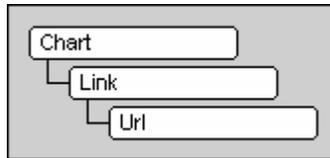
The resulting image map has the following characteristics:

The map is embedded in the resulting page. External image maps may be obtained using the **GetHtmlDataEx** method. The Image Map is client side. Meaning the browser is responsible for processing the mouse position. The Title attribute of the <AREA> tags may be modified using the **ImgMap** property.

Note: Image Maps are not supported for Curve charts or when you are using the Bit Stream chart generation capability of the Chart FX server-side component (May be obtained using the GetHtmlDataEx method).

How are parameters appended to the actual link in the image map?

The answer to this question can be found in the Drilldown Capabilities section. Using special flags in combination with the flag wildcard (%), you may configure parameters embedded into the url string that will be updated based on the specific element selected by a user:



Note: Please see the programmer's guide of the resource center for more details concerning the flags supported in the **Url** string properties.

Note: Chart FX handles escaping its own strings (+ instead of space). Therefore, you don't have to worry about encoding the urls. In other words, when invoking a page with parameters that contains blank spaces you must replace blank spaces with + signs; this can be done through an IIS function called URLEncode. Chart FX, however, does this for you, in case you're passing a string that contains blanks.

Can I control tooltips that are shown in the chart?

Absolutely! The **TipMask** property allows you to configure the text that is to be shown when the user positions the mouse over a particular element in the chart that belongs to the image map. However, because browsers support tooltips in different ways, Chart FX provides the **ImgMap** property which allows you to control how the tooltip will appear.

ImgMap.TitleTip

If you are displaying your charts on Internet Explorer 3.0 and above, the tooltip will be displayed with the contents of the TITLE attribute. Therefore, when you set the ImgMap property to this setting it will show the tooltip in Internet Explorer. This feature is not supported by Netscape but looks great in Internet Explorer.

ImgMap.MouseTip

If you want to achieve browser independence for displaying tooltips you can set the ImgMap property to this setting which allows you to invoke a JavaScript to show the Tooltip on any browser; adding function calls to the AREA tag in the resulting Html file. In the following client-side example, this function would be called when the user's mouse passes over the corresponding Image Map area:

```
onmouseover="popupon('<ToolTip>',event)"
```

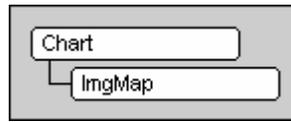
This function will be called when the mouse abandons the Image Map area:

```
onmouseout="popupoff()"
```

Note: Additional settings are supported using the ImgMap property. Please see the API reference in the Resource Center of your Chart FX installation.

You will be responsible for the implementation of these functions. For further details on how to implement browser independent tooltips see the programmer's guide section in the Resource Center of your installation.

The following diagram depicts the general API used to configure Image map functionality in a chart image:



Note: Annotation Tools also support access to the Link object and may include a corresponding Image Map. This way you can not only link using chart elements but any annotation object in your chart.

Additional Output Writers

SVG Generation

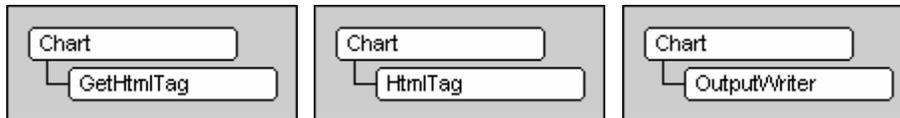
One of the most useful features of Chart FX is its versatility to render charts in different formats, in order to take advantage of the browser capability to display active content. One of the formats supported by Chart FX is SVG (Scalable Vector Graphics).

Based on XML (Extensible Markup Language), SVG is a graphics format that will allow web designers to add high-quality interactive vector graphics and animation to web pages using only plain text commands. SVG enables web developers and designers to create dynamically generated graphics from real-time data with precise structural and visual control. With this powerful new technology, developers can create a new generation of web applications utilizing data-driven, interactive, and personalized graphics.

Chart FX can provide drill down charts using the SVG format, in the same way that is done with Image Maps. For further details on how to implement drill down using SVG and/or Image Maps, please refer to the Image Maps Generation section earlier in this manual.

SVG is still a Proposed Recommendation and additional information and status is available at the official SVG site: <http://www.w3.org/Graphics/SVG/Overview.html>.

The syntax required for rendering the chart as an SVG may be done using the HtmlTag, GetHtmlTag or OutputWriter chart members:



An SVG reader is required on the client in order to view the chart in SVG format. If no SVG reader is found on the client, Chart FX will instruct the browser to download a reader from the URL specified in the **SVGDownload** tag of the configuration ChartFX.Internet.config file. By default, the download URL is <http://www.adobe.com/svg/viewer/install/>.

Flash Generation

One of the most useful features of Chart FX is its versatility to render charts in different formats, in order to take advantage of the browser capability to display active content. One of the formats supported by Chart FX is Macromedia Flash.

Macromedia Flash Player is one of the most widely viewable rich client technology to deploy content and applications. It implements vector-based content and applications download faster than their bitmap equivalents. Streaming data content appears immediately, without having to wait for the entire piece to download.

Chart FX can provide Drill down charts using the Flash format, in the same way that is done with Image Maps. For further details on how to implement Drill Down using Flash, SVG and/or Image Maps, please refer to the Image Maps Generation section earlier in this help file.

The chart may be rendered in this format as well using the HtmlTag, GetHtmlTag or OutputWriter chart members.

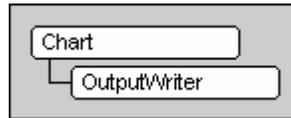
Accessibility Output

In 1998, Congress amended the Rehabilitation Act to require Federal agencies to make their electronic and information technology accessible to people with disabilities. Inaccessible technology interferes with an individual's ability to obtain and use information quickly and easily. Section 508 was enacted to eliminate barriers in information technology, to make available new opportunities for people with disabilities, and to encourage development of technologies that will help achieve these goals.

Chart FX includes an library that is used to provide an alternate text version of any chart generated. Chart FX has been developed with the idea that end users of charts will consist of a very diverse group. This feature also allows web sites using Chart FX as a charting engine to comply with the Section 508 of the Rehabilitation Act (<http://www.section508.gov>).

The accessibility writer works as an enhancer Chart FX and is only available with Server based applications. An enhancer writer is a Chart FX writer that is used to complement a standard writer, such as PNG, JPG, SVG or Flash.

To generate this output format, you must add a reference to the Accessibility library in your project. Once included, you may create an accessibility writer object and assign it as the chart **OutputWriter** to generate the text version of the chart. The API below is used to generate the default alternate text.



Once you have created the ChartText object, you may set attributes to customize the alternate text generated. Some of these properties include WriteOrder, OutputMode, WriteMask, PreviousText, PostText, ShortPause, LongPause, FontFace, FontSize and FontColor. Please see the API reference for details.

Chart Export

Chart FX provides the Export method that enables you (or your end users) to save chart files, images and data in a variety of formats. Some of the formats available differ from platform to platform, as only web based products (i.e. Chart FX WebForms) will generate web formats (i.e. PNG or JPEG).

There are also other file types in Chart FX called "Chart Templates" that allow you to save the appearance (Colors, Chart Types and Styles, Visible Tools, etc.) of the chart in a file so you can later apply it to another chart. This allows you to reuse code instead of setting the same attributes for all of your individual charts.

Note: Templates save all the information that is not data related. This means, it will not save the values, number of series or points, or any other property related to this data, such as MultiType settings and Min, Max values in the chart axes.

File Formats

The following file formats are supported by the Export method:

Binary: Exports and imports chart settings to a binary file.

BinaryTemplate: Exports chart settings to a binary file with 'Template' pre-assigned, using the **FileMask** property.

XML: Exports chart settings to an XML file.

Text: Exports chart data to a Text File.

Bitmap: Exports the chart as a Windows Bitmap.

Metafile: Exports the chart as a Windows Metafile.

JPEG: Exports the charts as a JPEG image (only supported in web server versions).

PNG: Exports the chart as a PNG image (only supported on web server version).

External: Uses an external encoder to export the file (only supported on Web Forms version).

The Export Method

This method is used to export the chart data and/or set properties to the clipboard, file or stream in different formats. Check the API reference in your Resource Center for details on the different overloads supported by this member.



The External File Format

The External file format is only supported by Chart FX Server based products and provides support for other formats, using external encoders. In order to use the External format, the chart's ImageTags property must be loaded with the format to be exported:



The FileMask Property

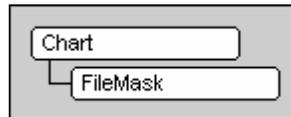
Whether you are saving chart files in Binary or XML format, you can control what is saved in the file by manipulating the **FileMask** property. The Binary format will save visual attributes and data; XML will save only visual attributes.

This property is particularly useful because it allows you to customize how chart files are saved. This way, you can build your own chart templates. By default, this property is set to save all visual attributes including the data.

However, if you want to save everything but the titles in the chart, this could be done by simply excluding the Titles using this mask property. Please see your API Reference in the Resource Center of your installation for details on using this property.

This property is a mask property, which means all settings represent a bit in the word that you need to turn on or off accordingly. For this purpose you'll use the bitwise operators (And, Or, Not, Xor) provided by your development tool. Because it is a mask property, you must make sure you use these operators to turn on/off bits and avoid losing previous settings to the property.

Please see the API Reference of your Chart FX Resource Center for samples and details on using the FileMask property.



Advanced Features



Overview

Charts are most often used to depict data for analysis. To help end-users convert the data into useful information, effective charting tools must do more than show such data in a nice graphical way. They must provide a superior end-user experience in terms of how the user analyzes such data and discovers important information embedded within it.

Real Time Charts

Chart FX supports true real time charting capabilities by providing specific functions that support chart scrolling in an accelerated painting mode (without flickering). To prevent data overflow, Chart FX provides real time charts with a pre-defined maximum point buffer. This allows the library to accept points up to a buffer value; once the buffer is full, a new value will be inserted into the chart while the initial value set is lost. Chart FX supports two different types of real time charts: **Limited** and **Unlimited** Real-Time Charts.

Limited Real Time

A limited real time chart will display only a pre-defined number of points at any one time (previously allocated buffer). This type of chart is the fastest real time chart available in Chart FX since memory is allocated only once (setting **BufferSize** property). Therefore, the chart can only display this number of points at any one time (i.e. setting 15 as the BufferSize means you will lose point 1 when passing point 16 to the chart).

If losing previously passed points is not important, we suggest using this type of chart for real-time purposes when implementing a rapid data input rate. Two variations of this kind of chart are also available in Chart FX:

Standard: When the buffer is full (BufferSize has been reached) and a new point is inserted, the data will "scroll". This means you will lose the first point plotted and the nth point will become the nth-1.

Loop Position: Same as standard but every time you set a new value, a customizable vertical line will pass through the point being updated (the last acquired point). When reaching the end of the data set, the looping marker will move to the beginning.

Unlimited Real-Time Charts

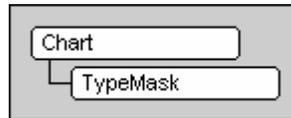
This type of real time chart will continue to add points without losing any previously set values. You may also choose to set the chart to scroll every time it receives a new point. Depending on the context of your application, this allows you to set automatic scrolling real-time charts.

Note: Real time functionality is only supported for windows based (Desktop) development.

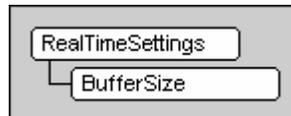
Passing Data to Real-Time Charts

Below are the steps used to create and pass data to a real time chart:

You will need to set the **EvenSpacing** flag of the **TypeMask** property. This may be set at design time or run time to ensure the points will be equally distanced in the X Axis.



Set the **BufferSize** property to a positive integer value to represent the real time charts point buffer if you want to create a Limited Real-Time Chart (strongly suggested!). This will allow the points in the chart to be plotted much faster.



Set the real time **Style** property to select the real time style you want to be applied to the chart. The different styles allow you to implement a loop marker, as well as, hide the default hourglass icon.



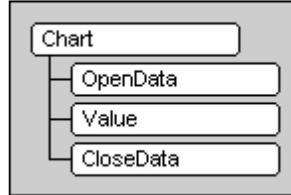
Open the communication channel (**OpenData** Method) to the real-time chart using the 'AddPoints' Or 'InsertPoints' flags combined with 'Values' enumeration (see explanation later in this section). This will allow the pointers to the data array to be relative to the last point added, so there is no need to remember the number of points the chart currently has or the index of the last point passed.

Set the corresponding value of the new points using an offset instead of an absolute index to the points. Therefore, if you want to add two (2) new points to the chart (before the **CloseData** method) you must use index 0 and 1 for the point index in the **Value** property.

Call the CloseData method with a combination of the 'Values' flag with any of the following constants:

RealTime - Chart FX will not scroll to the end of the data set.

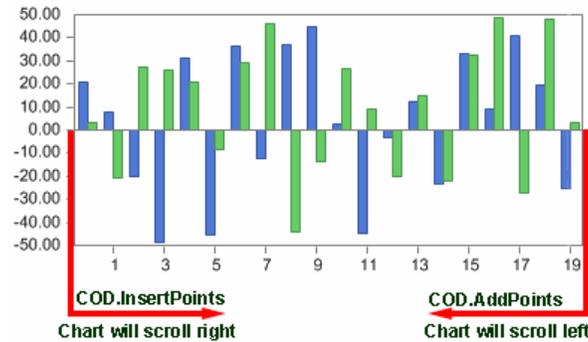
RealTimeScroll - Chart FX will scroll the chart to the end of the data set.



AddPoints vs. InsertPoints

Sometimes you want newly acquired points to be appended to the right of the chart forcing the chart to scroll to the left. In this situation you will use AddPoints COD enumeration. If you want the opposite behavior (points inserted in the beginning of the chart forcing the chart to scroll to the right), you will use the InsertPoints COD enumeration with the OpenData method call.

This behavior is depicted in the following figure:



Setting the Real-Time Style

The real-time style refers to the visual features you can control during display of the chart in your application. Basically, the two different settings available to this feature are: showing the loop marker and hiding the hourglass cursor from the real-time chart.

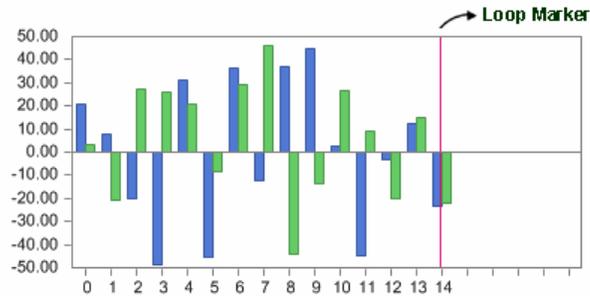


The RealTimeSettings object Style property allows you to control the cursor icon and loop marker with the following enumerations; you may also combine the flags:

LoopPos: Show Loop Marker

NoWaitArrow: Hide Hourglass cursor

The following figure includes a loop marker:



Scrolling the X Axis Labels in Real-Time Mode

If you're working with a real-time chart and also assigning legends to the points in the X Axis, it is imperative you scroll these legends in order to have your real-time chart display the appropriate legends every time new data is received.

In order to scroll the X Axis legends you must follow these rules:

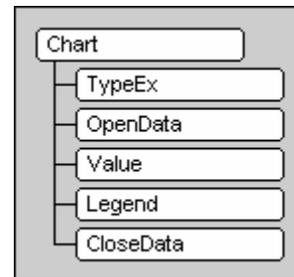
Turn on the 'NoLegnInvalidate' enumeration of the **TypeEx** property. When this flag is on, the chart will not be invalidated when the legends are set.

Open the communication channel (**OpenData** method) in combination with either the 'AddPoints' or 'InsertPoints' enumeration. One of these must always be used when creating real-time charts.

Set the value for the new point using the relative position in the Value property.

Within the same communication channel opened to pass the data using the Value property, you can use the Legend property to assign the X Axis label. You should set the new label using the relative position with the Legend property. (nOffset is the incrementing counter used to assign the label).

Finally, in the **CloseData** Method include the 'ScrollLegends' enumeration to force Chart FX to scroll the legends.



Note: Scrolling legends do not apply when using the loop marker in your real-time charts. Additionally, if you are not working with a buffer assigned to your real-time chart (BufferSize property) you must use the actual number of points in order to set the last point legend. You can read the nValues property and set the legend using the nValues-1 index instead of the BufferSize-1 index.

Customizing the Chart FX Toolbar

The Chart FX Tools provide an easy way for users to access the supported commands. However, if you plan to restrict access to any of those commands, keep in mind there are many ways to access any one command. For example, an end user may select the gallery command via the toolbar, menubar or right-click context sensitive menus.

This introduces an additional level of complexity as customizing only the toolbar does not mean other tools are not exposing the same command. For example, if you do not want end users to select an area chart, you may end up changing the toolbar, menubar, the right-click menus, or any other tool the Area button may appear in.

To increase control over these commands, Chart FX introduces the **Commands** object. The Commands object contains all the Chart FX commands available and their definitions (picture, text, etc). This allows the developer to remove or disable a command once. This will affect all the tools used to create and display the user interface for Chart FX.

Other important properties supported by the Chart FX API are the chart object **ToolBarObj** and **MenuBarObj** properties which allow a programmer to control toolbar and menubar positioning, visual attributes and command customization.

Relationship between the Commands & ToolBar Objects

When customizing the Toolbar, it is imperative you understand the relationship between the Commands and ToolBar objects.

When it comes to changing general settings, such as position and visibility, the supported members of the ToolBar object should be used. The chart ToolBarObj or MenuBarObj properties expose the members used to change the general visual attributes of the selected toolbar.

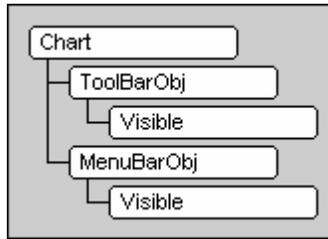
On the other hand, if you want to customize items (buttons) in the Toolbar, many of the features you may want to change (ToolTips, Picture, etc) are configured using the members of the Commands object. Items accessed using the Commands object are usually available to any of the Chart FX Tools.

In the following figure, many of the important visible attributes for a particular button in the toolbar like text, picture (icon), and style are controlled by the Commands object and not by the ToolBar object.



Showing/Hiding the ToolBar

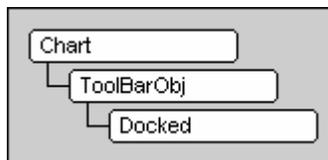
The ToolBar object exposes the **Visible** property allowing you to show or hide a selected toolbar in the chart. This Boolean property may be accessed using the API as follows:



Note: You can also use the properties dialog to configure this setting at design time.

Positioning the ToolBar

The ToolBar object exposes several members allowing you to position the toolbar in the chart's bounding rectangle. By default, the default position of the toolbar is docked to the top margin. However, by using the Docked property you can position the selected toolbar anywhere in the chart. For example, if you want to dock the toolbar to the left margin programmatically, you can use the Docked property as follows:



Removing Items from a ToolBar

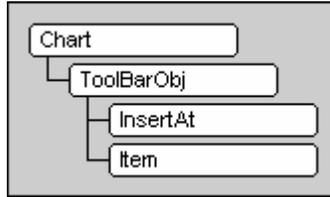
You may want to remove buttons in the Toolbar as a result of a specific condition in your application. For example, you may not want the end user to switch between 2D and 3D modes therefore you can remove that button in the toolbar. Or, you may want to allow users to save or open chart files. In this case, you can remove those buttons from the Chart FX Toolbar.

The ToolBar object supports the **RemoveAt** method to remove one or more items in the Toolbar. When using this method, be aware removed buttons will no longer exist in the toolbar and all the indexes will be recalculated by Chart FX. For example, if you remove the first button in the toolbar using the RemoveAt method, the index of the second button will become index 0, the index of the third button will become index 1, etc.

The RemoveAt method accepts 2 parameters; the first parameter receives the position from where to start removing items and the second parameter specifies the number of items to remove. Also, be aware vertical separators count as additional toolbar items, so if you plan to remove several buttons and they are separated, you must include the separators when calling the RemoveAt method. The RemoveAt method may be accessed as follows:



Once you have removed the gallery selector from the toolbar, you will want to insert the desired gallery commands into the toolbar. In order not to overwrite other commands in the toolbar, you will need to insert some "place-holders" for the additional gallery commands. The toolbar object provides the **InsertAt** method allowing you to specify the index of where you would like to insert items into the toolbar as well as the number of items you would like to add. Then, using the **ToolBarObj** property, you can specify the indices of the empty toolbar items to selected Command bar Items (CommandID enumeration). Visit the API Reference and Programmer's Guide in your Resource Center for details.



When you use the **InsertAt** method, the indexes for other buttons in the toolbar will be recalculated appropriately. In other words, because we have added 3 buttons at the beginning of the toolbar, the index for the button previously located in the first position is no longer 0 but 3, and the buttons we have added will have the 0, 1 and 2 index respectively.

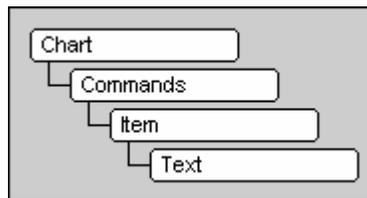
Note: Modifying just the toolbar is no guarantee the end user can only switch to those 3 chart types. The above example shows how to use the **InsertAt** method to add buttons to the toolbar. If you really want to prevent the end user from switching to other chart types, you have to modify the Commands list appropriately so those chart types may not be accessed from another tool in the Chart FX UI.

Modifying ToolTips

A tooltip is text appearing when the user positions the mouse over any button in the Toolbar. This is very helpful as it informs the end user about the functionality of the button, without actually pressing it and performing the action.

You may think this tooltip is part of the Toolbar; however, this text is also used in other parts of the Chart FX UI and therefore is controlled by the Commands object. Therefore, when you change the tooltip associated with a particular command, this change will be consistent throughout the Chart FX UI. Another point to keep in mind when changing the tooltip the text can be displayed somewhere else such as the menubar or context menus. Therefore, you must be careful not to make those tooltip strings too long or it will not be readable elsewhere.

To change the tooltip associated with a particular command, you must first identify the command ID for the command you wish to modify. For example, if you want to change the tooltip for the Gallery selector, you would reference the gallery enumeration (CommandID.Gallery). Once the desired command is referenced, you can set the text property for the command to modify the tooltip:



Changing ToolBar Icons

The entire collection of Chart FX tools shares the same icons used in the toolbar; therefore, the Commands object controls them. When you change an icon for a particular command, the change will be reflected in all of the other tools allowing access to that command. For example, the Data Editor icon is used in the toolbar as well as the right-click context menus. When you change the icon associated with the Data Editor command, both tools will reflect the change.

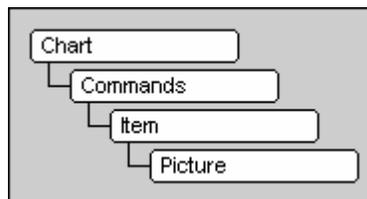
Any command may be assigned a pre-defined Chart FX icon from the icon list or a custom icon you add to the pre-defined list. This section will explain how to do both.

Selecting a Pre-Defined Icon

Chart FX provides a pre-defined list of 16x15 icons used for all the pre-defined commands. You can assign these pre-defined icons to other commands throughout the Chart FX user interface. To do this, the Commands object supports the **Picture** property. Setting this property to the index of a desired icon in the icon list, will assign that icon image to the referenced CommandID. Below is an example of the pre-defined icon list:



To change the Gallery command icon to the 'File' icon (index 1 in commands picture), you would use the following code:



Adding and Replacing Command Icons

Selecting an icon from the pre-defined list is not particularly useful because most of them are used by other Chart FX commands. This can cause confusion for the end user because the icons will not be unique to the function of a particular command. The **AddPicture** and **ChangePicture** methods may be used to add and replace icons in the pre-defined icon list.

The AddPicture method is particularly useful when you are adding custom commands to the Chart FX UI. This method will append one or more icons to the pre-defined icons list you can use to assign to existing or new commands.

The ChangePicture method is most useful when you want to change icons already being used by Chart FX. This method allows you to replace all or part of the existing icons list.

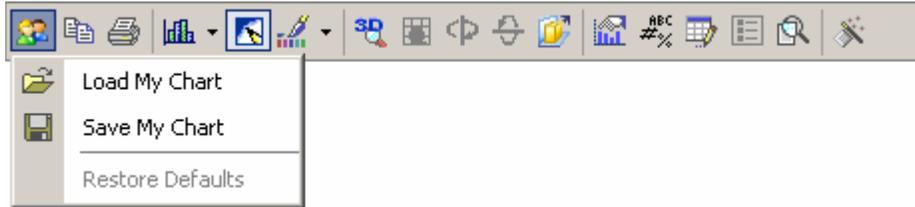
When adding or replacing any image in the icon list, it is very important your custom icons are the proper dimensions (16X15 pixels). When adding multiple icons you must ensure there are no spaces between the custom icon images.

AddPicture

Suppose you have two new icons you want to add to the icons list. Both of these icons are placed into a single bitmap file. Once you have added the new icons, you want to assign the first icon to the 'Gallery' command and the second for the 'PrintDialog' command. The AddPicture method returns an integer specifying the index position where the icons were added to the list. This reference can be used when assigning the new icons to the desired command using the Picture property. Visit the API Reference and Programmer's Guide in your Resource Center for details.

Working with Subcommands

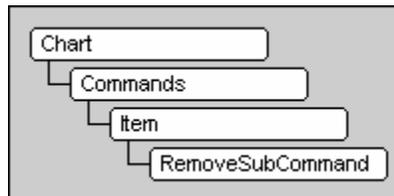
Although most buttons in the Chart FX user interface (ToolBar, MenuBar, etc) perform a specific action when pressed, there are other buttons presenting a list of subcommands, where one may be selected. For example, the Personalization button in the Chart FX Toolbar presents another list allowing the user to select personalization preferences, as depicted in the following figure:



These are called **Subcommands** lists and you can create or modify existing subcommands lists in the Chart FX user Interface. For this purpose, the Commands object exposes the **SubCommandID** property as well as the **RemoveSubCommand** and **RemoveAllSubCommands** methods.

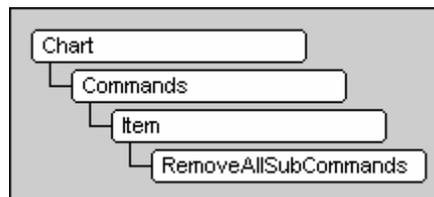
Removing existing Subcommands

In the figure shown above, you can see how the Personalization Options command presents a list of subcommands allowing the end user to select which preference to apply to the chart. Let's suppose you want to remove end user access to the 'Restore Defaults' subcommand selection. You can easily remove that option from the Personalization Options subcommand list by invoking the **RemoveSubCommand** method. By specifying the Personalized Options **CommandID** in the Commands property, you can remove items from the options list.



When you remove a particular subcommand from the subcommand list, the indexes will be shifted to the actual number of subcommands available in the command. For example, if you wanted to remove the 'Remove Defaults' subcommand and the separator above it, you would need to remove the same index twice.

You can also remove all subcommands from a command list by using the **RemoveAllSubCommands** method:



By removing a subcommand from the list, you will also remove this option from other tools using the same command ID.

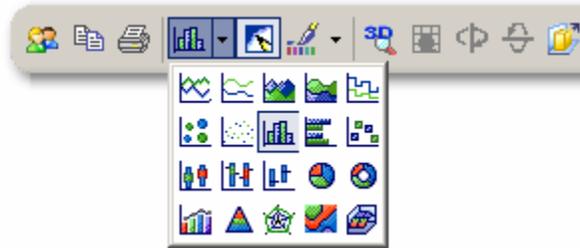
Adding Subcommands

To add a subcommand to a list you will use the **SubCommandID** property. If you are adding subcommands to a specific command you must make sure to keep the indexes in order as non-consecutive indexes may cause unpredictable results.

This feature is particularly useful when you are working with custom commands. For more information, refer to the "Adding Custom Commands" chapter below, where we show a custom command containing several subcommands.

Working with Selectors

A selector is a command containing a list of subcommands; when pressed, a list of images (icons) will be shown so the user can make a selection and when closed the selector will show the current selection. In Chart FX, the gallery icon in the toolbar is a selector as shown in the following figure:



The only notable difference between selectors and regular subcommand lists is the style for the subcommand has been set to 'Selector'. For more information on controlling the style for a particular command, refer to the **Style** property in the commands object.

Adding Custom Commands

In previous topics you learned the API provided by the ToolBar and Commands objects to add, remove or modify the Chart FX pre-defined list of commands. This was useful as you were able to customize the default Chart FX Toolbar, by removing existing commands, changing the icons and other important aspects related to the Chart FX User Interface.

Chart FX not only allows customization of the default toolbar with pre-defined Chart FX commands, but it allows you to add custom commands you can process internally in your application.

Let's say, for example, you want to add a button in the Chart FX toolbar that when pressed loads another chart with new data and new visual attributes. It is clear Chart FX will not provide this functionality as a pre-defined command because it will not know how to react when end users actually press such a button. Therefore, you'll need to add a custom command to give you the desired results.

Furthermore, you can also create groups of buttons, lists and selectors that will seamlessly integrate into your application. This section explains the necessary steps to add custom commands to the Chart FX User Interface.

Adding a custom command to the toolbar means setting the correct attributes using the Commands object and then using the properties and methods provided in the ToolBar object to physically add (or replace) that button in the toolbar.

The following are the steps you need to follow to add a custom command to the Chart FX Toolbar:

Add the Command ID to the Chart FX Commands list

The very first step to add one or more custom commands to the Chart FX UI is to add their IDs to the Chart FX Commands list using the **AddCommand** method supported by the Commands object.

The ID is an integer used to identify the command when the user presses or interacts with it. This is a unique ID associated to the custom command and should never be duplicated for use with another command. Chart FX defines its pre-defined IDs between 'CommandID.First' (29440) and 'CommandID.Last' (29951), so make sure you avoid a number in this range as it will cause unpredictable results.

Setting the Visual Attributes of the Custom Command

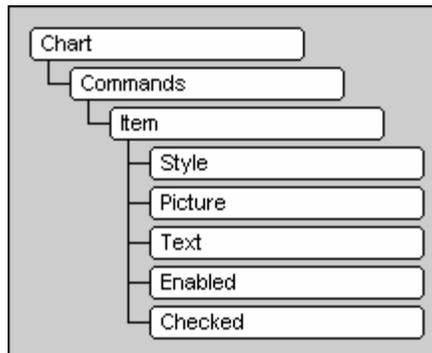
Once the Command ID has been added to the list, you must instruct Chart FX how to display this command in the UI. Will it be a button or a list? What will be the icon used? What is the text to be displayed? Is it enabled by default?

There is a property for each particular attribute in the commands object as shown in the following API:

Style. The Style property allows you to tell Chart FX if the command you're adding is a two-state button, a list, a selector, or if the command belongs to a group. If you do not set the Style property, the custom command is added as a regular button.

Picture. The Picture property allows you to specify the icon to be used in the button. You can add your own buttons as described in the "Changing/Adding icons" section.

Text. The Text property allows you to set the tooltip to be displayed, or the text if the command is to be shown in the menubar. For example, to set a new tooltip to the button you can set the Text property.



Adding the Custom Command to the Toolbar

To add the custom CommandID to the toolbar, you can use the **InsertAt** method and the CommandID properties exposed by the toolbar object. Visit the Resource Center and API reference for detailed samples.

Processing the Custom Command

Since you have added a Custom command to the Chart FX commands list and the button has been added in the Toolbar, you need a mechanism to add your custom code to processes that particular command. The **UserCommand** Event is posted to your application when the user clicks or interacts with the custom command you have added. So if you want to change the chart gallery to lines when the user presses the button, you will process the UserCommand event.

Customizing Context Menus

Each Chart FX context menu has its own CommandID and is comprised of a list of subcommands. Custom, as well as pre-defined, subcommands may be added or removed from selected context menus by utilizing the supported members of the CommandBar and Commands objects.

There are context menus for the axes, background, constant line, series, stripe, title and Chart FX tools. By selecting a particular context menu, you can add or remove commands from the context menus to control the options users have access to when selecting a menu on demand for a particular chart element.

The **UserCommand** Event is raised in your application when the user clicks or interacts with the custom command you have added. Using the UserCommand event, you may process custom commands you add to your applications. Visit the API Reference and Programmer's Guide in your Resource Center for details on customizing context menus.

Capturing Mouse Events

Chart FX provides different properties, methods and events allowing you to control how the chart reacts to mouse interactions from the user. For example, you can customize the way tooltips are displayed, if the user is capable of dragging markers, thus changing the value for a particular point; or you can add your own code when the user right-clicks on a chart title. You can even simulate mouse clicks at a particular location in the chart to see if the user is positioned over a particular element in it.

Let's start by introducing the properties provided by Chart FX affecting how the chart reacts to mouse interaction.

Marker Dragging

By default, charts are created with the **AllowDrag** attribute defined as false. When set to true, end users may change the value for a particular marker by positioning the mouse over a particular data point and dragging the marker to a desired value. Although this is a cool feature it is not useful if you do not want users to modify values in the chart.

Menus on Demand

When charts are rendered as objects (not images), a series of menus on demand are supported when a user right-clicks in the chart area. Depending on the type of element selected when the right-click event is raised, a corresponding context menu is prompted to the user. For example, if a user right-clicks on the Y Axis, a menu will appear and the user will be able to access all axis properties. By default, these context menus are enabled, however this may be controlled using the **ContextMenu** property.

Note: If you want to customize certain options "inside" each menu appearing when the user right-clicks a particular element in the chart, refer to the "Customizing the Toolbar" chapter.

Customizing DataTips

By default, when the user positions the mouse over an element in the chart (Not the Toolbar) a tooltip will appear with specifics about the element. Particularly important are data points displaying information, such as the series legend, the point legend and value markers. This behavior can be easily modified using the **TipMask** property. Using this property, you may configure a string containing special flags or macros that when rendered in the chart will represent information specific to that chart element inside the tooltip. The same flags may also be used in similar properties like the **PointLabelMask** and **Link.Url**.

For example, if in an XY Plot, you want to modify the text displayed in the tooltip with just the X & Y Coordinates of the point, you can set the TipMask property to the following string which contains only the special macros:

```
"%x%v"
```

This property can also be combined with custom text to provide a clearer message to the user. For example if you want the tooltip to display: "I'm Series 1 and my value is 13.5", you can set the TipMask property using the following string:

```
"I'm series: " + "%S " + " and my value is: " + "%v"
```

Although the TipMask property is the simplest way to customize data tips, the chart's **GetTip** event is provided so you can add context sensitive information to particular data points in the chart. For example, you want a particular data tip to show specific text different than other points. The GetTip event returns a set of arguments used to detect which data tip is about to be displayed and then modify the text for that particular point. Visit your Resource Center and API Reference for more information regarding this and other supported chart events.

Note: Similarly, the **GetPointLabel** and **GetAxisLabel** events may be used to modify the text for their respective objects.

Other Mouse Related Events

Chart FX provides several mouse events allowing you to customize how the chart reacts to mouse interaction coming from the user such as: `DoubleClick`, `MouseDown`, `MouseMove` and `MouseUp`. You may process these events to add specific functionality in your application.

All mouse events return a **MouseEventArgsX** object which contains all necessary data to handle the event: `Buttons`, `Clicks`, `HitType`, `Points`, `Series`, `X`, `Y`, etc.

Tracking the Mouse

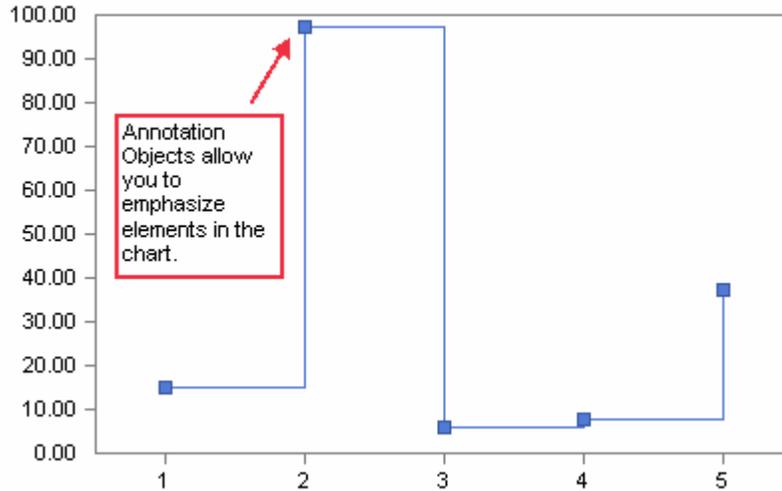
Sometimes you may need to track mouse movements in the chart area, detect where the mouse pointer is positioned and which element the mouse is pointing to. This may be accomplished using the **MouseMove** event in conjunction with the **HitTest** method to determine where the mouse pointer is located and to which chart element it is pointing. For more information, reference your Resource Center and API Reference.

Using Chart Events in Web Applications

Capturing mouse events in web applications requires Active Client, for detailed information on this please refer to Appendix C.

The Chart FX Annotation Extension

The Chart FX Annotation Extension allows you to include floating objects in the chart. It also provides a User Interface so your users can create objects and highlight different areas in the chart. Using the Annotation Extension, you can add text, arrows, different shapes and even images you can freely move around the chart area. This allows you to direct the attention of an end user directly to a particular portion of the chart. Below is an example of an annotation arrow and text object used in a chart:

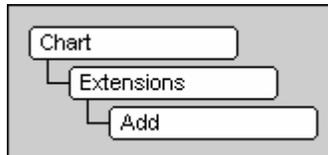


Integrating the Annotation Assembly into your Project

If you plan to utilize the Annotation features of Chart FX, it will be necessary to reference in your project to the Annotation component. Check your Resource Center for details on adding the correct component for your Chart FX product.

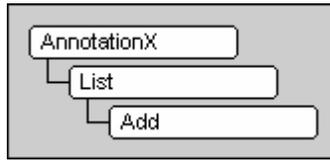
Creating/Adding the Annotation Extension Object

Once you have added the reference to the Annotation library in your project, you are ready to create the Annotation Extension object. The Annotation Extension object must be created as a type **AnnotationX**. Once created, the Extension object may be added to the chart using the Chart object **Extensions** property. Below is an example of the API call used to add the Annotation Extension to the chart.



Configuring the Annotation List Object

Every Annotation Extension object created will include an Annotation Object List. The Annotation Object List is a collection of all the individual annotation objects you wish to appear in the chart. To add objects, you would simply create the desired annotation object, set the visual attributes and finally add that object to the Annotation List. Once the AnnotationX Extension object has been added to the chart, the added objects will appear in the chart.



By using the **List** property of the AnnotationX object, you may access methods and properties allowing you to add, clear, count, remove and access particular elements in the Annotation Object List. To clear all the objects currently included in the object list you can use the following API:



Creating Instances of Annotation Objects

There are a variety of different Annotation objects exposed when you add the Annotation Extension to your project. The following objects may be included in your charting applications by implementing the Annotation Extension:

- Arc (AnnotationArc object)
- Arrow (AnnotationArrow object)
- Balloon (AnnotationBalloon object)
- Circle (AnnotationCircle object)
- Group (AnnotationGroup object)
- Picture (AnnotationPicture object)
- Polygon (AnnotationPolygon object)
- Rectangle (AnnotationRectangle object)
- Text (AnnotationText object)

Once the selected object has been created, you may begin to assign supported attributes to the object. Many objects do not expose any special properties or methods specific to the annotation object type; rather they share a set of common properties (**AnnotationObject** object) which may be applied to any of the annotation objects.

Many objects expose specific properties to be set to display the object appropriately in the chart area. For example, all objects expose the **Top** and **Left** properties allowing you to position the element; additionally, the annotation arrow object also exposes a specific property **TailStyle**, which allows you to configure the arrow tail style.

It is also important to call the **Refresh** method for an object when added or when any attribute of the object is modified. Due to performance reasons Chart FX does not do this automatically.

Using the **Enabled** property of the AnnotationX object, developers may allow or prevent users from interacting with annotation objects added to charts programmatically; by default this property is set to False. Setting this property to True, allows end users to interact with annotation objects in the chart area (size, location, color, etc.). When displaying the annotation toolbar, the Enabled property must be set to True before the annotation object is added as an extension to the chart. Refer to the annotation toolbar section for more information.

Note: The AnnotationObject object supports the **AllowMove** and **AllowModify** properties to also allow and deny modification of individual annotation objects.

Height, Width and Object Orientation

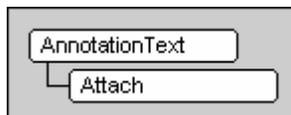
When working with annotation objects, it is very important to set the appropriate **Height** and **Width** for any object you create. By default the height and width of each annotation object is zero (0), so by neglecting to set these properties the object will not appear in the chart.

These properties also have different effects on individual object types. For example, manipulating the height and width values for a circle object allows you to create ellipses and oval shapes. When creating an arrow annotation object, setting the height and width to negative values will configure the arrowhead to point in different directions. For example, assigning a positive width will cause the arrow to point to the right-bottom portion of the chart. Similarly, assigning a negative value for the width will cause the arrow to point to the top-right corner of the chart. The orientation of many of the annotation objects can be altered through the use of the height and width properties including the rectangle, circle, arrow, arc and balloon objects.

Positioning an Object in the Chart

The most common means of positioning an annotation object in a chart area is utilizing the **Top** and **Left** properties. These properties accept pixel values for configuration, where the origin of the chart (0,0) is located at the top left corner of the chart area.

The **Attach** method of the **AnnotationObject** object has also been implemented to attach annotation objects to a particular point in a chart. The beauty of using the Attach method is it accepts chart axis values instead of pixel values the Top and Left properties require. The Attach method is an overloaded method supporting different configurations. Refer to the API Reference for details:



When attaching an object using the two-parameter configuration, you are attaching the center of the annotation object to the selected point. Because you are attaching at the center, it is still required to set the height and width of the annotation object. The parameter **dx** represents the X Axis point and the **dy** represents the Y Axis point where you wish to attach the object.

Note: When attaching an object to a chart using the attach center method (Attach(dx, dy)), it is important to understand the center of the annotation object will be attached to the configured point. Therefore, if you attach an arrow object to a chart, the head of the arrow will not be attached to that point; rather the center point in the line of the arrow will be attached to the configured point.

When using the four-parameter configuration, the parameter **dxLeft** represents the left X Axis value, **dyTop** represents the top Y Axis value, **dxRight** represents the right X Axis value and **dyBottom** represents the bottom Y Axis value. When attaching an object using the four-parameter configuration, each of the corners of the annotation object is attached to a particular point in the chart. Because this automatically sets the size of the object, the height and width properties are not needed unless you wish to control the object's orientation. Another benefit to using the four-parameter approach is each of the four corners of the object is anchored to a point in the chart. This allows the object to resize when the chart itself is resized.

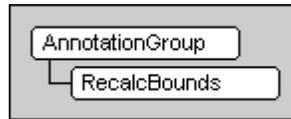
Annotation Group

The AnnotationGroup object allows you to create annotation group objects. An annotation group will consist of multiple instances of annotation objects. One advantage of working with this is the group object exposes all common properties (AnnotationObject object), so you could change visual attributes of all of the objects in a group with just one property call. Another advantage is when objects are grouped together, they may be rotated and moved as a group. For example, let's say you create a stick person figure using circles and line annotation objects. If each object is rotated individually, the stick figure loses its shape. Conversely, if you create a group of objects that make a stick figure and then rotate the group, all the objects will keep their shape and the stick figure will stay intact.

Creating an annotation group object is exactly like creating any other annotation object. Once the group object has been created, you may begin to add individual objects to the group. Once all the desired objects have been added to the group, the group can be added to the annotation list object. By setting attributes to the group object, all the individual objects added to the group are altered.

Recalculating the Group Object Bounds

Whenever a group is added to the annotation list, the bounds of the group object must be set. This may be done by setting the height and width of the group as well as attaching or configuring a position in the chart for the group. Instead of setting these properties manually, you can instruct Chart FX to calculate these values for you by calling the **RecalcBounds** method. This method should also be invoked every time an object is added to a group or when any of the objects contained in a group are resized or moved.

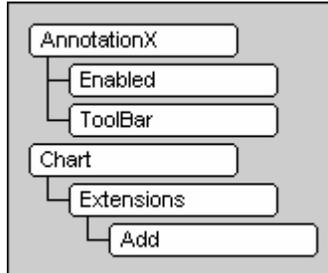


Annotation ToolBar

The Chart FX Annotation Extension supports an Annotation ToolBar. The Annotation ToolBar allows end users to add annotation objects to a chart as well as rotate, group and even flip them around. Below is an example of the annotation toolbar:



In order to enable the annotation toolbar, you must create an annotation extension object, set the **Enabled** property to "True", add the newly created object to the chart and set the **ToolBar** to True. Below is an example of adding the annotation object to a chart and enabling the annotation toolbar programmatically:



Note: You must set the Enabled property to "True" before adding the extension for the toolbar to be visible.

Using the Annotation ToolBar, users may add rectangles, circles, arrows, arcs, pictures, text, balloons and polygons to charts. There are also some properties affecting the way these objects are drawn.

DefaultBorderColor: Used to set the border color for annotation objects created using the annotation toolbar.

DefaultFillColor: Used to set the fill color for annotation objects created using the annotation toolbar.



Bitstream Chart Generation

Since we released Chart FX Internet many large corporations have incorporated Chart FX as an essential reporting tool for their Internet, intranet and extranet sites. Many of these companies impose strict security on their users through the use of firewalls and proxy servers; some of them even go to the extent of configuring the browser so it does not accept anything harmful to their network. Needless to say many of these companies do not allow users to download and use Client Control objects, even if they come from trusted third party vendors like Software FX.

Even though Chart FX always generates images by default, for many corporations the fact that these images are being generated and stored in a particular directory poses a security concern to many of them. For example, a hacker could determine a chart name, create a page and load the chart in his machine before it is actually erased. Many sites display very sensitive data where that risk is simply unacceptable. With this security concern in mind, Chart FX allows web developers to easily setup pages returning charts as bitstreams instead of generating chart files and letting browsers download files previously generated in the Temp directory.

The **GetHtmlData** method is a method provided by Chart FX allowing the browser to receive a bitstream instead of a chart file from the Temp directory of your web server. The GetHtmlData method receives the same parameters as the GetHtmlData method, except the result will be a data push directly to the browser. No references to download a file will be generated when using this GetHtmlData method.

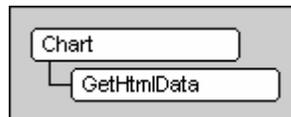
In fact, when the user attempts to view the source for the resulting page, no information about the chart file will be displayed. This is because all the necessary information was sent to the browser via a bitstream instead of a chart file or image.

Creating a Chart Image without Downloading a File

This technique is used when a chart image (PNG or JPEG) will be generated by the server side component. The idea is as simple as including the tag wherever you want the chart to be displayed and the SRC parameter points to the file generating the bitstream by invoking the GetHtmlData method. The IMG tag should read as follows:

```
<IMG SRC="chart.aspx">
```

The chart file will contain the necessary code to generate the desired chart and invokes the GetHtmlData method as a result of the page. A very important issue when generating the server GetHtmlData page is to make sure there are no other HTML tags or Line feeds in the resulting file as the chart may not be correctly returned and displayed in the browser. In other words, make sure the chart file only returns the results of the GetHtmlData method with no additional characters.



There must not be any spaces or carriage returns before the start of the scripting anchors (i.e. <%). If you press [Enter] then a carriage return will be generated in the result of the script and the IMG source will not be able to read the image generated by Chart FX. Therefore it is strongly advised you use a plain text editor when generating these files as many advanced editors may include undesired HTML tags in the resulting file.

Note: If you are using "Image Maps" in the charts, you should be aware this feature is not supported when using the bitstream generation. The Method **GetHtmlDataEx** should be used instead.

Creating a SVG Chart without Downloading a File

This technique is used when a chart image (SVG) will be generated by the server side component. The idea is as simple as including the <EMBED> tag wherever you want the chart to be displayed and the SRC parameter points to the script generating the bitstream by invoking the GetHtmlData method. The IMG tag should read as follows:

```
<EMBED SRC="chart.aspx" NAME="Chart1" TYPE="image/svg+xml"
PLUGINSPAGE="http://www.adobe.com/svg/viewer/install/" WIDTH="400" HEIGHT="300">
```

The chart file will contain the necessary code to generate the desired chart and invokes the GetHtmlData method as a result of the page. A very important issue when generating this JSP is to make sure there are no other HTML tags or Line feeds in the resulting file as the chart may not be correctly returned and displayed in the browser. The chart file should only return the results of the GetHtmlData method with no additional characters.

Creating a Chart Object without Downloading a File

This technique is used when a chart image (PNG or JPEG) will be generated by the server side component. The technique involves including an <OBJECT> tag wherever you want the chart to be displayed and setting the DataPath parameter to point to the page generating the bitstream by invoking the GetHtmlData method. The IMG tag should read as follows:

```
<object id="Chart1" classid="/SomeApp/chartfx6/download/
ChartFX.MainClient.DLL#SoftwareFX.ChartFX.Internet.MainClient"
WIDTH="450" HEIGHT="280" >
<PARAM NAME="DataPath" VALUE="http://127.0.0.1:8080/ChartFX/chart.jsp">
<PARAM NAME="AssemblyTarget" VALUE="ChartFX.Internet.Client,
Version=6.0.1507.41460, Culture=neutral, PublicKeyToken=a1878e2052c08dce" >
<PARAM NAME="ClassTarget" VALUE="SoftwareFX.ChartFX.Internet.Client.Chart" >
</object>
```

The chart file will contain the necessary code to generate the desired chart and invokes the GetHtmlData method as a result of the page. A very important issue when generating this file is to make sure there are no other HTML tags or Line feeds in the resulting file as the chart may not be correctly returned and displayed in the browser. The chart.aspx file should only return the results of the GetHtmlData method with no additional characters. Additionally, be sure to update the version number to match your installation.

Performance & Scalability



Overview

This section is geared for developers using Chart FX in resource intensive situations. You will learn how to fine-tune Chart FX to improve your application's ability to accommodate a growing number of users and give each user a satisfactory level of responsiveness.

Tuning Settings and Features for Performance and Scalability

Tuning Chart FX

While it is true that many development tools and web development technologies significantly reduce development time and allow for quicker deployment, this paradigm presents a whole new challenge for developers; among them, server scalability and performance.

The real issue in web application design is user experience. To make an application perform as a traditional desktop application when it is actually a distributed application that could be simultaneously servicing hundreds or even thousands of users.

To understand how Chart FX impacts server performance, consider every time a web page containing a chart is accessed, the Chart FX Server component creates a chart in memory using a device context created on the server. Therefore, hardware and software settings on your server affect how quickly these charts are created.

In general, Chart FX performance depends to a great extent on the chart format generation (Image, .NET Control, SVG, Flash, etc) and the chart return mechanisms (**GetHtmlTag** or **GetHtmlData**) chosen. Also, when generating chart images you should pay close attention to the Chart FX default settings and chart size as major performance factors when the server is under a heavy load.

Settings, Features and Their Impact on Performance

Chart FX was designed to conform to web development guidelines and to take advantage of the server-bound nature of web applications. Nevertheless, many of the product's default settings are tailored for improving the chart's look rather than focusing on server performance. In situations where Chart FX is heavily used, these default settings may need to be configured to improve performance.

For example, Chart FX uses a default smoothing algorithm (anti-aliasing) on text and markers that negatively impacts server performance. This feature can be turned off by using the **SmoothFlags** property to "None".



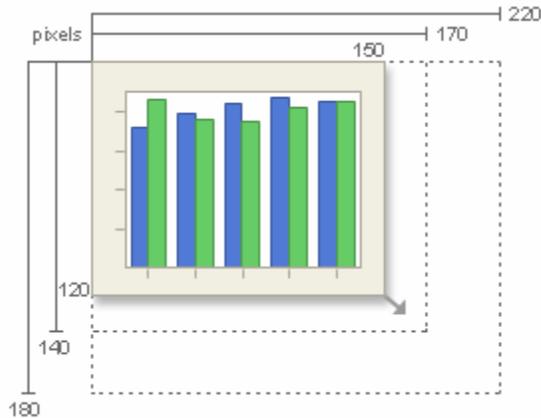
Turning off the smoothing feature also increases the amount of users you can serve concurrently.

In general, if server performance is of real concern you should refrain from making extensive use of the features geared towards achieving a cosmetic effect on the chart. Particularly, features like **Borders**, **Transparency** and **Gradients** which heavily use the system graphical API as a way to improve the way charts look.

Chart Render Size and Format

Render Size

A common practice among web developers is to create a big chart that can be easily read on a browser. However, this practice can be an important factor on how your server behaves and performs under heavy load. Essentially, a bigger chart means a larger image must be processed, generated, stored and finally downloaded; affecting, in one way or another, the overall application's performance. Therefore, you must be careful when choosing the final chart rendering size in the page if server performance is a real concern.



To illustrate this, suppose you increase the size of a 400x300 PNG image of a chart and measure its impact on server performance.

We found that for every 30% increase on the overall chart size, the server experienced a significant decrease in the number of Requests per Second it could process.

Changing the chart size may be costly on your overall page design. However, it may be an excellent source to boost performance if you have exhausted all other options.

While reducing the chart size considerably will benefit server performance, you must be careful not to select a size too small that will compromise the chart's readability.

Format

As a server component, Chart FX allows developers to generate charts in a myriad of formats. Which one to choose depends not only on performance and scalability, but also on other important issues such as browser compatibility, interactivity, accessibility and security.

The Chart FX server control can dynamically generate and render PNG, JPEG, .NET, ActiveX, SVG and Flash files.

In general, chart formats that require a viewer (i.e. .NET, ActiveX, Flash and SVG) enhance server performance since chart files are very small and each client provides much of the processing load in painting the chart. These viewers will also allow a better analytical experience by allowing browser interaction without additional intervention from the developer or trips back to the server.

Specifying a specific viewer through the HtmlTag property may improve performance, however this may also affect accessibility as the chart may not be viewable by certain browsers depending on the output type chosen. For example, specifying Flash as an output format is not visible on browsers that do not have Flash already installed.

In general, viewers are acceptable under a controlled environment like an intranet. However, if you must reach a wide audience, you may be forced to generate universally accepted images like PNG and JPEG.

These raster images decrease performance since each chart needs to be painted and even stored on the server. Also, these charts will provide limited or no analytical capabilities on the browser since they are rendered as static images, except for PNG images that support hot spots or URL links on most chart elements. However, the greatest advantage in using chart images is that they provide universal access to charts from any browser, platform or operating system.

Chart Return Mechanisms and Web Farms

Return Mechanisms

To improve the level of responsiveness as well as enhancing support for different server architectures, Chart FX provides two ways of processing and returning charts to the browser. These mechanisms are provided by 2 methods: **GetHtmlTag** and **GetHtmlData**.

When a user hits a page containing the **GetHtmlTag** method, a chart file is saved to disk and a HTML tag (IMG, OBJECT) is returned so the browser can know the location of the chart file on the web server. Finally, a second trip back to the server is necessary to pull the chart from the server and display it on the page as a static image or an active component. One of the greatest benefits for using **GetHtmlTag** is that charts can be easily integrated into your existing web pages and allows support for automatic browser detection. However, saving a file to disk can have its toll on server performance.

On the other hand, the **GetHtmlData** method prevents the chart file from being saved to disk by bit-streaming the chart directly to the browser; this process leads to fewer round-trips between the client and the server. However, integration to the web page is awkward because you must point to another file that returns the bit stream to the browser. Also, limited browser detection is provided using this methodology preventing your application from making "smart" decisions based on the user's browser capabilities.

In terms of performance, although the **GetHtmlData** method is processor intensive, it allows developers to boost performance and scalability and permits the use of Chart FX on intricate server architectures such as web farms.

For more details and samples about the methods **GetHtmlTag** and **GetHtmlData**, please consult the Chart FX Programmer's Guide.

Web Farms (Server Clustering)

Once you hit a certain threshold, cost-effective scalability requires the use of multiple processors spread across multiple servers. In other words, you'll need a Web Farm.

We mentioned that when using the **GetHtmlTag** method a chart file is saved to disk. The one problem on web farm architectures is that when the browser comes back to the server to retrieve such file; the HTTP request could be directed to another web server on the farm on which the chart was not originally saved, resulting in an empty chart placeholder on the page.

While changing the server affinity setting, which lets you instruct the service to route HTTP requests over the same physical server once the first request has gone through, may solve this issue, affinity slows the system down by requiring it to maintain a user-to-server mapping and perform a lookup upon each request. In other words, changing server affinity compromises scalability and the benefits of request-based load balancing are lost. For this reasons, Chart FX provides configuration mechanisms that will take care of this issue, allowing the servers to know exactly where the chart files are. For more information about this settings please consult the Chart FX Programmer's Guide.

However, generating a chart file that will be saved on disk poses many server performance and scalability problems on a Web Farm. Therefore, the use of the **GetHtmlData** method bit-streams the chart directly to the browser, preventing many of the above mentioned issues and providing a more natural integration to a web farm is more suitable in this environment.

Chart FX PSS

The Chart FX PSS (Performance, Scalability & Security) Add-On allows organizations to attend to crucial **Performance**, **Scalability** and **Security** issues when using the Chart FX in a web server environment. With Chart FX PSS you can expect a substantial increase in your Web server's throughput, commonly measured in requests per second and avoid security issues associated with generating and storing server side charts.

Chart FX PSS is basically a service that communicates with the Chart FX server component and handles caching, delayed generation and security according to configured conditions. Chart FX PSS can be configured through static configuration files, or dynamically through API.

Caching

Caching is a technique used to store items in memory the initial time they are requested with the purpose of improving performance and scalability. In a web application, this technique allows a server to avoid recreating information that satisfied a previous request, saving time and resources by utilizing server time more efficiently.

Chart FX PSS provides a powerful caching mechanism. To understand how caching increases server's throughput, consider that when the web page is running, the Chart FX server component will essentially create and paint a chart in memory using a device context created on the server, the server must then store the chart on disk for immediate access by the browser or bit-stream it to the client browser. This process is repeated for each subsequent request, whether the chart intrinsic data has changed or not. For example, without caching, a page that displays the "Chart of the Day" will force the server to create the chart as many times as the page is requested thus wasting server time and resources. In contrast, with caching this chart will be created, stored in memory and delivered only once according to an expiration time set by the programmer.

As soon as the Chart FX PSS is installed on your server, you are capable of activating caching on a per chart basis by loading the extension as showed in previous paragraphs and by setting the `ExpirationTime` property. Once you have enabled caching, Chart FX PSS will place the chart in direct access memory for the amount of time you specify, and only at that moment will generate a new chart.

In most cases, charts are created with a number of possible parameters that are used to connect to a data source and generate various responses or intrinsic chart data. For example, if a sales chart is generated as a result of an HTML form that allows users to filter data by store location, date, city and zip code input boxes, the chart will be different for each user and thus caching a single chart will not render the desired results.

By default, Chart FX PSS will cache a version of the chart based on the page parameters. In other words, each chart will have stored the string or form POST parameters that are passed with a request, by the HTTP headers passed with a request. This behavior is controlled by the `CacheParameters` property. For more detail in this topic please refer to the Chart FX Programmer's Guide.

Sliding Cache Expiration

In some applications, the absolute nature of the `ExpirationTime` property can have adverse effects on the server. For example, if an application handles hundreds or even thousands of charts, the caching mechanism may end up storing charts in memory that are not being frequently used or accessed and thus not needed in memory.

The `MemorySlidingExpiration` property sets a time that will be renewed with each request. When the `MemorySlidingExpiration` reaches its value and the chart has not been requested, it will then be removed from memory and stored on disk. In contrast, if the chart is accessed within the limits of the `MemorySlidingExpiration` property the chart will be kept in memory and the time will be reset with each subsequent request. This mechanism allows the server to make better use of its memory and other server resources.

For example, you can set the `ExpirationTime` property to 2 days and the `MemorySlidingExpiration` property at 1 hour. If a chart is accessed for a second time within an hour, then the sliding cache time will be renewed to 1 hour and the chart will be kept in memory. If the chart is not accessed within an hour, the chart will be removed from memory and stored on disk. When the chart is then requested again it is loaded into memory and the sliding cache will be reset to 1 hour. It is important to note the chart will be removed from the cache 2 days after it was first accessed (as instructed in the `ExpirationTime` property).

Chart Delayed Generation

When page output is buffered, the server does not send a response to the client until all of the server scripts on the current page have been processed. For long scripts, this may cause a perceptible delay.

Chart FX PSS delay generation mechanism can significantly improve the performance of large web applications by making better use of the roundtrip time thus maximizing server processing time.

This mechanism allows the GetHtmlTag method to generate a tag that can be used by the browser and immediately return control to the page without waiting until the chart gets generated on the server. When the page output is buffered, this will result in a significant improvement on performance since server scripts will be processed faster without waiting until the Chart FX server controls generates, paints and stores the chart, and returns control so the rest of the script completes execution.

In essence, this mechanism of returning control immediately so the rest of the server script can continue running without further delay from the chart control allows for a better use of server idle time when data is traveling from the server to the clients.

At a later time, when the chart is requested by the browser, Chart FX PSS will automatically dispose it. Nevertheless, it can be configured to dispose of charts that have been generated but not requested by the client browser.

Security

When Chart is used in a web server, it is necessary to create a folder where it will store the temporary images generated by a GetHtmlData call. This folder must be part of the web server since it needs to be accessed from the client browsers. This process unveils two critical security issues:

First, while files remain in this directory, an "unauthorized user" may find ways to view and inspect charts that were generated by the Chart FX server component. Secondly, since this directory is used by the Chart FX server component as a general directory, it can't apply discretionary authentication information to files being generated by the server component. For example, charts with private data such as employee salary information may end up requiring authentication, but since it is stored on the same directory it will inherit the same security restrictions as the rest of the files in this folder. In most cases, this Temp directory is created with anonymous access which means all files can be accessed by virtually anyone.

These issues are minimized by the fact that each chart name is generated with a random seed that makes it impossible to detect.

Nevertheless, Chart FX PSS takes a safer approach and addresses both issues by avoiding generated chart files from being stored on disk as it was previously explained; and also, each chart can be signed with each user authentication and will be returned only to the user who initially requested the chart thus making Chart FX security as strong as the security policies imposed by your web server.

Chart FX Extensions



Overview

There are multitudes of charts available for displaying the multitude of complex data available in today's applications – each one addressing a unique need. While Chart FX tries to address each of these needs by providing a multitude of galleries within the core product, there are some charts and needs that are specific to special markets such as financial charts for financial markets as well as real time charts for on demand applications.

In order to address the special markets and needs, Chart FX introduces the notion of Chart FX Extensions. With extensions, Chart FX is able to address the needs of specific markets in detail without interfering or bloating the core Chart FX product used by many. This section introduces the various extensions available for Chart FX.

Introduction / FAQ

Our developers have spent a great deal of time designing Chart FX around an open and adaptable architecture. Chart FX Extensions take advantage of this open and adaptable architecture and work with the core Chart FX products to extend their capabilities or to provide a specific functionality.

Why aren't the free Chart FX Extension included as part of the core products?

Many commonly used features are indeed included in the product itself. However, if the functionality is significant enough where it might cause stability problems in a released product, our developers prefer to create an extension and isolate this functionality in a different library until it can be included as part of a major release. Sometimes the new extension is also kept separate permanently for performance reasons.

Why are some extensions free and some aren't?

There are essentially two types of extensions: extensions that complement core products and extensions that create new products. Extensions that complement core products are usually free, since they provide a functionality that we feel will benefit most Chart FX users. On the other hand, extensions that create new products, although not free, provide a cost effective method of adding specific functionality to the core products without the need of purchasing a full-blown product to provide such features.

How do I know which Chart FX core products a Chart FX Extension is compatible with?

Each extension is compatible with one or more core Chart FX products, and we are working to provide support for these extensions with the entire suite of Chart FX products. Our extensions web site contains the latest compatibility information.

Can I try out an extension?

Yes! If an extension is not FREE, we offer a 30 day trial version. Download the free trial versions of the extensions from our extensions website.

Chart FX Extensions Website

The Chart FX Extensions website (<http://www.softwarefx.com/extensions/>) is the best way to get the latest information regarding the supported extension products and keep up to date with the latest extension developments from Software FX. From the website, you may download trial versions and the free Chart FX Extension at any time.

As of the printing of this manual, the following extensions are available:

Free Extensions

Chart FX PSS Extension
Chart FX MS Office Add-On
Accessibility Extension
Polar Gallery Extension
Flash Generation Extension

Licensed Extensions

Chart FX Maps
Chart FX OLAP
Chart FX Statistical
Chart FX Financial
Chart FX Real-Time
Chart FX Wireless

Chart FX Maps

For over a decade, Chart FX has helped developers integrate charts and graphs in their reporting and data visualization applications. Unfortunately, not all situations lend themselves to be represented easily with traditional charts.

For example, consider the case of depicting data on a US geographical map or an airplane seating map. First, it will not be intuitive for end users if data would be represented in a bar, line or pie chart. Also, although you could easily create hotspots or hyperlinks in different areas of the image; as a developer you don't have access to individual objects (such as states or seats, respectively) to set other attributes, such as fill colors and borders to visually convey real-life conditions in the map.

If your business problem would be better represented by a customized vector graphic that changes according to the underlying data then Chart FX Maps is your answer.

What is Chart FX Maps?

Chart FX Maps facilitates developers in creating solutions using SVG, a versatile vector-based imaging standard. This allows integration of dynamic maps and deployment of very impressive Windows and Web Forms applications.

As a developer, you could simply select and integrate hundreds of pre-built geographical maps; or, if you are building an application that requires a custom map, you could custom design one and tailor it to your specific requirements.

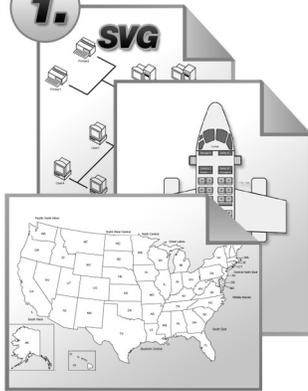
Chart FX Maps integration into Visual Studio® .NET is simple and straightforward; enabling the creation and deployment of powerful applications where end users can visualize and analyze data graphically and interactively.

Additionally, Chart FX Maps works as an extension of Chart FX, allowing map data to be viewed as other popular chart types for further analysis without additional development or coding efforts. This effectively increases developer productivity and customer satisfaction.

The process is as simple as selecting or creating a vector map, feeding the data, and finally, specifying visual attributes such as fill color and border styles. Just take a closer look...

Chart FX Maps At-a-Glance

1.



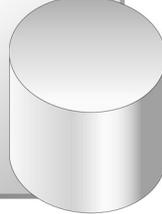
Select your map or create one from scratch.

Chart FX Maps uses the versatile SVG (Scalable Vector Graphics) image standard to allow developers integrate and create the most impressive graphical .NET interfaces for Windows and Web Forms applications. As a developer, you could select and integrate hundreds of pre-built geographical maps or create SVG files from scratch that are tailored to your specific application requirements.

2.

Database or XML

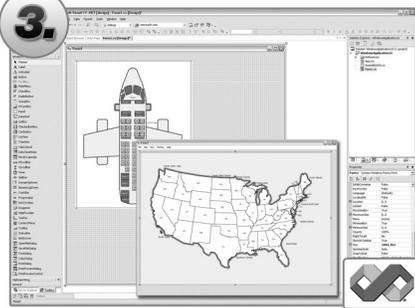
1A vacant
1B vacant
1C reserved
1D reserved
2A reserved
2B vacant
2C vacant
2D reserved
3A vacant
3B reserved
3C vacant
3D reserved



Connect to your data source and feed the map.

A data source such as an XML file or a database is all you need to populate individual or groups of objects in the map. Once you have loaded your data, Chart FX Maps will read the object names in the map file and match them to the labels in your data source for further access from Visual Studio .NET.

3.



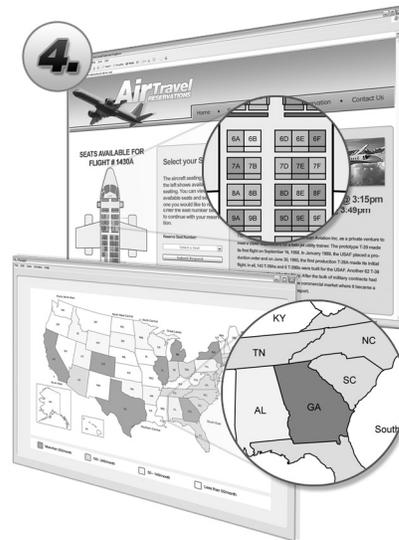
Define your rules and customize map options with Visual Studio .NET.

Chart FX Maps integrates seamlessly to Visual Studio® .NET and allows you to use the powerful C# or VB.NET languages to set conditions that will be reflected in the map as visual and dynamic attributes such as the fill color, border and hyperlink settings that end users can recognize and interact with.

Deploy your application and let your users visualize and explore data!

Chart FX Maps can not only be integrated to Windows Forms applications but provides a powerful ASP.NET server side component that allows you to display dynamically created images in browser based applications. Also, a powerful .NET client component allows users to interact with the map by summarizing data, zooming portions of the map and even drill-down to other related maps.

4.



Selecting or Creating Your Map

Chart FX Maps provides a full library of geographical maps that you can quickly integrate and deploy royalty-free. They are:

World	----->	Continents	----->	United States
Europe		3-Digit Zip Codes		
Asia		5-Digit Zip Code***		
Africa		Counties by State		
North America Districts***				
South America				
Oceania				
Antarctica				

*** Contact Software FX for availability.

Although Chart FX provides an extensive library of geographical maps, there will be many situations where you may want a map of something that we do not have available. For example, a network or a stadium seating diagram. This does not mean, however, that you cannot use Chart FX Maps.

Creating custom SVG images is easy and fully documented in the Chart FX Maps Help system. All you need is a standard SVG editor, such as Adobe Illustrator®, or, if you prefer, you can commission Software FX to create the vector map for you.

For additional information please contact us at sales@softwarefx.com or visit our web site at www.softwarefx.com.

Populating the Map

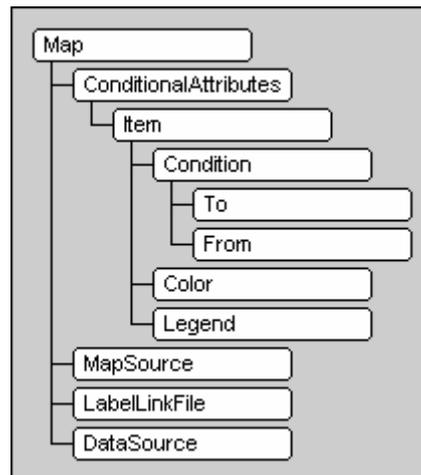
Once the map has been selected or created you can use Visual Studio .NET to connect to most popular data sources like text and XML files as well as most popular database engines.

Chart FX Maps will match objects in the map to the labels in your data source. If the data source labels do not match the naming convention on the SVG file, an XML conversion file can be used to prevent unnecessary changes to the data source or the SVG map.

Setting the Map's Visual Attributes

There are many conditional and per-point attributes you can use to highlight objects in the map; including colors, borders, gradients, point labels, hyperlinks, patterns, drilldowns and much more!

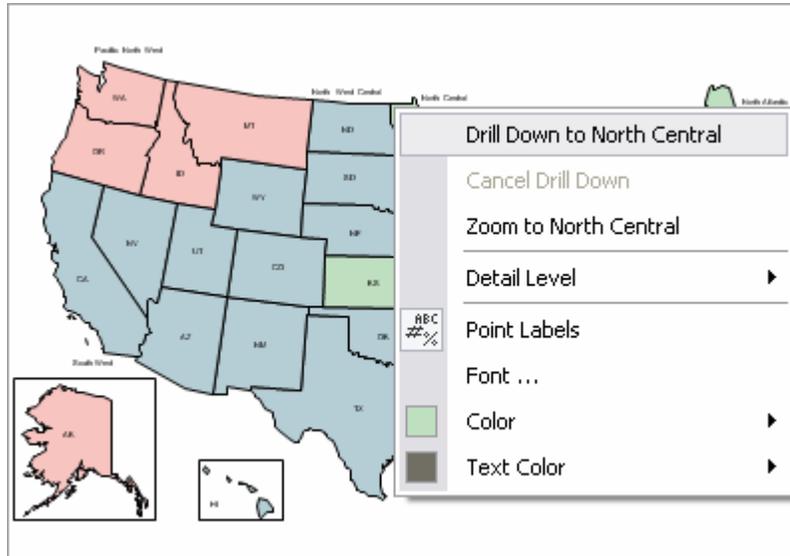
The following piece of pseudo code demonstrates how easy is to create a map of Quarterly Sales by State. Note that maps may be populated from any of the supported data providers such as ADO.NET recordsets, XML files, etc.



End User Interaction

At runtime, Chart FX Maps provides an intuitive user interface that allows end users to further customize the map view. End users may zoom, drilldown, adjust detail level and modify object attributes (point labels, font, and colors) by simply right clicking objects in the map and accessing a context menu and toolbar provided by the Chart FX Maps client control.

The following image illustrates Chart FX Maps end user interaction in the context of a browser based application:



Additionally, end users may select other views such as bar or line charts that also inherit the same visual attributes in the map, improving the data visualization and analysis capabilities of your application.

Note: This interaction is possible in applications when rendering maps as objects, not images.

Chart FX Maps and Internet/Intranet Applications

Not only has Chart FX Maps been developed with the server implementation in mind, but its small memory footprint, browser independence and scalability makes it the perfect tool for any commercial or corporate grade application. Also, the Chart FX Maps client control can be easily and securely deployed across the organization via a single file download from Internet Explorer.

Performance wise, Chart FX Maps can serve a multitude of end users with sub-second maps; and the solution is fully scalable and compatible with intricate server architectures, such as web farms.

Adding Chart FX Maps in a VS.NET Application

Open Visual Studio .NET and select the New Project from the File Menu.

Add Chart FX Maps to your toolbox.

Selects Tools-> Customize Toolbox.

Select the .NET Framework Components tab and check the Map control checkbox. Important: use the SoftwareFX.ChartFX.Map.WinForms namespace if you created a Windows Forms applications; or SoftwareFX.ChartFX.Map.Server if you want to use Chart FX Maps in a Web Forms project.

Once you have selected the chart control, click OK.

To integrate the extension using Design-Time controls: Create a Windows Form or Web Forms Charting Application within VS.NET and select the Map control from your Toolbox and drop it to your form or page.

Chart FX OLAP

Companies worldwide use relational databases and OLAP to increase the productivity of business managers, developers, and whole organizations. OLAP enables organizations to respond more quickly to market demands. Market responsiveness, in turn, often yields improved revenue and profitability.

Developers benefit from OLAP for a simple reason: end users of OLAP applications can become self-sufficient since they do not have to rely on IT to make schema changes, to create joins, or build their own models.

Unfortunately most, if not all, OLAP vendors provide standalone applications that are expensive, proprietary and provide limited deployment options. If you want to OLAP-enable your intranet of enterprise applications, read on—Chart FX OLAP may just be the right solution for you!

What is Chart FX OLAP?

Chart FX OLAP is the first .NET OLAP front end seamlessly integrating with Visual Studio .NET allowing you to build and deploy fully OLAP provisioned charts in your Windows Forms and Web Forms applications.

Chart FX OLAP virtually connects to any relational database through ADO-MD and XML for Analysis serving as the only data analysis charting solution geared for developers and integrating to major leading OLAP server vendors such as Microsoft SQL Server®.

Chart FX OLAP picks up the information and presents the hierarchical structure to the user in an intuitive way, enabling users to pivot, slice and dice data to uncover vital information. All this end user functionality is provided with little or no intervention from the developer, thus reducing development time and increasing customer satisfaction.

Chart FX OLAP is a sensibly priced solution that supports a myriad of deployment scenarios; whether you are building a browser based intranet application or a commercial or enterprise grade application, Chart FX OLAP is the right solution for you!

Chart FX OLAP Features

Menus on Demand

Each dimension tab features an icon that provides access to different options such as sub categories selection and summary functions. This icon provides immediate and intuitive access to each dimension subcategories.

Toolbar

Chart FX OLAP Toolbar provides end users access to powerful charting functionality such as 3D, rotation, axis settings, chart gallery, printing, and exporting straight from the browser or application!

Automatic Legends and Titles

Chart FX OLAP assigns Legends and Titles automatically according to the selections made by end users in the category bar.

Color Persistence

Chart FX OLAP provides a state-of-the-art color coding algorithm that renders chart markers and icons with unique and identifiable colors throughout the analysis process, considerably improving the chart's readability.

Drag & Drop Pivoting

Pivot or transpose data in the chart by dragging the Legend box or X-axis labels.

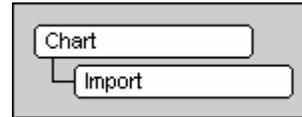
XP and Windows 2000 Category Bar

Located in the left hand side of the chart allows users to navigate from all available dimensions and levels contained in the OLAP cube. For added convenience, this interface accommodates Windows XP, Windows 2000 and browser views. All elements are accompanied with a checkbox that enables end users to easily filter additional data displayed in the cube. In addition, every user interaction is automatically and immediately displayed in the chart.

Connecting Chart FX OLAP with sample data

For evaluation purposes, we have included a portable binary file that allows you to test Chart FX OLAP free from the bounds of connectivity to a real OLAP data source. This binary file contains multidimensional information from the Food Mart 2000 database, which is included with Microsoft SQL Server: Analysis Services package.

On a production level, however, you will be required to use ADO-MD or XML for Analysis to populate the chart with real multidimensional data. Included with the Chart FX OLAP installation, you will find a sample directory where a file named OLAPCHART.BIN is located. In order to test Chart FX OLAP, you will need to instruct Chart FX OLAP to load this file, using the Import method:



Connecting to Multidimensional Data Sources

If you are interested in populating Chart FX OLAP with data coming from your own OLAP database, you can do it via ADO-MD or XML for Analysis. If you are using SQL Server with Analysis Services, your best option is to use an ADO-MD connection and let Chart FX OLAP read from it automatically. The following pseudo code snippet can help you connect Chart FX OLAP to your existing database:

Note: For additional information on XML for Analysis please refer to the Chart FX OLAP API Reference.

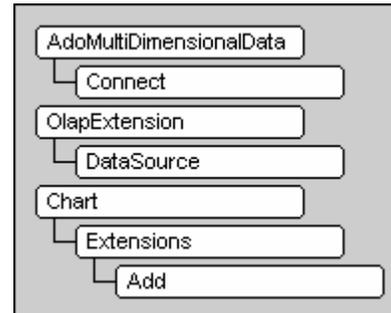


Chart FX OLAP and Web Forms Internet/Intranet Applications

Not only has Chart FX OLAP been developed with the server implementation in mind, but its small memory footprint and scalability makes it the perfect tool for any commercial or corporate grade application. Also, the Chart FX OLAP client control can be easily and securely deployed across the organization via a single file download from the Internet Explorer browser.

Performance wise, Chart FX OLAP can serve a multitude of end users with sub-second charts and the solution is fully scalable and compatible with intricate server architectures, such as web-farms. The picture below depicts a browser-based intranet application using Chart FX OLAP.

Adding Chart FX OLAP in a VS.NET Application

To add Chart FX OLAP, you must perform the following steps:

Open Visual Studio .NET and select the *New Project* from the *File* Menu.
Add Chart FX for .NET to your toolbox.
Select Tools -> Customize Toolbox.
Select the .NET Framework Components tab and check the Chart control checkbox. Important: use the SoftwareFX.ChartFX namespace if you created a Windows Forms applications; or SoftwareFX.ChartFX.Internet.Server if you want to use Chart FX OLAP in a Web Forms project.
Once you have selected the chart control, click Ok.
Draw the Chart FX control to the form. If the Chart FX for .NET wizard pops-up select a side-by-side bar chart from the wizard gallery and exit the wizard pressing the Finish button.
Chart FX OLAP acts as an extension to Chart FX for .NET. Therefore you need to add the OLAP Extension references to the project. Select Project -> Add Reference. From the Component list, select the Chart FX OLAP Extension, Chart FX Category Bar Extension, and the Interop.ADOMD components.

Chart FX Statistical

In the global business and economic environment of today, vast amounts of business data are readily available. In fact, data is being collected and stored at ever increasing rates.

The most successful managers and decision makers are the ones who can understand the data, determine what useful information may be hidden within it and ultimately use that information effectively.

Some of the most crucial information a store of data may hold is its collection of statistical properties and trends. Statistics are used to analyze all aspects of business from markets to quality to processes and even people.

The business leaders of today's success stories know what information they want from their business data. Chart FX Statistical was designed with just these individuals and the developers they work with in mind!

If you think Chart FX Statistical could give you a fresh perspective on your organization's information, Read On!

What is Chart FX Statistical?

Chart FX Statistical extends the power of Chart FX by adding statistical analysis capabilities to your charting applications.

Chart FX Statistical works with a chart's data to give deeper insight by complementing the core functionality of Chart FX with an assortment of studies organized into groups. Developers can devote effort to application specific tasks and let Chart FX Statistical handle mathematical operations and algorithms to realize information including Standard Deviation, Variance, Regression Line, Normal Distribution and F-Test to name a few.

In addition, a collection of calculations have been assembled that can easily be applied to chart data; Developers may even choose to perform tests or apply the supported analysis studies to chart data using this feature.

To present these features to the end user, Chart FX Statistical expands its gallery object with new members for displaying statistical information including Box Plot, Histogram and Regression.

Analysis Studies

Chart FX Statistical has defined a large set of statistical analysis formulas to allow you to concentrate on your application's business logic and leave the mathematical equations to be handled by Chart FX Statistical. These formulas have been logically classified in to three categories: distribution formulas, analysis formulas and test formulas.

There are a number of common situations that require statistical distributions. One common use is to evaluate the confidence limits for a fitted model, like the mean of a Normal distribution. Chart FX Statistical defines a number of distribution formulas including: Chi Square, Cumulative-Normal, Cumulative-T, F, Inverse-Cumulative-Normal, Normal and a number of others.

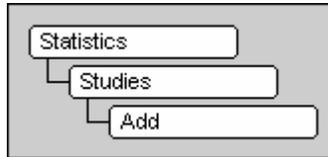
The analysis studies, like linear regression, form a group of studies that deal with the classic statistical problem of determining a relationship between multiple random variables. Chart FX Statistical defines many different formulas to deal with various types of data. The "Descriptive Statistics" group defines formulas on a single sample of data like median, mean, mode, standard deviation, etc.

Chart FX Statistical can also help you to make inferences about the relationship between two independent variables. Examples of this variety include Covariance, Pearson Correlation Coefficient, Linear Regression, etc.

Another important subgroup of analysis studies is Statistical Process Control (SPC). Chart FX Statistical has included a number of SPC calculations including Sigma, Ppk and Weco.

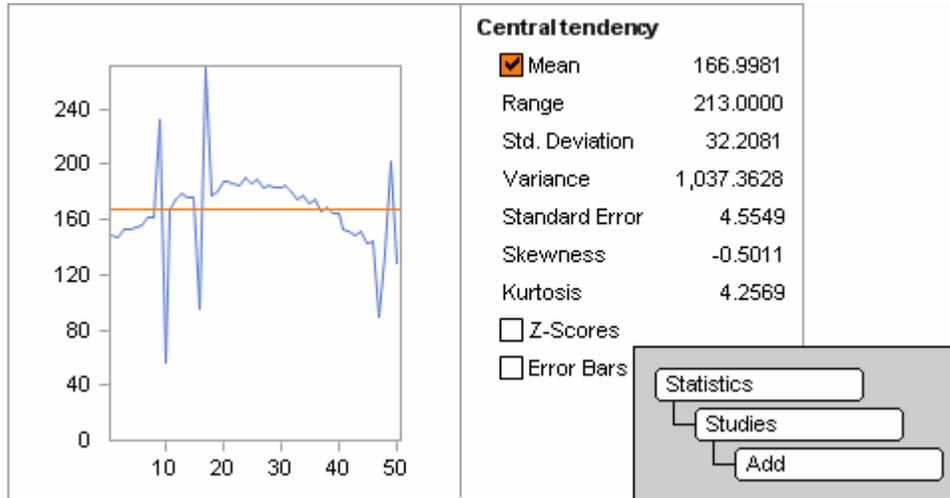
The last category defined by Chart FX Statistical is the test analysis calculations. These tests are used to test for a certain relationship between the data and return a value indicating whether or not they passed the test. The members of this group include RegressionFTest, RegressionTTest and Weco.

Although these calculations would be complicated to code from scratch Chart FX Statistical has made it very easy to include these studies to your application, for example:



Study Groups

As there are so many different studies that can be applied to your data, Chart FX Statistical created a number of "prefab" study groups to help you rapidly integrate the frequently used studies into your charts. These groups allow you to include a number of studies with very little code! For example:



Statistical indicators have been grouped according to end user preferences.

Central Tendency Median
 Median
 Lower Quartile
 Upper Quartile
 IQR

Central Tendency Mean

Mean
 Std. Deviation
 Variance
 Standard Error
 Skewness
 Z-Scores
 Error Bars

X/Y Correlation
 Linear Regression
 Pearson's Coefficient
 Coefficient of
 Determination
 Standard Error of
 Estimate
 F-Statistic (Regression)
 t-Statistic (Regression)

Two Populations

Equal Variances F-Test
 Equal Means t-Test

Matched Populations
 Equal Variances F-Test
 Equal Means t-Test

Anova

Sum of Squares
 Treatments
 Sum of Squares Error
 Sum of Squares Total
 d.o.f. Treatments
 d.o.f. Error
 d.o.f. Total
 Mean Square Treatments
 Mean Square Error
 F-Statistic (Anova)

Calculations

In many applications, developers may choose to retrieve actual values from the supported statistical functions and tests rather than simply displaying values in the legend or chart area.

Chart FX Statistical includes added support to gain access to these values by providing a Calculators feature. This feature provides developers with additional statistical functions as well as access to the previously discussed analysis studies and tests.

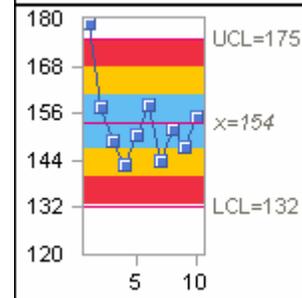
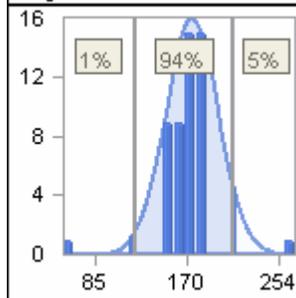
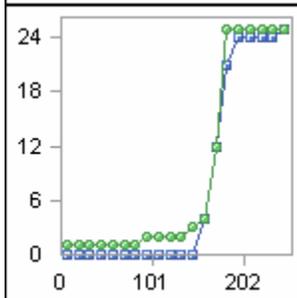
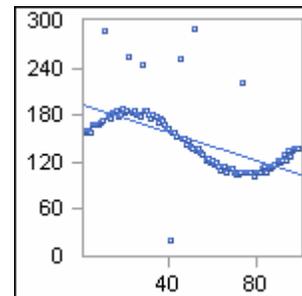
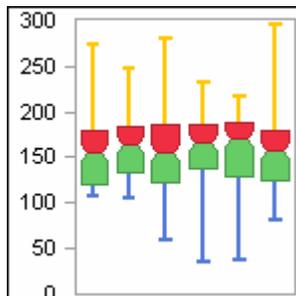
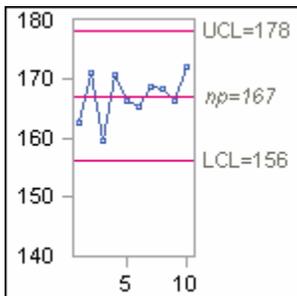
Additional Statistical **Calculators** include:

CalculateMeanInterval	CalculateRegression
CalculateStandardDeviationInterval	CalculateVarianceInterval
CalculateZScore	CalculateZScoreInverse

Utilizing this implementation, developers may obtain values from samples passed to the chart for further comparison and processing logic.

Statistical Galleries

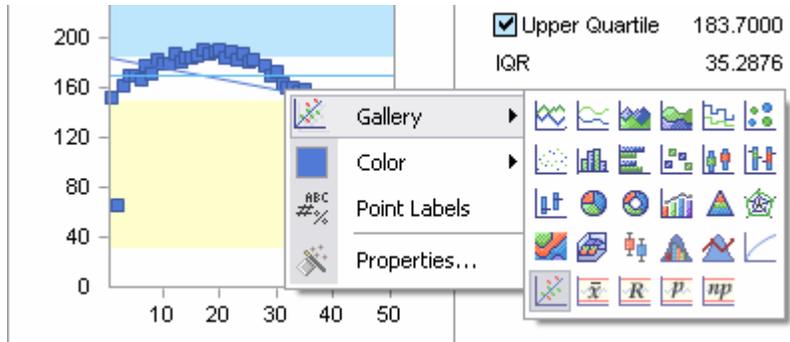
Chart FX Statistical Galleries are designed to give the best possible presentation to the end user. These include the BoxPlot, Frequency Polygon, Histogram, Regression, Ogive, and the SPC chart galleries.



End User Interaction

At runtime, Chart FX Statistical provides an intuitive statistical legend that allows end users to view calculations and interact with selected analysis studies. By interacting with the legend, end users may enable studies and distributions in the chart area. Also, as with all the Chart FX Extension, the chart's default user interface controls are available. Once Chart FX Statistical is added to an existing chart, the statistical gallery types are available from the toolbar and context menu controls.

The following image illustrates Chart FX Statistical end user interaction in the context of a browser based application:



Note: This interaction is possible in applications when rendering maps as objects, not images.

Chart FX Statistical and Internet/Intranet Applications

Not only has Chart FX Statistical been developed with the server implementation in mind but its small memory footprint, browser independence and scalability makes it the perfect tool for any commercial or corporate grade application. Also, the Chart FX Statistical client control can be easily and securely deployed across the organization via a single file download from Internet Explorer.

Performance wise, Chart FX Statistical can serve a multitude of end users with sub-second statistical charts and the solution is fully scalable and compatible with intricate server architectures, such as web farms.

Adding Chart FX Statistical to a Visual Studio .NET Application

Open Visual Studio .NET and select the New Project from the File Menu.
Add Chart FX for .NET to your toolbox.
Selects Tools-> Customize Toolbox.
Select the .NET Framework Components tab and check the Chart control checkbox. Important: use the SoftwareFX.ChartFX namespace if you created a Windows Forms applications; or SoftwareFX.ChartFX.Internet.Server if you want to use Chart FX Statistical in a Web Forms project.
Once you have selected the chart control, click Ok.

To integrate the extension using Design-Time controls

Create a Windows Form or Web Forms Charting Application within VS.NET. Select the Statistical control from your Toolbox and drop it to your form or page. This should create a Statistical Extension icon at the bottom of your design pane.

Add a chart control to your form or page. Access the Statistics Properties dialog and configure the Chart property of the Statistics object to the name of the chart object you wish to add the extension to, i.e. chart1.

Note: Due to a limitation in VS.NET, if you add the Chart before adding the Statistical extension to the project, you must save and close the project before you are able to configure the Chart property of the statistics object.

Chart FX Financial

Banks, brokerage firms, investment firms and other financial companies perform technical analysis on financial data and display such data using a variety of predefined financial indicators and special chart types. Chart FX Financial allows programmers and developers to integrate powerful financial charts within their applications quickly and easily, without laboriously hand coding each financial indicator and analysis study. This product has been designed to provide a vast variety of predefined financial indicators and special chart types to perform Technical Analysis on financial market data. Just plug in and you are ready to start analyzing graphically.

What is Chart FX Financial?

Chart FX Financial extends the power of Chart FX by adding financial analysis capabilities to your charting applications.

Chart FX Financial works with a chart's data to give deeper insight by complementing the core functionality of Chart FX with an assortment of financial indicators and analysis tools. Developers may now concentrate time and energy toward application specific tasks while allowing Chart FX Financial to handle the various complex algorithms to calculate such things as Price Indicators, Technical Indicators and even Interactive drawing features.

In addition, a collection of calculations have been assembled that can easily be applied to chart data; developers may even choose to perform tests or apply the supported analysis studies to chart data using this feature.

To present these Analytical Chart Features, Chart FX Financial expands its gallery class with new members for displaying financial information including Renko, Close, Close Volume, Open High Low Close, etc.

Financial Chart Types

Chart FX Financial supports two main categories of financial charts, Analytical Charts and Renko Charts. Please read below for specifics of each financial chart type.

Analytical Charts

Chart FX Financial provides this chart type where many indicators, both price and technical, can be displayed in a chart. Because it usually contains more data than just closing prices (like open, hi, lo and volume) many other indicators can be displayed in the chart. These indicators usually enhanced the Analytical capabilities of the candlestick or hi-lo-close charts. In Chart FX Financial, we provide many advantages in Analytical Charts, among them:

- Price Indicators that are calculated automatically.
- Technical Indicators that can be used to enhanced the chart analysis.
- Technical Indicators may be displayed in a single screen.
- Custom Interactive Drawing features that allow the user to select different areas in the chart to perform and display additional Technical Analysis information (i.e. Fibonacci Fans, Arcs, etc)

An Analytical Chart is an [Open]-Hi-Lo-Close or Candlestick chart which displays Price indicators in the section where the main chart is displayed and Technical indicators in a separate chart. The beauty of Chart FX Financial is that it exposes both an API (Application Programming Interface) and a User Interface that allows developers as well as end users control and display both technical and price indicators.

Renko Charts

As mutual fund prices are based only on the closing price and candlestick charts require the open, hi, low and closing prices. Japanese charts, like Renko, are particularly useful for analyzing stock mutual funds. The main advantage is ease of interpretation of bullish and bearish signs in the stock.

The Renko chart looks similar to the Three-Line Break chart since they both have blocks. The individual blocks that form the Renko chart are sometimes referred to as bricks. For a Renko chart, a block is drawn in the direction of the prior move if a fixed amount has been exceeded. For example, if there is a white brick on the Renko chart, the stock has to advance by a predetermined fixed amount before a new white brick can be drawn.

Point & Figure Charts (P&F)

Point and figure (P&F) charting concentrates on price and its changes by eliminating other elements such as time and volume. This chart is popular because it is simple and provides the following advantages:

- Point & figure charting uses price data with minimum manipulation.
- P&F charts are indigenous to stock market trading which give you both the immediate and long term view.
- Formations, patterns and signals are easy to recognize and interpret, and tend to repeat themselves.
- It enables the investor to do what he should do - stay on a winner while it is winning and get off a loser quickly.
- The fixed interval of plotting may be made to fit whatever you want to chart - stocks, bonds, commodities, speculation or investment.
- It ignores the static time factor, the confusing volume indications, and irrelevant minor fluctuations.

The basic point and figure chart shows columns of X's and O's which display the underlying supply and demand of prices. A column of X's means the stock price or index is going up. A column of O's means the stock price is going down. Columns of X's and O's alternate back and forth. However, they never appear in the same column.

Three Line Break Charts

As opposed to Analytical charts, Three Line Break charts ignore the passage of time. They specialize in trend analysis, which is similar to P&F charting. Their main advantage is that there is no arbitrary fixed reversal amount; it is the market's action that will give the indication of a reversal.

The Three-Line Break chart looks like a series of black and white boxes of varying heights (Although in Chart FX Financial you can assign different colors to the boxes). A new block is in a separate column. Each of these blocks is called a line. Using the closing price, a new white box is added if the previous high is exceeded and a new black line is drawn if the price reaches a new low. If there is neither a new high nor a low, nothing is drawn.

If a rally is powerful enough to form three consecutive white lines, then the low of the last three white lines has to be exceeded before the opposite color line is drawn. The term "Three line Break" comes from the fact that the market has to "break" above the prior three lines before a new opposite color line is drawn.

Kagi Charts

The goal of the Kagi chart is to catch longer term trends. As mutual fund prices are based only on the closing price and candlestick charts require the open, hi, low and closing prices. Japanese charts, like Kagi, are particularly useful for analyzing stock mutual funds.

Using a Kagi, it is easy to obtain an insight into whom has the balance of power- the bulls or the bears. They are also very easy to interpret, buy when line goes from thin to thick and sell when the line goes from thick to thin.

Kagi charts display a series of connecting vertical lines where the thickness and direction of the lines are dependent on the price action. If closing prices continue to move in the direction of the prior vertical Kagi line, then that line is extended. However, if the closing price reverses by a pre-determined "reversal" amount, a new Kagi line is drawn in the next column in the opposite direction. An interesting aspect of the Kagi chart is that when closing prices penetrate the prior column's high or low, the thickness of the Kagi line changes.

Passing Data to Chart FX Financial

The required data varies according to the Chart Types you want to display and the indicators you want to show in financial charts. For example, to make a Kagi Chart, you'll only need the closing price and the date associated with each data point. On the other hand, to display an Analytical Chart such as a open-hi-lo-close with volume and several technical indicators, you'll need the respective data required for that particular chart as well as for the selected indicators.

It is likely that you want to be able to display all chart types and all price indicators Chart FX FE supports, it is also likely that your data source contains all the necessary data required by the Chart FX Financial. Therefore, we strongly suggest you always pass the following data to the extension: **Open, Hi, Lo, Close, Volume, Date.**

When this data is properly loaded, you'll be able to get the most out of Chart FX Financial as all charts, Technical and Price Indicators will be available to you and your users.

Chart FX Real-Time

Real-time systems must, without fail, provide a response to some kind of event within a specified time window. For example, manufacturing process controls, medical monitoring equipment, telecommunications and aircraft controls are practical examples of real-time applications that have greatly contributed to the progress and prosperity of mankind. These applications are characterized by strict fault-tolerance guidelines that force them to employ operating systems and specific hardware devices with special device drivers to ensure promptness and accuracy.

The Internet has created new and exciting opportunities for businesses to embrace a new generation of computing solutions. But it was designed as a general-purpose platform that is not as deterministic as a real-time system, even though it has the capability to provide very fast response times.

In general, the abovementioned real-time applications are not good candidates for Internet adaptation and deployment. However, technology advances are allowing the possibility of more demanding Internet applications that are often called upon to manage time-critical responses. In other words, to produce results in "real time".

These applications have reduced constraints, but still must respond quickly within specific time boundaries. That is, they must service events quickly enough so that the required response window is satisfied. In this sense, it is possible for real-time applications to be served over the Internet. This can significantly improve the responsiveness and increase customer satisfaction by connecting people to real-time information anytime, anywhere.

For example, in the consumer market, Instant Messaging (IM) is one of the fastest growing Internet communication mechanisms because it's convenient, simple and productive. IM is in real-time. Likewise, a real-time warehousing application will give organizations an unprecedented competitive advantage and Real-Time stock market data feeds have become the norm among day traders and investors. None of these applications could be implemented without the networked and distributed nature of the Internet.

As the leading front-end data-analysis and reporting tool, Chart FX allows organizations to quickly and easily integrate charts into their Internet/intranet applications for over a decade. Now, with Chart FX Real-Time, developers will be able to meet the pervasive need of accessing real-time information and respond to today's fast paced business environment.

You will learn that Chart FX Real-Time provides server-side in-memory real-time data storage to boost server performance and application scalability. Also, it provides an agent specification that allows smooth real-time data interchangeability between heterogeneous systems. Finally, the product provides support for .NET and COM controls that can display real-time data on the browser without manually refreshing, making Chart FX the complete charting solution for real-time Internet applications.

What is Chart FX Real-Time?

Chart FX Real-Time is a complete software solution that allows developers to build modern, scalable, multi-tier real-time Internet applications using Microsoft technologies like Internet Information Server, COM, .NET and Internet Explorer.

Performance and Scalability

While the ultimate goal is to display a chart repainting on the browser, a real-time Internet charting solution entails more than just a client control pulling data from a web server. Using just client controls is, at best, inelegant and, at worst, unreliable and prone to stop responding.

One of the disadvantages of pull mechanisms used by client controls is that they repeatedly make requests to the server (data feed) even if there haven't been any updates or new data to be plotted. This poses serious performance and scalability issues as the user load increases.

Chart FX Real-Time features HeartBeat™, a new server-side in-memory real-time data storage technology from Software FX that allows information to flow rapidly and accurately between the server and the browser. Chart FX Real-Time can read real-time data from the HeartBeat™ in-memory repository transparently; allowing your server to respond instantaneously to browser requests without significantly impacting server performance thus maximizing the number of concurrent users your application can serve.

Distributed Computing

Chart FX Real-Time is the first charting solution to fully embrace and integrate the Internet model of distributed computing solutions. Essentially, Chart FX Real-Time partitions functionality into separate components that can be deployed on the network in a wide variety of physical configurations.

For example, HeartBeat™ provides a state-of-the-art agent specification based on protocols like SOAP and XML. These agents can reside locally or remotely and funnel data from the source to the Chart FX Real-Time Server. These agents establish an asynchronous connection to the HeartBeat™ Real-Time Server. Examples of asynchronous data scenarios include database queries, complex calculations, and slow data retrieval. Since the connection speed and network traffic can often slow down access to web pages, it is nice that data access can be done asynchronously so that the application doesn't have to be locked while reading from the data source.

Easy to Use and Integrate

Because Chart FX Real-Time is based on COM, .NET and open standards, developers can use any language or tool to integrate real-time charts into their Internet/intranet or client-server applications. Through COM and .NET, developers can integrate Chart FX Real-Time in any part of their application via pluggable software components that work seamlessly in Visual Studio, Visual Basic and other popular development tools.

Security

HeartBeat™ provides a wide range of possibilities for security configuration. Taking advantage of the Internet Information Server security, and in combination with its own security implementation, HeartBeat™ allows a developer to define multiple security scenarios at server, agent and client levels.

Fault Tolerance

HeartBeat™ is server-side in-memory real-time data storage. Real-time data is not meant to be kept for long periods of time; however, memory as storage, can be volatile and losing that "short term" collection of data may have a huge impact on a real-time application. HeartBeat™ is prepared to react to unpredictable situations, including mechanisms that will preserve its configuration, which can help to avoid a major loss of data.

Chart FX Wireless

The Chart FX Wireless Extension allows developers to create wireless applications in a minimal amount of time without the need of intensive programming knowledge. With an easy-to-use, flowchart-style interface, a developer can easily create an application's logic, connect with any database and then automatically generate server and client-side code.

What is AireLogic?

AireLogic is an IDE that gives a developer the ability to create wireless applications in a minimal amount of time without the need of intensive programming knowledge. With AireLogic's easy-to-use, flowchart-style interface, a developer can easily create an application's logic, connect with any database and then automatically generate server and client-side code.

AireLogic Features

Automation

AireLogic automates many of the time consuming processes that hamper wireless application development. First, wireless code is automatically generated via proprietary algorithms for multiple server and client side technologies. Second, the application incorporates real time debugging to automatically identify logic and program development errors. The product's automation feature makes wireless application development faster, more effective and most efficient.

Modularity

AireLogic was designed based on a modular architecture. The architecture utilizes standard building blocks, modules that are referred to as components based on Microsoft's Component Object Model ("COM ") and .NET objects, which can be created, duplicated or modified to create an endless variety of robust, fully functional and platform independent wireless applications. Modularity allows new libraries to be imported or created as necessary. Enhanced server functionality is enabled and the modular architecture also allows for program adaptability to any future situation. For example, if the server side language needs to be enhanced, or the emulators need to be changed to accommodate future changes in wireless standards, it can be accomplished without affecting the rest of the application. In addition, extra COM modules, .NET objects and ActiveX controls can be added without changing the product's core technology.

Scalability

By adding or removing the basic building block modules, AireLogic can be utilized to create applications that are as simple or as complex as the developer wishes them to be. Additionally, standard modules, or sequences from other applications, can easily be copied and replicated to other applications without the need to regenerate applications or subsequent code manually. The application's modular design makes it completely scalable to any future changes in wireless standards and server standards.

Agnosticity

AireLogic is capable of generating code compatible for a number of server architectures. Scripting languages AireLogic supports include MS Active Server Pages, Java, Perl and .NET. The IDE also utilizes a variety of wireless protocols including WAP, HTML and SMS allowing for virtually unlimited deployment possibilities to a user's cellular phone, PDA or pager. Additionally, the product can be integrated with existing Internet security measures and middleware, which distinguishes AireLogic as the leading enabler of wireless software applications.

Just-in-Time Compilation

AireLogic's proprietary integrated emulator (PC program that simulates the wireless device on which the application is to run) and debugger allow for virtually error-free applications through three levels of scrutiny. At the first level, applications are checked on a card-by-card basis as they are created. Any cards containing an error are clearly highlighted and can be corrected immediately. The second level occurs when the application is run on the emulator. The IDE highlights each card being executed and allows the user to view the code that is generated in a window at the bottom of the screen, helping to identify errors in a complex application that may otherwise take significant time to locate. The final level of error-checking is made possible by the emulator's ability to simulate complex variables and databases which allow the application to be tested as a whole to ensure it functions as it was intended. AireWeb is the only wireless application development technology with an integrated debugging environment.

Usability

AireLogic's graphical user interface, automation feature, modular design and comprehensive menus make its application enabling software easy to learn and use. AireLogic's proprietary algorithms automatically convert flowchart information into programming code, reducing the need to hire expensive and typically unavailable programmers. This enables wireless applications to be created quickly by minimally trained application developers who understand flowchart logic. Experienced programmers with sophisticated coding skills are not needed. The integrated emulator and debugger provide just-in-time compilation allowing even novice users to produce work with minimal errors.

Collaborative Development

AireLogic's modularity permits multiple developers to create an application collaboratively. Each developer can work on a separate module, bringing them together at the end to form a complete application.

Chart FX Wireless Features

Besides all the features inherent to Chart FX, the Wireless Extension provides:

Auto detection, a.k.a. Write once, run everywhere

The Chart FX Internet Wireless Extension adds mobile capabilities to Chart FX core by generating the chart type required by the accessing device. The extension automatically detects the device's features and capabilities, such as size, colors, etc. and generates the chart accordingly. This feature allows chart rendering for different devices with minimal coding.

Appendix A - Globalization

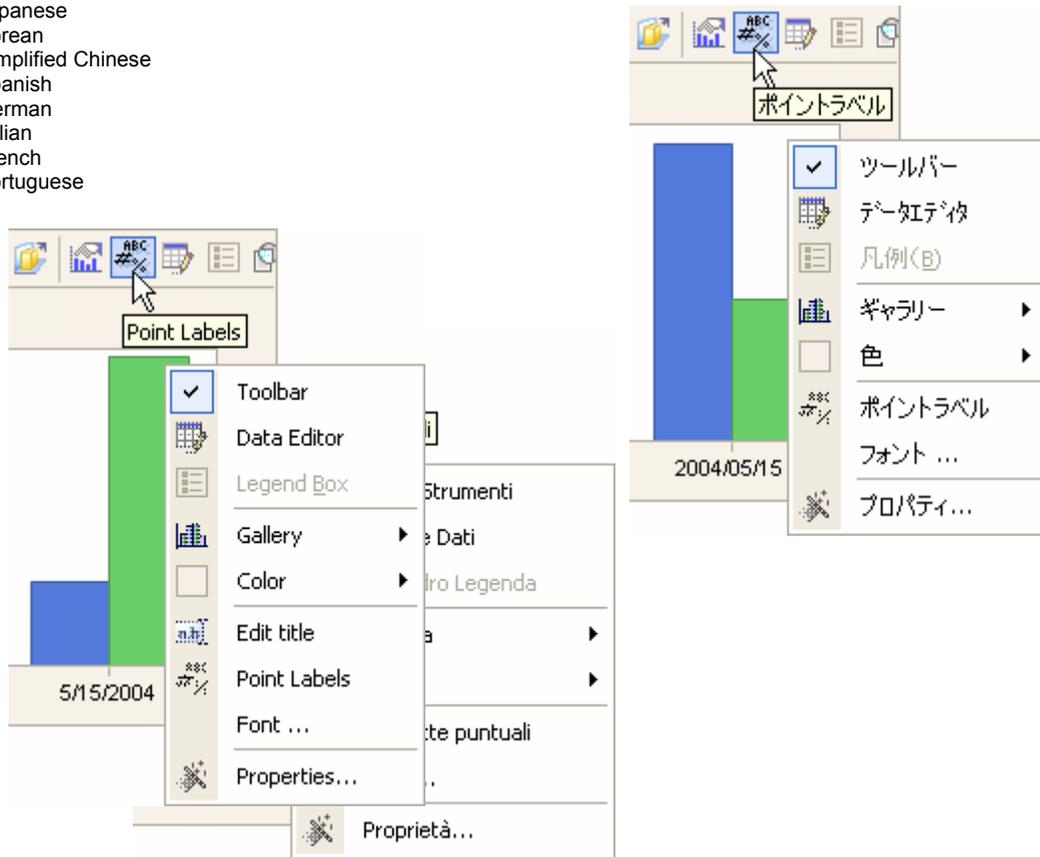


Globalization Overview

Globalization is the process of designing and developing an application that supports localized user interfaces and regional data for users in multiple cultures. Chart FX can be localized to any language or culture, including the writing system, calendars in use, date and time formatting conventions, numeric and currency conventions, and sorting rules.

Chart FX provides localization resources for a number of cultures. However, you can create your own localization resources, for any language. The following localized resources are currently included at no extra cost. Please refer to the electronic documentation for details about how obtaining and/or create your own localized resources.

Japanese
Korean
Simplified Chinese
Spanish
German
Italian
French
Portuguese



Appendix B - Printing



Introduction

The **Printer** object provides important and very helpful properties to print a chart. Using this object, you will have control over the margins, paper orientation, color or pattern printing, among others. The supported methods will also allow you to prompt the user with page setup, preview and print dialogs to ensure the user prints exactly what they want.

Setting the Page Layout

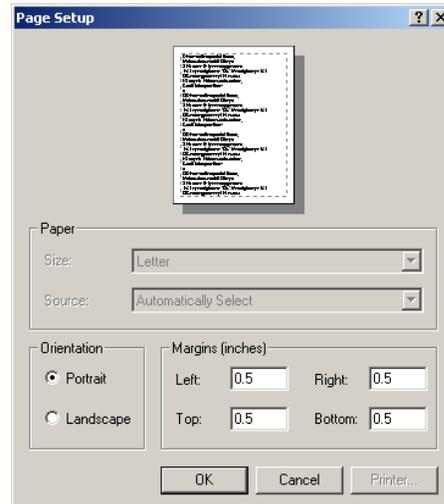
When you are ready to add printing functionality to your chart, there are questions you should take into account:

Would you like to use the default printer margins or specify custom margins?

Should the printed chart use a landscape or portrait page orientation?

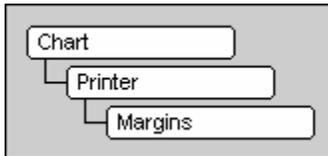
Should you prompt the user with a preview or setup dialog to make these choices themselves?

Chart FX for .NET supports properties and methods that will allow you to specify exactly the page layout desired.

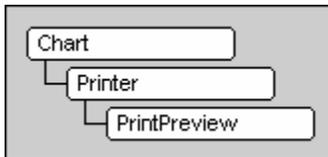


Margins

If no margins are set for a chart, the default printer margins will be used when the chart is printed. Chart FX allows you to specify the top, bottom, right and left margins using the **Margins** property. The margins property is set using an integer and is measured in hundredths of an inch; therefore, if you would like to set an inch margin around the chart, each margin must be set to 100, using the following API calls:



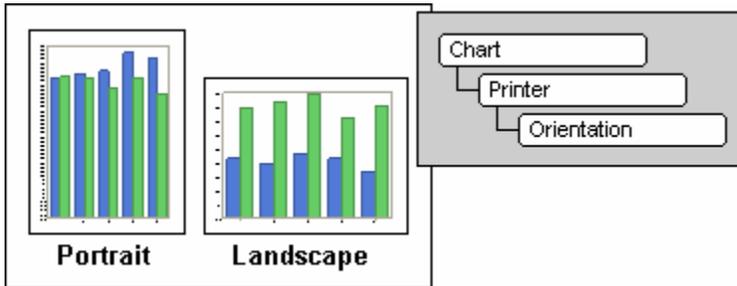
If you would rather allow the user to configure the printing margins, you may do so by calling the **PageSetup** method. The PageSetup method will prompt the user with a page setup dialog box (shown in the End User Experience section) that allows end user access to the page margins and page orientation (Landscape/Portrait). The print preview dialog may also be prompted to the user by calling the **PrintPreview** method, with the following API:



Orientation

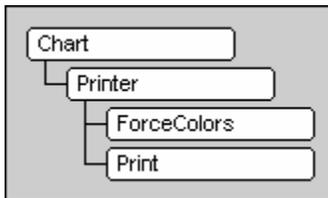
From the page setup dialog, the end user may select to print the chart in either the landscape or portrait formats. The landscape format is longer from left to right, while the portrait format is longer from top to bottom.

You can also set the page orientation programmatically using the **Orientation** property, using the API calls below:



Printing Colors

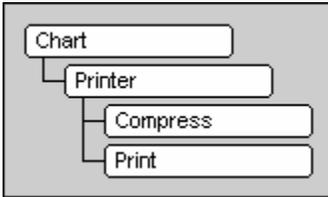
Normally if you have a black and white printer and try printing a color chart, you will receive a printout that is virtually unreadable due to the grayscales that are printed instead of the true palette colors. To resolve this type of issue, Chart FX includes the **ForceColors** property. When a color chart is printed to a B&W printer while this property is set to "False", each color in the chart is printed as a hatched pattern. This helps the end user read different series in the chart based on the contrasting patterns created in the printout. If the chart is to be printed with a color printer, you can force Chart FX to print a color chart by setting this property to "True". The API below should be used to set the ForceColors property and print the chart:



Note: The ForceColors property will have no effect when the Chart is printed with a B/W printer.

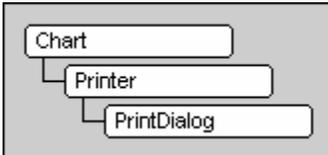
Printing the Chart

Another major consideration is the number of pages that are needed to print a chart. When you have a chart with a large number of points (Scrollable) and you print that chart, Chart FX will print as many pages as necessary. The **Compress** property forces Chart FX to print the chart in just one page by recalculating the appropriate values so that all the points can be fixed in one page. Please note the printed chart may have a different look if you use the Compress property before printing. Below is the API used to compress and print a chart:



When the Print method is called with no parameters, Chart FX will print the entire chart. To print selected pages of a chart, the **nFrom** and **nTo** parameters can be passed to the chart when the method is called. This will print only the specified range of pages for the chart. If the nTo parameter is configured to zero, Chart FX will print from the specified nFrom page to the end of the chart. And if both the nFrom and nTo parameters are set to -1, only the screen shown is printed (used to print a zoomed area of a chart).

If you would like to prompt the user with a print dialog, you may do so by calling the **PrintDialog** method. The print dialog allows the user to select the printer to be used, specify the print range and configure the number of copies to be printed. This property also uses a Boolean parameter called **fromTo**. When set to "True" users can modify the print range. When set to "False", the print range options are not available to the user. The API below will prompt the user with a print dialog:



Appendix C - Passing Data Using Databases



Chart FX for .NET

In most cases, you will want to populate charts with data coming from a database table or query. Chart FX for .NET supports data binding through ADO.NET.

ADO.NET offers you two basic ways to work with data: using a DataSet or working directly against a database. In the DataSet model, you create an in-memory store of the records you want to work with, load the store using a data adapter, manipulate the data, and then, optionally, use the data adapter to write changes back to the database.

Alternatively, you can work directly against the database. In this model, you configure a data command object with a SQL statement or the name of a stored procedure. You then execute the data command.

Development strategies used for database access vary greatly from application to application. Furthermore, ADO.NET can be used to integrate data from complex application tiers. Therefore, this section only describes basic techniques on how Chart FX for .NET uses ADO.NET connections to populate charts in Web Forms or Windows Forms applications.

Dataset

Creating a DataSet in Visual Studio .NET

Before you bind the chart to a DataSet in Visual Studio .NET, you must first create a DataSet that contains a copy of the database records. In this section you will add a DataSet to a form that connects to an Access database to fetch records from a single table.

Essentially, creating a DataSet at design time in Visual Studio can be accomplished by using the Data Adapter Wizard. The adapter contains SQL statements used to read database information. If necessary, the wizard also creates a database connection. Then a data schema must be generated.

To create the data connection and data adapter

1. From the Data Tab of the Visual Studio Toolbox, drag an **OleDbDataAdapter** object onto the form. **The Data Adapter Configuration Wizard** will start.
2. In the wizard do the following:
 1. In the second pane, create or choose a connection to the desired database. In our case we'll point to an Access database called Test.mdb.
 2. In the third pane, specify that you want to use a SQL statement to access the database.
 3. In the fourth pane, type the desired SQL statement. In our case we will write the following statement that contains monthly sales information for 2 different products in a table. Please note, you can simply type the SQL statement or get assistance from the wizard **Query Builder**.

```
SELECT Month,Product1,Product2 from Sales
```
3. Click Finish.

To create the DataSet

Visual Studio can generate the DataSet automatically based on the query you specified for the data adapter. The DataSet is an instance of the DataSet class based on a corresponding schema (.xsd file) that describes the class' elements (table, columns and constraints). To generate a DataSet do the following:

1. Click on the Form, from the **Data** menu, choose **Generate DataSet**.
2. In the Generate DataSet dialog, select the **New** option and name the dataset. In our case we will name the DataSet **dsSales**.
3. Check **Add this DataSet to the designer**, then click **OK**.

Visual Studio generates a typed DataSet class (dsSales) and a schema that defines the DataSet. Finally, Visual Studio adds an instance of the new DataSet class (dsSales1) to the form.

Configuring the chart control with the new DataSet class

1. Drag a chart control onto the form, select the chart.
2. In the chart **DataSource** property, select the new DataSet class and its member (in our case **dsSales1.sales**).

Setting this property binds the sales table in the dsSales1 DataSet to the chart.

Filling the DataSet and displaying data in the chart control

Although the chart is bound to the DataSet you've created, the DataSet itself is not automatically filled. Instead you must fill the DataSet yourself by calling a data adapter method. After the DataSet is filled, the chart control automatically displays the data.

1. In the Page_Load event handler call the data adapter's Fill method, passing it the DataSet you want to populate, as follows:

```
OleDbDataAdapter1.Fill(dsSales1)
```

If you are building a Web Forms application, you do not need to refill the DataSet and bind the chart with each round trip. Once the chart control has been populated with data, its values are preserved in view state each time the page is posted. Therefore, you only need to fill the dataset and bind the chart the first time the page is called. You can test this using the page's **IsPostBack** property as follows:

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    If Not IsPostBack Then
        OleDbDataAdapter1.Fill(DsSales1)
    End If
End Sub
```

Creating and configuring a DataSet by code in an event handler

Sometimes, the steps for creating a Data Adapter and filling a DataSet need to be executed as a result of a condition or as part of an event handler. For example, when the user clicks a button the data adapter is created and the DataSet is filled.

Sometimes, adding the data adapter and configuring it at design time is not practical or does not provide enough programming flexibility. The following VB.NET code accomplishes the same tasks as described in previous pages for creating and configuring a dataset within Visual Studio .NET, except that the code can be assigned to an event handler or used outside the designer environment:

```

Dim myConnectionString, mySelectQuery As String
Dim dsSales As New DataSet()
Dim adapter As System.Data.OleDb.OleDbDataAdapter
Dim myConnection As System.Data.OleDb.OleDbConnection

'Create the connection string
myConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=Test.mdb"

' Fill the query statement
mySelectQuery = "SELECT Month,Product1,Product2 from Sales"
myConnection = New System.Data.OleDb.OleDbConnection(myConnectionString)

'Create the DataSet and Data Adapter
dsSales = New DataSet("Sales")
adapter = New System.Data.OleDb.OleDbDataAdapter()
adapter.SelectCommand = New System.Data.OleDb.OleDbCommand(mySelectQuery, myConnection)

' Fill the dataset
adapter.Fill(dsSales)

'Bind the chart to the dataset
Chart1.DataSourceSettings.DataSource = dsSales.Tables(0)

```

Creating and configuring a DataSet by code in a single-file web application

If you are developing Web Forms applications with Visual Studio, the environment will automatically perform much of the configuration process. However, if you want to connect to a database in a single-file .aspx application, you must remember to import the appropriate namespaces in the .aspx file.

The following C# code embedded in a single-file web application will configure a data adapter and bind the chart to a desired database table:

```

<%@ Import Namespace="SoftwareFX.ChartFX"%>
<%@ Import Namespace="SoftwareFX.ChartFX.Internet.Server"%>
<%@ Import Namespace="System.Drawing"%>
<%@ Import Namespace="System.Data"%>
<%@ Import Namespace="System.Data.OleDb"%>
<%
Chart Chart1 = new Chart(this);

// Create the database connection object
OleDbConnection myConnection = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0; Data
Source=Test.mdb");

// Create the command and DataSet
OleDbDataAdapter myCommand = new OleDbDataAdapter("SELECT Month,Product1,Product2 From
Sales",myConnection);
DataSet dsSales = new DataSet();

// Fill the data set
myCommand.Fill(dsSales, "Sales");

// Assign the contents of the ResultSet to the Chart
Chart1.DataSourceSettings.DataSource=dsSales.Tables[0];
%>
<%= Chart1.GetHtmlTag(500,350)%>

```

Data Command

By using data commands, you can execute SELECT statements that return a result set you can read directly, rather than loading it into the DataSet. To read the results, you use a data reader (OleDbDataReader or SqlDataReader object), which works like a read-only, forward-only cursor, to which you can bind charts. This is a useful strategy for reducing memory usage and loading read-only data very quickly.

The following VB.NET code creates a Data Command and uses a data reader to fetch records directly from the database (without filling a DataSet) to populate the charts:

```
Dim mySelectQuery As String = "SELECT Month,Product1,Product2 FROM Sales"
Dim myConnString As String = "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=test.mdb"
Dim myConnection As New OleDbConnection(myConnString)
Dim myCommand As New OleDbCommand(mySelectQuery, myConnection)
myConnection.Open()
Dim myReader As OleDbDataReader
myReader = myCommand.ExecuteReader()
Chart1.DataSourceSettings.DataSource = myReader
myReader.Close()

' Close the connection when done with it.
myConnection.Close()
```

Data Type and Data Style Property

Data Type property

The **Data Type** property is an array property that indicates the type of every field and how it should be extracted from the SQL query and used in the chart. For example, in a 5 field SELECT statement such as:

```
SELECT year, sales, projected, returns, name FROM Sales
```

The Chart FX default behavior is that the year field will be considered another series since it is a numeric field. Instead, we want this field to be used as a label in the X Axis tickmarks. Also, since this SQL query is also being used to fill a grid control (which contains more information than the chart) we do not want to get rid of the **returns** and **name** fields in the query but we also don't want them to be used anywhere in the chart. However, the Chart FX default behavior would cause these fields to be appended as X Axis labels.

We can easily override the Chart FX default behavior by filling the Data Type property as follows:

```
' We have to instruct Chart FX to consider the year field as a label for the x-axis.
Chart1.DataSourceSettings.DataType(0) = DataType.Label

'Then assigned the Value enumeration to the numeric fields
Chart1.DataSourceSettings.DataType(1) = DataType.Value
Chart1.DataSourceSettings.DataType(2) = DataType.Value

'Finally, assign the NotUsed enumeration to those fields we don't want to plot.
Chart1.DataSourceSettings.DataType(3) = DataType.NotUsed
Chart1.DataSourceSettings.DataType(4) = DataType.NotUsed
```

DataStyle property

When having different string or date fields Chart FX will construct a long string with every string and date field to assign to every legend point in the chart. If you want to avoid this behavior, just turn OFF the appropriate constants using the DataStyle property. For example, if you want to force Chart FX not to use field names as series legends and use date fields as point legends, you should use the DataStyle property as follows:

```
Chart1.DataSourceSettings.DataStyle = Chart1.DataSourceSettings.DataStyle Or DataStyle.UseDateAsLeg  
And Not DataStyle.SerLegend
```

Note: The DataStyle property applies to all data-binding methods covered in this chapter: Text File, XML, Collections, etc.

Chart FX for Java

In most cases, you will want to populate charts with data coming from a database table or query. Chart FX for Java supports data binding through JDBC.

JDBC offers you a basic ways to work with data: using a `ResultSet`, where you create an in-memory store of the records you want to work with.

It is important to note, development strategies used for database access vary greatly from application to application. Furthermore, JDBC can be used to integrate data from complex application tiers. Therefore, this section is limited to describe basic techniques on how Chart FX for Java uses JDBC and its `JDBCDataProvider` to populate charts in JSP applications.

JDBC ResultSet

When you execute `SELECT` statements against a JDBC connection, you get in return a `ResultSet` that contains all the records selected. To read the results, we provide a `JDBCDataProvider`, which works like a read-only, forward-only cursor, to which you can bind charts. This is a useful strategy for reducing memory usage and loading read-only data very quickly.

The following Java code creates a `ResultSet` and uses a `JDBCDataProvider` to populate the charts:

```
String url="jdbc:jtds:sqlserver://dbserver:1433/NetSamples";
java.sql.ResultSet rs = null;
java.sql.Connection conn = null;
Class.forName("net.sourceforge.jtds.jdbc.Driver");
conn = java.sql.DriverManager.getConnection(url,"webguest","");
String query = "SELECT * from SampleFinacial1";
java.sql.Statement stmt = conn.createStatement(java.sql.ResultSet.TYPE_SCROLL_INSENSITIVE,
java.sql.ResultSet.CONCUR_READ_ONLY);
rs = stmt.executeQuery(query);
JDBCDataProvider provider = new JDBCDataProvider(rs);
chart1.setDataSource(provider);
conn.close();
```

Note: The above example uses the jTDS JDBC driver from The jTDS Project as an example. Any JDBC driver may be used to load data into the `JDBCDataProvider` and Chart FX for Java.

DataType and DataStyle Property

DataType Property

The **DataType** property is an array property that indicates the type of every field and how it should be extracted from the SQL query and used in the chart. For example, in a 5 field SELECT statement such as:

```
SELECT year,sales,projected,returns,name FROM Sales
```

The Chart FX default behavior is the year field will be considered another series since it is a numeric field. Instead, we want this field to be used as a label in the X-axis tickmarks. Also, since this SQL query is also being used to fill a grid control (which contains more information than the chart) we do not want to get rid of the **returns** and **name** fields in the query but we also don't want them to be used anywhere in the chart. However, the Chart FX default behavior would cause these fields to be appended as X-axis labels.

We can easily override the Chart FX default behavior by setting DataType property as follows, using the **setDataType** method:

```
// We have to instruct Chart FX to consider
//     the year field as a label for the X-axis
chart1.setDataType(0,DataType.LABEL);
// Then assigned the Value enumeration
//     to the numeric fields
chart1.setDataType(1,DataType.VALUE);
chart1.setDataType(2,DataType.VALUE);
// Finally, assign the NotUsed enumeration to
//     those fields we don't want to plot
chart1.setDataType(3,DataType.NOTUSED);
chart1.setDataType(4,DataType.NOTUSED);
```

DataStyle Property

The **DataStyle** property is used to control how Chart FX takes the information from a data source. In other words, this property specifies what field types should be considered when data is passed to the chart. This property may be used for, transposing data, reading values and X-values, setting legends, etc.

For example, if the following SQL statement:

```
SELECT Month,Product1,Product2 FROM Sales
```

returns 5 records, the chart will contain 2 lines (assuming it is a line chart) and each line will have 5 point markers. If you want to create a chart with 5 lines (each with two point markers) you would set the Transpose datastyle before the **setDataSource** call:

```
chart1.setDataStyle(chart1.getDataStyle() | DataStyle.TRANSPOSE);
```

Another useful DataStyle flag is **ReadXValues**, which instructs Chart FX that the data read will include both values and x values.

Let's say that your data looks like this, and you want to plot an XY chart:

```
1 200 3 400 2 100 6 300
```

By just calling the following API:

```
chart1.setDataStyle(chart1.getDataStyle() | DataStyle.READXVALUES);
```

Chart FX will read the data as if you have done:

```
chart1.setXValue(0,0,1);  
chart1.setValue(0,0,200);  
chart1.setXValue(1,0,3);  
chart1.setValue(1,0,400);  
chart1.setXValue(0,1,2);  
chart1.setValue(0,1,100);  
chart1.setXValue(1,1,6);  
chart1.setValue(1,1,300);
```

3

3D Bar Charts. *See* Gallery

A

Access. *See* Development Environments

Accessibility, 152

Section 508, 152

AddCommand, 166

AddPoints, 158

Advanced Features, 155

Real Time Charts, 157

AllowDrag, 168

AlternateColor, 103

Annotation, 170

Assembly Integration, 170

Attach, 172

Group, 173

Height, 172

List, 171

Object Creation, 170

Object Instances, 171

Orientation, 172

Positioning, 172

Recalculating Bounds, 173

ToolBar, 174

Width, 172

Annotation Extension, 170

Area Charts. *See* Gallery

Attach, 172

AutoScale, 96

Axes Titles, 89

Axis, 91

Additional Axes, 105

Axis Object, 95

Categorical, 94

Formatting, 97

Gridlines, 102

Interlaced Grids, 103

Introduction, 93

Labeling, 98, 99

Numerical, 94

Panes, 107

Positioning, 101

Scaling, 96

Scrolling, 104

Secondary Y Axis, 106

Sections, 109

Tickmarks, 102

B

Background Color, 79

Bar Charts. *See* Gallery

Bitstream, 175

Borders, 179, *See* Visual Attributes

Background Color, 79

Other Tools, 80

Bubble Charts. *See* Gallery

BufferSize, 157

C

C# Builder. *See* Development Environments

Caching, 182

Calculating Axis Scale. *See* Axis

Capturing Mouse Events, 168

Categorical Axis. *See* Axis

CF. *See* Development Environments

Chart Render Size, 180

Color Palettes. *See* Visual Attributes

Colors, 212

Combination Charts. *See* Gallery

CommandID, 162, 167

Commands

Custom, 166

List, 166

Compact Framework. *See* Development Environments

Compress, 213

Conditional Attributes, 118

Constant Lines, 114

Context Menus, 125

Customizing, 167

Contour Plots. *See* Gallery

Crosstab. *See* Passing Data

Custom Command

Processing, 167

Visual Attributes, 167

Custom Commands, 166

Custom Labeling, 98

Customizing DataTips, 168

D

Data Editor, 113

Databases, 215

.NET, 217

Java, 222

DataEditorObj, 113

DataTips, 168

DefaultBorderColor, 174

DefaultFillColor, 174

Design Time

Java Designer, 57

- Properties List, 56
- Wizard, 51
- Development Environments, 13
 - .NET Compact Framework, 22
 - Access, 19
 - C# Builder, 16
 - Delphi 6 and 7, 17
 - Visual Basic 6.0, 18
 - Visual Studio .NET, 15
- DockArea, 89
- Drill Down
 - API, 142

E

- Enabled, 137, 174
- End User Experience, 121
 - Axis Settings, 127
 - Context Menus, 125
 - Customizing Toolbar, 160
 - Introduction, 123
 - Menu, 124
 - Point Attributes, 126
 - Rotation & Perspective, 128
 - Series Attributes, 126
 - Toolbar, 124
 - Tooltips, 130
- Extension
 - Annotation, 170
- Extensions, 185
 - Financial, 198
 - Introduction, 187
 - Maps, 188
 - OLAP, 192
 - Real-Time, 201
 - Statistical, 194
 - Wireless, 203

F

- Factor, 86
- Features
 - Performance, 179
- Financial Charts. *See* Gallery
- FirstLabel, 99
- Fonts. *See* Visual Attributes
- ForceColors, 212
- Format, 97
 - Chart, 180
- Formatting Axis. *See* Axis

G

- Gallery
 - Polar Charts, 41
 - Radar Charts, 41
- Gallery, 28
- 3D Bar charts, 31

- Area Charts, 33
- Bar Charts, 29
- Bubble Charts, 40
- Combination Charts, 45
- Contour Plots, 42
- Financial Charts, 39
- Line Charts, 34
- Maps, 46, 188
- Pareto, 44
- Pie Charts, 36
- Scatter Plots, 38
- Stacked Charts, 30
- Statistical Charts, 47
- Surface, 42
- Generation
 - Bitstream, 175
 - Image Maps, 142, 149
 - SVG, 151
- GetHtmlData, 181
- GetHtmlTag, 148, 181
- GetTip, 168
- Globalization, 205
- Gradients, 179. *See* Visual Attributes
 - Bi-color, 85
 - in chart background, 84
 - in chart markers, 84
- Gridlines, 102. *See* Axis:Gridlines

H

- Help, 5
 - Dynamic Help, 8
 - IntelliSense, 8
 - Javadocs, 10
 - Newsgroups, 11
 - Resource Center, 7
 - Support Web Site, 11
- Highlighting, 116
- HitTest, 169
- HtmlTag, 147

I

- Image Map
 - Parameters, 149
- Image Maps, 142, 149
- ImageBorders, 79
- ImgMap, 131, 150
- InsertAt, 162
- InsertPoints, 158
- InsideColor**, 103
- Integrating
 - Annotation Assembly, 170
- Interlaced Grids. *See* Axis:Interlaced Grids
- Inverting Axis. *See* Axis:Positioning

J

Java Designer
 Installation, 57
 Licensing, 58
 Running, 57
 Using, 58
JPEG, 148, 180

L

Label, 99
Label Axis. *See* Axis
Labeling Axis. *See* Axis
Line Charts. *See* Gallery
LineAlignment, 89
Load My Chart, 139
Logarithmic Scales, 97
LogBase, 97

M

Maps. *See* Extensions, *See* Gallery
Margins, 211
Marker Dragging, 168
Maximum, 96
Menubar, 124
MenuBarObj, 160
Menus on Demand, 168
Minimum, 96
MinorStep, 96
Mouse, 168
Mouse Events, 168, 169
MouseEventArgsX, 169
MouseMove, 169
MouseTip
 ImgMap, 131, 150

N

Numerical Axis. *See* Axis

O

Orientation, 212
Output, 145
 Image Format. *See*
 Writers, 148
OutputWriter, 152

P

PageSetup, 211
Panels. *See* Axis:Panels
Parameters
 Image Map, 149
Pareto, 44, *See* Gallery
Passing Data
 Crosstab, 71

Databases, 68
Getting Started, 61
Real-time, 157
Series and Points, 62
XML, 69
Per Point Attributes, 119
Performance, 179
 Chart Size, 180
Performance and Scalability, 177
Pie Charts. *See* Gallery
PNG, 147, 180
Polar Charts. *See* Gallery
PrintDialog, 213
Printer, 211
 , 209
 Colors, 212
 Margins, 211
 Orientation, 212
 Page Layout, 211
PrintPreview, 211
PSS, 182

R

Radar Charts. *See* Gallery
Real Time Charts. *See* Advanced Features
Real-time
 AddPoints, 158
 InsertPoints, 158
 Limited Real-time, 157
 Passing Data, 157
 Scrolling Labels, 159
 Style, 159
 Unlimited Real-time, 157
RecalcBounds, 173
RecalcScale, 96
RemoveAllSubCommands, 165
RemoveAt, 161
RemoveSubCommand, 165
Rotation and Perspective. *See* End User
 Experience

S

Save My Chart, 139
Scalability, 179
Scaling
 Axis. *See* Axis
Scaling Axis. *See* Axis
Scatter Plots. *See* Gallery
Scrolling. *See* Axis
Secondary Y Axis. *See* Axis
Section 508, 152
Sections. *See* Axis:Sections
Semi-Transparency, 81
Server Clustering, 181
Settings
 Performance, 179
 SmoothFlags, 179
Stacked Charts. *See* Gallery

Statistical. *See* Extensions
Statistical Charts. *See* Gallery
Stripes, 115
Style
 Commands, 166
 Real-time, 157
SubCommandID, 165
Subcommands, 165
 Adding, 165
 Removing, 165
Surface. *See* Gallery
SVG, 148, 151, 180

T

Tilting Axis. *See* Axis:Positioning
TipMask, 131, 150, 168
Title, 88
TitleDockable, 88, 89
Titles. *See* Visual Attributes
 Axes, 89
TitleS, 89
TitleTip
 ImgMap, 131, 150
Tool Tips, 150
Toolbar, 124
ToolBar
 Adding Custom Commands, 166
 Adding Subcommands, 165
 Adding/Replacing Icons, 164
 Annotation, 174
 Changing Existing Functions, 162
 Changing Icons, 164
 Customizing, 161
 Inserting Items, 162
 Modifying ToolTips, 163
 Positioning, 161
 Pre-Defined Icon, 164
 Relation between ToolBar and Commands, 160
 Removing Items, 161
 Removing Subcommands, 165
 Selectors, 166
 Showing/Hiding, 161
 Subcommands, 165

ToolBarObj, 160
Tools
 Customizing Context Menus, 167
 Data Editor, 113
ToolTips, 163
Tracking the Mouse, 169
Transparency, 179, *See* Visual Attributes
TypeEx
 Real-time, 159
TypeMask
 Real-time, 157

U

UserCommand, 167
UserScroll, 104, 136

V

Visual Attributes, 75
 Borders, 79
 Color Palettes, 82
 Fonts and Titles, 87
 General, 77
 Gradients, 84
 Per-Series, 78
 Per-Series Attributes. *See* Visual Attributes
 Transparency, 81
Visual Basic 6.0. *See* Development Environments

W

Web Farms, 181
Wireless. *See* Extensions
Wizard, 15, 51, 52, 53, 54, 55, 80, 85, 217
 Design Time Data, 55
 Disabling, 54
 Re-entry, 54

X

XY Plots. *See* Scatter Plots