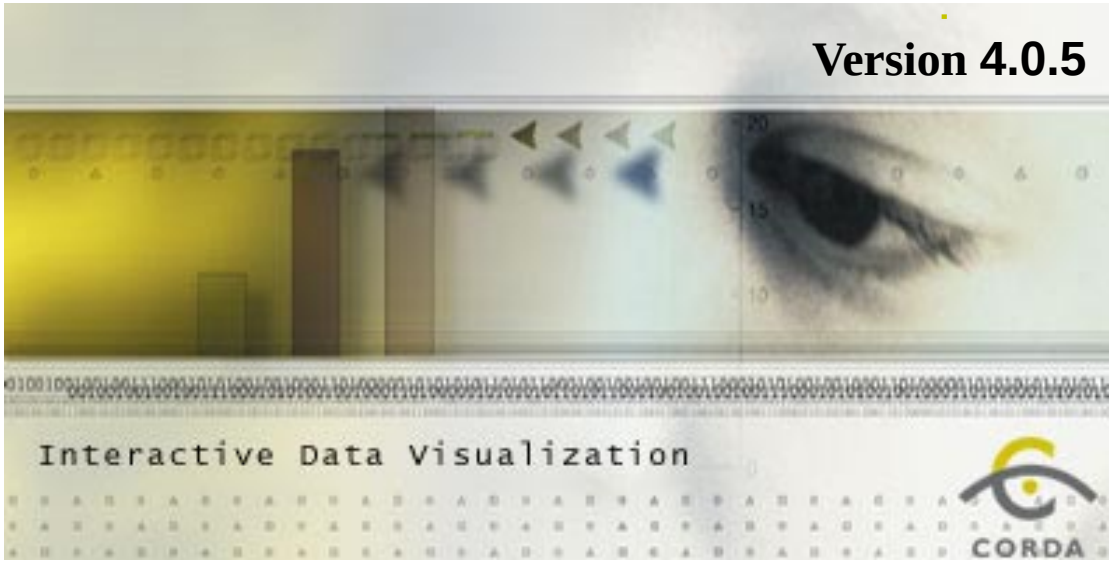
PopChart Server Reference



Last Updated on August 6, 2002

Corda Technologies, Inc.

350 South 400 West, Suite 100

Lindon, UT 84042

Headquarters: (801) 805-9400

Fax: (801) 805-9405

Sales: (801) 805-9500

Technical Support: (801) 805-9505

Press Contact: (801) 805-9431

Sales Email: sales@corda.com

Support Email: support@corda.com

Contents

CHAPTER 14 Image Formats

1 INTRODUCTION

Welcome to the *PopChart Server Reference* manual for PopChart Server, version 4.0.5. In this manual you will find complete command lists, APIs, and document specifications for PopChart Server.

This book is intended for reference purposes only. It is designed for an advanced user who needs to look up a function or command quickly. If you are new to PopChart Server, you should refer to either the *PopChart Quick Start* manual or the *PopChart Server User Guide*.

This documentation will be updated frequently as performance is enhanced and new features are added. Please visit the Corda Technologies website (http://www.corda.com) regularly for the latest documents.

WHAT IS POPCHART?

In a world of ever increasing information, perhaps no skill is more valuable than the ability to convey that information in the most understandable and accessible format possible. Raw data is no exception to this rule, yet because understanding raw data often involves wading through spreadsheet after spreadsheet in search of key figures and trends, it is often the hardest type of information to convey.

For that reason, data visualization is paramount. Graphs and charts can convey in a few seconds information that is often not clear even after hours of analyzing numbers. Data visualization is what Corda Technologies is all about. With our easy-to-use PopChart tools, you can translate your data into state-of-the-art PopChart images—eye-catching, high-resolution, and interactive data-driven graphics, such as the ones on the next page.

A PopChart image can contain a variety of charts and graphs, fed with on-demand dynamic data. It can include explanatory text boxes, callout notes, or PopUp text that appears as a viewer rolls over certain parts of the graph. It can even include interactive drill-down effects, such as linking to another PopChart image as a user clicks on a certain data item, or executing your own custom JavaScript™ functions.

As you read through this documentation, you will learn all about how you can utilize PopChart technology to convey your information. And, if you haven't already, you will soon discover why PopChart is your all-purpose data visualization tool.

ABOUT PopChart Server

PopChart Server does exactly what its name implies—that is, it serves images of charts and graphs. But these aren't just run-of-the-mill static charts and graphs. These are dynamic and interactive images generated by PopChart Server on the fly.

It works like this. You create an appearance file (kind of like a template for a graph) with PopChart Builder. Then, you send this appearance file to PopChart Server, along with data and an appearance file, and PopChart Server returns a PopChart image—a graphical representation of your data, complete with PopUp text and the ability to drill-down to another graph that explains a data item in greater detail.

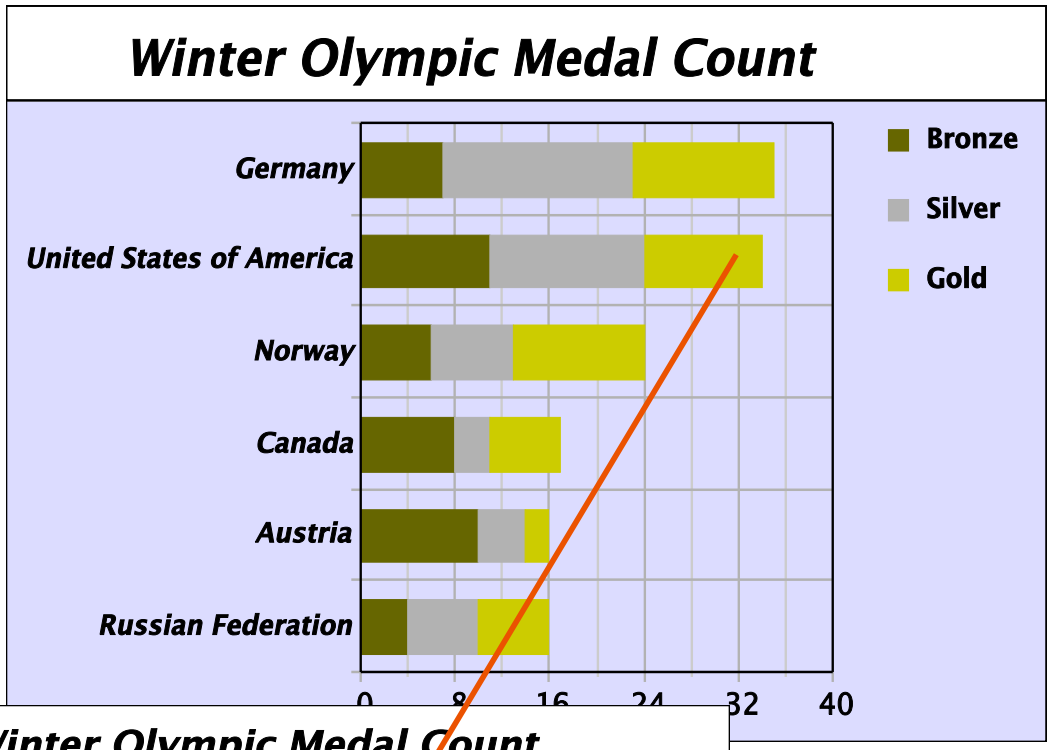
The image can be in one of many different types of formats, including FLASH, SVG, PNG, GIF, PDF, EPS, WBMP, and even d link descriptive text for the visually impaired. You can also interface natively with PopChart Server in a variety of environments, from simple HTML to ColdFusion®; from Java™ Application Servers to Microsoft®'s.NET framework. PopChart Server can accept data from most database and data file formats. It even supports XML, making PopChart Server easy to integrate with your existing database system.

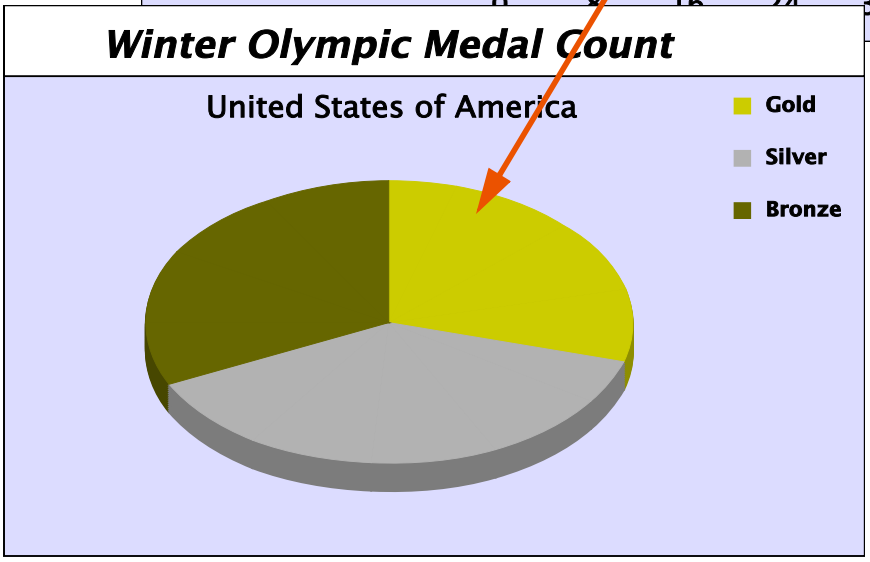
PopChart Server is the fastest, most robust, and most versatile data visualization and charting tool on the market today. Best of all, because PopChart Server is written in 100% Java, it can run on any platform. No matter what environment you operate in, you can take

advantage of PopChart Server's patented DataFunnel™ technology to deploy the latest in state-of-the-art interactive data-driven graphics.

Over 20 different graph types, including Bars, Pies, Gauges, X-Y, Time, and Radar.

Data can be dynamic or static. PopChart also supports XML!





Data items drill-down to other web pages or PopCharts as the user clicks on them. Or you can execute a JavaScript function.

Crisp, colorful, 3D graphics in 7 different formats, including FLASH, GIF, SVG, and PDF.

OTHER POPCHART TOOLS

In addition to PopChart Server, you may also be interested in the following PopChart tools:

POPCHART BUILDER

PopChart Server works hand-in-hand with PopChart Builder, a graphical design tool that helps you design appearance files (templates) for your PopChart images. This easy-to-use development tool can be purchased separately from PopChart Server, as many companies have multiple developers creating appearance files, but only one or two servers.

For your convenience, the download for PopChart Server includes PopChart Builder, but unless you have a separate license key, you will only be able to run the evaluation version of it. If you chose to install documentation when you install PopChart Server, the *PopChart Builder User Guide* is also included for your reference.

POPCHART XPRESS

For those looking to publish static PopChart images from their desktop, there's PopChart Xpress, a program that can run on any operating system and is easy enough is easy enough for even non-technical employess to use. You simply choose a graph type in the PopChart Wizard, copy data from a spreadsheet program such as Microsoft Excel, select a few formatting options, and PopChart Xpress generates everything you need to publish an image of your graph on the web.

PopChart Xpress and PopChart Builder can be downloaded for evaluation or purchased from the Corda Technologies website at http://www.corda.com.

CONVENTIONS USED IN THIS DOCUMENTATION

To make this manual easier to read, we use some special conventions when referring to files, URLs, example code or text, and options or buttons in dialogs and menus. For the most part, these should be pretty intuitive, but we mention them here just in case.

FILENAMES

Names of files are shown in a slightly larger arial font and are colored gray (e.g. myfile.txt).

Unless stated otherwise, files are all relative to the *PopChart root directory*. This directory will vary from system to system. It is the folder where you installed PopChart Server. Usually, this is C:\Program Files\Corda40 on Microsoft Windows® systems, or /usr/local/Corda40 on UNIX® systems.

Another commonly used term is document root, which is where you keep all of your data, xml, image, and appearance files for PopChart Server. Unless you have changed it in the Administration Console, your document root will be the chart_root folder in your PopChart root directory.

URLS

URLs are always shown in a slightly larger arial font. They are shown in two colors. If the URL actually exists, it will be blue (e.g. http://www.corda.com). You will also be able to click on it as if it were a normal link. If the URL is for example purposes only, it will be colored gray (e.g. http://www.yourserver.com).

If the example URL is in reference to PopChart Server, you can usually make it a valid URL if you replace www.yourserver.com with the address (and port) of your PopChart Server.

EXAMPLE CODE

Small segments of code (including HTML and XML) that appear in the body of a normal paragraph are shown in courier font, and are colored light green or yellow (e.g. `graph.Transposed(true)`).

Medium sized segments of code appear on a single indented line in a courier font, as shown below:

```
<img src="http://www.yourserver.com:2001/?@_LOG>
```

Long segments of code look similar, but are on multiple lines and are delimited with a thin horizontal line above and below the code. At the top left corner of the segment, there will be an example number for reference. The code segment will also be given a title. Example 1.1 below demonstrates a longer segment of code.

Example 1.1 Example of Long Code Segment

```
myPopChart = new PopChartEmbedder();
myPopChart.appearanceFile("apfiles/test.bin");
myPopChart.pcScript("graph.setCategories(Jan;Feb;Mar;Apr;May;Jun)gr
                aph.setSeries(Temperature;30;40;50;60;70;80)graph.a
                ddPopupText(1,1,Look at the pretty pictures)");
myPopChart.getEmbeddingHTML();
```

EXAMPLE TEXT

Any other example text, including text that should be typed into a dialog window and values of attributes or variables, will appear italicized in a slightly larger arial font (e.g. *false* in "The default value of `graph.Transposed` is *false*").

MENU/DIALOG NAMES AND OPTIONS

Names of menus, dialogs, and options are in a slightly larger arial font and are colored light green or yellow (e.g. Project Properties).

For brevity, whenever you have to go through multiple layers of menus or dialogs to arrive at the desired menu, dialog, or option, a greater than > sign will be used to indicate that the next menu, dialog, or option can be found under the previous.

For example, instead of saying, "Select the Save option from the File pull-down menu," this documentation will say, "Select File > Save." Similarly, instead of saying, "Change the value of the Port option in Address/Port screen of the Settings section in the Administration Console," this documentation will say, "Change the Settings > Address/Port > Port option in the Administration Console."

XML NOTATIONS AND CONVENTIONS

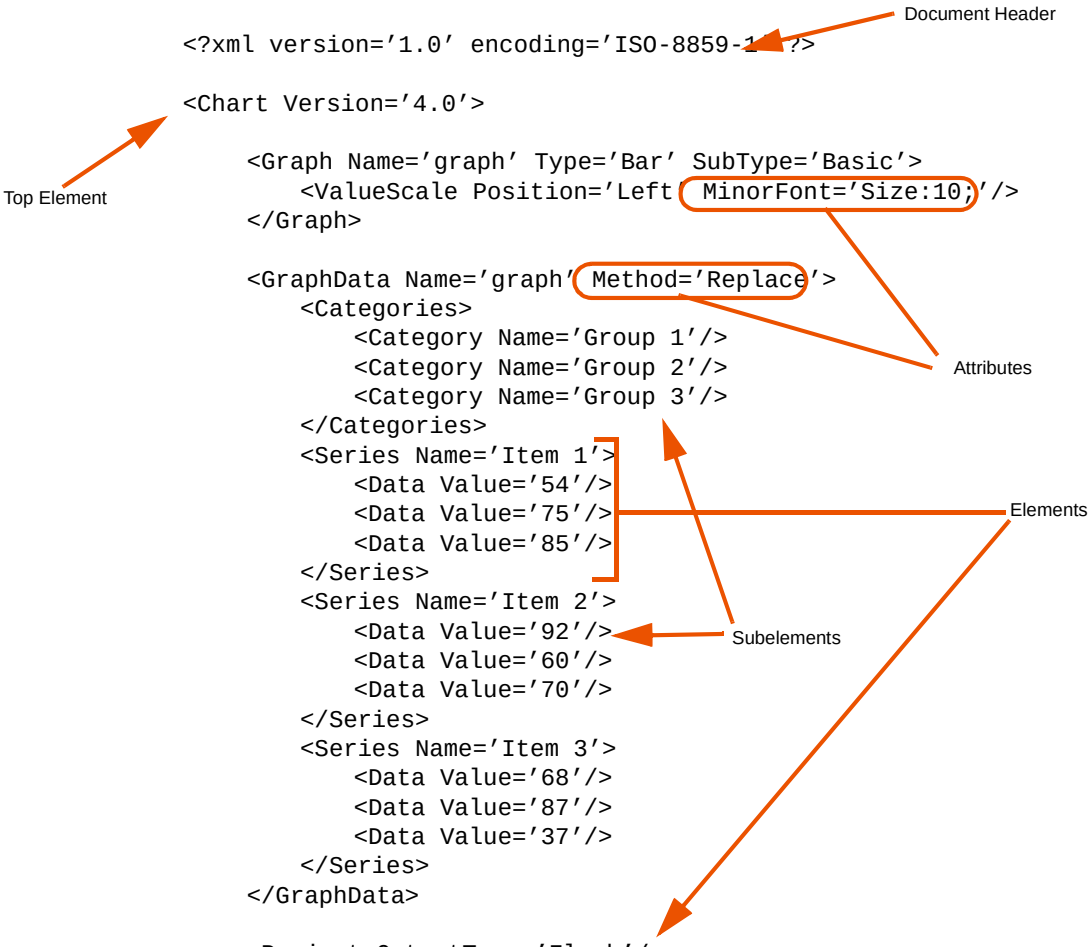
XML (eXtensible Markup Language) is meta-markup language that can be used to describe anything from data in a database, to word-processing documents, to (as you will soon see) graphs and charts. Its versatility, uniformity, and ease of use has led to its rapid acceptance as the standard for information sharing.

PopChart Server 4.0.5 provides extensive support for XML Data and appearance files. Additionally, almost all of the configuration files for PopChart Server are in XML. Thus, if you've been avoiding XML until now, PopChart Server may give you the perfect excuse to learn it.

There are numerous resources on the Internet that will teach you how to use XML. For that reason, this documentation does not endeavour to teach you everything about XML. Instead, we only offer a quick example to clarify the conventions and notations used in this documentation.

Example 1.2 shows a typical XML document.

Example 1.2 **XML Document**




```
<?xml version='1.0' encoding='ISO-8859-1'?>

<Chart Version='4.0'>

    <Graph Name='graph' Type='Bar' SubType='Basic'>
        <ValueScale Position='Left' MinorFont='Size:10;'/>
    </Graph>

    <GraphData Name='graph' Method='Replace'>
        <Categories>
            <Category Name='Group 1'/>
            <Category Name='Group 2'/>
            <Category Name='Group 3'/>
        </Categories>
        <Series Name='Item 1'>
            <Data Value='54'/>
            <Data Value='75'/>
            <Data Value='85'/>
        </Series>
        <Series Name='Item 2'>
            <Data Value='92'/>
            <Data Value='60'/>
            <Data Value='70'/>
        </Series>
        <Series Name='Item 3'>
            <Data Value='68'/>
            <Data Value='87'/>
            <Data Value='37'/>
        </Series>
    </GraphData>

    <Project OutputType='Flash'/>

</Chart>
```

Here is a brief explanation of the different parts of this XML document.

DOCUMENT HEADER This line specifies the document type (*XML*), version (*1.0*) and encoding (*ISO-8859-1*). It is good practice to include this line in your XML documents, but PopChart Server does not require it.

ELEMENTS An XML element is essentially a starting tag (e.g. `<GraphData>`), followed by data, followed by a closing tag (e.g. `</GraphData>`). Elements that do not have subelements will only have one tag with a slash at the end (e.g. `<Data />`).The starting tag can contain any number of attributes inside of it. The data can consist of any number of subelements.

TOP ELEMENT We refer to the first element in the document as the top-level element. All other elements in the document are subelements of the top element. In PopChart XML (which is what this document is), the top element is `<Chart>`.

ATTRIBUTE An element can have any number of attributes, all of which should be listed in the starting tag for that element. An attribute consists of the attribute name, an equals = sign, and an attribute value enclosed in quotation marks. For example, `Name` and `Type` are attributes of the `Graph` element.

SUBELEMENT A subelement is any element that is enclosed in another element. For example, `<Data>` is a subelement of `<Series>`.

2

ADMINISTRATION CONSOLE

This chapter provides a screen-by-screen reference to the PopChart Administration Console, as well as information on changing the port that the Administration Console runs on. The latter information can be found in the last section of this chapter, "Changing Server and Administration Port Settings."

The Administration Console is divided into the following sections. You can jump to a description of the screens in any of these sections by clicking on the appropriate section name.

Note: *You can view this help document any time you want to in the Administration Console by clicking on the help icon at the top right corner of the page.*

LOGIN & PASSWORD

This is the screen that you will see as you first enter the Administration Console. It allows you to specify the location of and password for your PopChart Server.

PASSWORD

This is your PopChart Server password. When you first install PopChart Server, this password is *password*. You should change this password immediately after you first login. You can change your password from the Security > Change Password page.

Warning: Your password can use alphanumerical characters and spaces only. Using other characters for your password will result in an error.

If you have forgotten your password, refer to "Unknown or Forgotten Administration Console Password" on page 14-9 of the *PopChart Server User Guide*.

SERVER

The screens in this section display configuration, usage, statistical, and version information about PopChart Server. You cannot change any information in these screens, however, you can stop, start, and restart PopChart Server.

These screens include:

- Home
- Performance Charts
- Statistics
- Version / License
- Stop All

HOME

This screen, which you will see immediately after you login to the Administration Console, contains configuration information for your PopChart Server. Below this information, there are buttons that allow you to stop and restart PopChart Server.

The following paragraphs describe the information that you will see on this page.

SERVER VERSION

The line will displays the server edition (i.e. PopChart Server, PopChart Server Pro, or PopChart Server Enterprise) and version (e.g 4.0.5).

SERVER ADDRESS

The address and port on which PopChart Server is currently running. You can change the address and port via the Settings > Address / Port screen.

DOCUMENT ROOT DIRECTORY

This is your PopChart Server root directory. In other words, it is the location that PopChart Server was installed to (relative to the server that it was installed on, of course).

SERVER UP TIME

The amount of time that PopChart Server has been running. This is calculated from the last time PopChart Server was restarted.

IMAGES SERVED

The number of images that PopChart Server has generated. This number represents the number of images generated since PopChart Server was *installed*, not since it was last restarted.

SERVER STATUS

The status of your PopChart Server. The status will be either *running* or *stopped*.

DEBUG MODE

Indicates the type of debug mode that PopChart Server is running in. The available modes are *off*, *on*, and *verbose*.

You can change the debug mode via the Settings > Debug screen. For more information about debug mode, refer to "Running PopChart Server in Debug Mode" on page 14-5 of the *PopChart Server User Guide*.

SERVER COMMANDS

Below this heading, you will find the following buttons.

STOP Stops PopChart Server.

RESTART Stops PopChart Server and then starts it up again. You will need to restart PopChart Server any time you change its settings.

CLEAR CACHE Clears PopChart Server's cache. This is equivalent to the @_FLUSH server command. More information about the PopChart Server cache can be found in "Caching" on page A-3 of the *PopChart Server User Guide*.

Note: *If PopChart Server is stopped, you will only see the Start button (equivalent to the Restart button).*

RESOURCES

Below this heading you will find a list of resources where you can find help on using PopChart Server.

PERFORMANCE CHARTS

This screen shows a usage report for PopChart Server. You will see information on how long PopChart Server has been running and how many images it has served, as well as graphs of how many graphs have been served per hour and per day.

STATISTICS

This screen shows statistical information for the PopChart Server cache. You will see some or all of the following statistics, depending on whether caching is enabled. Caching can only be enabled for PopChart Server Enterprise, and it should be enabled via the *Settings > Cache / Connections* screen.

TOTAL HITS The number of times a client has requested an image from PopChart Server.

MAX SIZE The maximum size, in images, of the PopChart Server cache. This can be set using the Settings > Cache / Connections > Cache Size setting.

CURRENT SIZE The current size of the PopChart Server cache.

MEMORY USED The amount of memory currently being used for the PopChart Server cache.

HITS The number of requests to PopChart Server that "hit" the cache. In other words, the requested image is already cached, so it does not have to be generated.

MISSES The number of requests to PopChart Server that "miss" the cache. In other words, the requested image has to be generated because it is not in the cache.

EFFECTIVENESS The percentage of the time that requests to PopChart Server hit the cache.

TOTAL CACHE MEMORY The total amount of memory used by the cache.

MAXIMUM THREADS AT A TIME The maximum amount of threads that PopChart Server will run at any given time.

Note: *This screen shows the same information that you would see if you used the `@_LOG` server command.*

Below this information, if you have enabled logging, you will see the transaction log. this is a log of the most recent images that PopChart Server has served. You can enable the transaction log by clicking the Enable Data Logging check box on the Files > Log File Location screen.

VERSION / LICENSE

This screen shows version information about PopChart Server.

The following paragraphs describe the information that you will see on this page.

VERSION

This is the first line of information that you see on this page. The line will display the server edition (i.e. PopChart Server, PopChart Server Pro, or PopChart Server Enterprise) and version (e.g 4.0.5).

SERVER OS

The operating system of the machine on which PopChart Server is running (e.g. *Windows 2000*, *Linux*).

JAVA VM

The version of the Java Virtual Machine that is running PopChart Server (e.g. 1.3.0_01).

LICENSE KEY

If you did not enter your license key when you installed, or if you are upgrading to a new license, You can enter your license key in this box. After you have finished typing the license key, you should click Apply to save the change. You should then restart PopChart Server via the Server > Home > Restart command so that your new license key will take effect.

Note: *For security reasons, your license key will not be displayed after you enter it. If you need to find out what your license key is, you should examine the contents of the config/server_license.txt file.*

STOP ALL

This page allows you to stop all PopChart Server related processes, including the Administration Console. You will need to restart PopChart Server via the appropriate shortcut or executable before you can access the Administration Console again.

SETTINGS

These screens allow you to change configuration settings for PopChart Server.

Note: *Anytime you change a setting, you should restart PopChart Server using the Server > Home > Restart command.*

These screens include:

- Address / Port
- Cache / Connections
- Image Type
- Debug
- Memory
- Embedder Session
- Custom Server Settings
- Clustering

ADDRESS / PORT

This screen allows you to change the address and ports that PopChart Server uses.

SERVER ADDRESS

The address that PopChart Server is bound to at startup. This address must be a valid IP or domain address for the machine on which PopChart Server is running. The only time you should change this setting is when you are running PopChart Server on a machine with more than one IP address, and you want to specify the IP address that PopChart Server should listen for requests on.

SERVER PORT

The port on which PopChart Server accepts requests from web browsers. By default, this port is *2001*.

COMM ADDRESS

The address that PopChart Server's comm port is bound to at startup. This address must be a valid IP or domain address for the machine on which PopChart Server is running. The only time you should change this setting is when you are running PopChart Server on a machine with more than one IP address, and you want to specify the IP address to which PopChart Server should bind its comm port.

COMM PORT

The port that PopChart Server will use as its comm port. The comm port is a port that the PopChart Embedder uses to communicate with PopChart Server. This extra port facilitates faster communication with PopChart Server.

By default, the comm port is set to *2002*.

CACHE / CONNECTIONS

This screen allows you to modify performance settings for PopChart Server, particularly the cache size and threading.

ENABLE APPEARANCE FILE CACHING

Available only in PopChart Server Enterprise.

Instructs PopChart Server to cache appearance files.

Note: *The appearance file cache is different from the image cache, which you can set using Cache Size below.*

CACHE SIZE

Available only in PopChart Server Enterprise.

The maximum size, in images, of the PopChart Server cache. Generally, a number between 100 and 500 is appropriate.

For more information about caching in PopChart Server, refer to "Caching" on page A-3 of the *PopChart Server User Guide*.

Note: *This refers to the image cache, which is separate from the appearance file cache. The image cache is always on in PopChart Server Enterprise.*

MAXIMUM CONNECTIONS

The maximum number of simultaneous connections that PopChart Server can accept at any moment.

For more information about this setting, refer to "Increasing or Decreasing the Maximum Number of Connections" on page A-4 of the *PopChart Server User Guide*.

IMAGE TYPE

This screen controls settings for image generation.

DEFAULT IMAGE TYPE

The default image format for all images that PopChart Server generates. PopChart Server will return images in this format unless the request specifies otherwise. You can set

the default format to *GIF*, *Macromedia® Flash™*, and *SVG™*. The latter two options will be available only in PopChart Server Pro and PopChart Server Enterprise.

For information about the different image formats, refer to Chapter 14, "Image Formats." For more information on requesting image images in formats other than the default format, refer to "Changing the Image Format" on page 4-13 of the *PopChart Server User Guide*.

AUTO PNG

This option tells the PopChart Server to serve PNG images when possible. PopChart Server automatically checks to make sure that the client's browser supports PNG. If it does, it will send a PNG image.

If this option is enabled, you should also choose a PNG compression method. These are as follows:

Default This parameter offers the best compromise between the following two image compression settings.

Smallest This parameter will make the file as small as possible, but image compression will be extremely slow.

Fastest This parameter will compress the image as quickly as possible, but the file size will be significantly larger.

Though actual performance may vary from system to system, the latter two methods usually offer only a small increase in performance, while causing very slow image generation time and very large image sizes, respectively. Thus, default is usually the best option.

GIF GENERATION OPTIONS

These options control how GIF and PNG images are generated in PopChart Server. On certain UNIX compatible systems, you may need to instruct PopChart Server to use the Pure Java AWT to obtain optimal performance (or, in certain circumstances, any performance at all).

Note: *You can tell if changing these settings has any effect on font quality by restarting PopChart Server under the new settings and looking at the fonts in any GIF image returned by PopChart Server.*

Depending on the configuration of your system, these settings may also enhance server performance, even on Microsoft Windows and Macintosh systems. More information about the Pure Java AWT is available in "Using the Pure Java AWT and Custom Fonts" on page A-2 of the *PopChart Server User Guide*.

Note: *PJA is automatically included with Java VMs 1.4 and higher. Because of this, if you are using Java VM 1.4 or higher, you must disable PopChart Server's Pure Java AWT in the Administration Console.*

Use Pure Java AWT for GIF Images Instructs PopChart Server to use the Pure Java AWT (PJA) when generating GIF or PNG images. By default, this setting is enabled on UNIX compatible systems, and disabled otherwise.

PJA Setup By default, this is set to Normal. Under certain circumstances, however, you will achieve better results using the Class Override setting.

PJA Font Support By default, this is set to Default System Fonts. Under certain circumstances, however, you will achieve better results using the Use Corda Font Set setting.

DEBUG

This screen allows you to specify if you want to run PopChart Server in debug mode, as well as what type of debug mode to run it in. Debug mode is useful when trying to troubleshoot PopChart Server, but will most likely slightly decrease PopChart Server performance.

Debug information will be output to the Administration Console, and can be viewed by selecting Files > View Console Output.

For more information about debug mode, refer to "Running PopChart Server in Debug Mode" on page 14-5 of the *PopChart Server User Guide*.

ENABLE DEBUG MODE

Checking this box enables debug mode.

NORMAL In normal debug mode, PopChart Server outputs image format and size information to the console whenever it generates an image.

VERBOSE In verbose debug mode, PopChart Server outputs every request string it receives to the console.

Warning: Request strings can be very large. Since console output is stored in a text file on your server, running PopChart Server in debug verbose mode for an extended period of time can create a very large console output file. Therefore, you should run PopChart Server in debug verbose mode only for very limited periods of time.

MEMORY

This screen allows you to specify the maximum amount of memory (RAM), in megabytes, that PopChart Server can use.

Note that PopChart Server will not necessarily use this much memory. Memory usage will depend on how much traffic PopChart Server and the type of images it generates. However, PopChart Server will never use more than this amount of memory.

By default, this value is set to 64 megabytes. Usually, somewhere between 64 and 128 megabytes is adequate. However, if you have enabled the Pure Java AWT (via Settings > Image Type > GIF Generation Options), you should probably set this to at least 128 megabytes.

EMBEDDER SESSION

These settings control how the embedder session cache behaves.

The embedder session cache stores information and data that PopChart Embedder sends PopChart Server. Changing the embedder session cache will not usually have any significant effect on PopChart Server performance. For more information on the embedder session cache, refer to "Embedder Session Cache" on page A-5 of the *PopChart Server User Guide*.

SESSION TIME

Specifies the session time for the embedder session cache. This is the amount of time information will remain in the embedder session cache. The default value is *20*.

MAX MEMORY

Specifies the maximum amount of RAM, in megabytes, that the embedder session cache can use. The default value is *10*.

DIRECTORY

Specifies the directory that will be used to store information from the embedder session cache, should it run out of space in RAM. By default this directory is chart_root/keycache.

CUSTOM SERVER SETTINGS

This screen allows you to control miscellaneous settings for PopChart Server.

SHOW SPLASH SCREEN

Check this box if you want PopChart Server to show a splash screen as it starts off. By default this is checked, except on UNIX compatible systems.

AGENT TIMEOUT

The number of seconds the PopChart Agent will wait before timing out. The PopChart Agent is the program that manages PopChart Server and the Administration Console. If

the PopChart Agent is unable to start PopChart Server after the specified number of seconds, it will return an error and shut down. The default value is 60. You may need to raise this value if you are running on a slower system.

EXTRA COMMANDS

You can specify additional server configuration settings for PopChart Server. Since you can manipulate all documented server configuration settings from within the Administration Console, you will probably not need to enter anything into this box. To learn more about server configuration settings, refer to Chapter 3, "Server Configuration Settings."

CLUSTERING

Available only in PopChart Server Enterprise.

This screen allows you to configure clustering for your PopChart Server.

To learn more about clustering, refer to Chapter 13, "Clustering," in the *PopChart Server User Guide*.

ENABLE CLUSTERING

This turns clustering on.

Auto Cluster Enables auto-clustering. PopChart Server will scan through the entire 255.255.255.0 subnet for other clustering-enabled servers. For example, if PopChart Server's address is 10.0.1.28 and you enable auto-clustering, PopChart Server will cluster with any server whose address is 10.0.1.x (e.g. 10.0.1.1, 10.0.1.2, etc.). In order for this to work, all PopChart Servers must be using the same comm port number.

Manual Cluster Enables manual clustering. PopChart Server will attempt to contact the servers listed in the Cluster Server List and request a complete list of all currently clustered servers.

CLUSTER SERVER LIST

A comma separated list of servers that will be in the server cluster. You should specify their comm port address (e.g. 10.0.1.7:2002).

You do not have to list all of the servers. In fact, you only need to list one valid server address, although it's safer to list more. PopChart Server will download the list of remaining servers in the cluster from the first server that it is able to contact.

You will need to enable clustering under the Manual Cluster setting for this list to take effect.

USING A SOFTWARE/HARDWARE LOAD BALANCER

This tells PopChart Server to compensate for software or hardware balancing between the redirector and server cluster.

Warning: Only use this setting when you are doing some sort of load balancing between a redirector and the server cluster.

LIST OF CLUSTERED PopChart ServerS

This list shows all of the PopChart Servers that are in the current cluster.

FILES

The screens in this section show you log information for PopChart Server and control how that log information is saved. Log information serves two purposes: 1) it helps give you an idea of what type of traffic your PopChart Server is handling; and 2) it will help you analyze and correct problems with your PopChart Server configuration.

Note: *Anytime you change a setting, you should restart PopChart Server using the Server > Home > Restart command.*

The screens in this section include:

- Log File Location
- View Console Output
- List Appearance Files
- Upload Appearance Files
- Upload Font Files

LOG FILE LOCATION

This screen allows you to specify locations for your log files.

ENABLE DATA LOGGING

This instructs PopChart Server to keep a record of the most recent images it generates. This record is called its transaction log. The log will appear at the bottom of the Server > Statistics screen.

NUMBER OF TRANSACTIONS TO LOG

This is the number of transactions that PopChart Server will store in its transaction log.

DAILY LOG FILE LOCATION

PopChart Server keeps independent daily and monthly access logs, which it uses to create the graphs you see in the Server > Performance Charts screen. You can change the directory where PopChart Server stores these logs by changing the path in this text box.

This path will be relative the PopChart Server root directory. Very few users will need to change this setting.

VIEW CONSOLE OUTPUT

This is what you would see if you were running PopChart Server from the command line (such as when you run the PopChart Server With Console icon). It shows you real-time console messages from PopChart Server. It will keep track of all messages since PopChart Server was last restarted.

LIST APPEARANCE FILES

This screen allows you to view the appearance files that are stored in any folder under PopChart Server's document root directory.

CURRENT DIRECTORY

This is the directory you are currently viewing.

APPEARANCE FILE LIST

This is a list of all files in this directory. Some of these files (e.g. readme.txt) may not be appearance files.

CHANGE DIRECTORY

This allows you to change the current directory. You must enter a path relative to the PopChart Server root directory. You *cannot* browse up a directory (i.e. enter any path with a .. in it).

UPLOAD APPEARANCE FILES

Note: *You do not need to restart PopChart Server after uploading appearance files.*

This screen allows you to upload appearance files to PopChart Server's document root directory, or any of its subfolders.

DESTINATION DIRECTORY

You should enter the location (path only) to which you would like to save the appearance file in this box.

This location is relative to the document root. For example, entering apfiles would upload the appearance file to the apfiles directory. Leaving this box blank will upload the file to the document root. You *cannot* enter any path that is not under the document root directory (i.e. a path with a .. in it).

FILE TO UPLOAD

Enter the appearance file that you want to upload in this box. Alternatively, you can click the Browse button to locate your appearance file. The path to this file will be relative to your local machine.

Once you have selected an appearance file, click Upload to upload the file.

UPLOAD FONT FILES

Note: *You do not need to restart PopChart Server after uploading font files.*

This screen allows you to upload font files that you have converted using the PopChart Font Converter. For more information about the Font Converter, refer to "Importing Custom Fonts" on page 9-3 of the *PopChart Server User Guide*.

CURRENTLY INSTALLED CUSTOM FONTS

This is a list of all the font files currently loaded into PopChart Server.

SPECIFY FILE TO UPLOAD

This allows you to specify a font file to upload to PopChart Server. You can type in a file name directly, or browse to it by clicking on the browse button. Make sure that the font you upload has a .fsd extension.

Once you have selected a font file, click Upload to upload the file.

SECURITY

The screens in this section allow you to change security settings for your PopChart Server. These screens include:

- Change Password
- Path / URL Permissions

CHANGE PASSWORD

This screen allows you to set your password for PopChart Server.

Your password is necessary for viewing the Administration Console, as well as executing certain server commands, including @_SAVE and @_LOG.

By default, your password is *password*. You should change this password immediately after installing PopChart Server. We recommend that your new password be at least eight letters long and contain at least one lower case letter, one upper case letter, and a number.

After you change your password, you must restart PopChart Server.

PATH / URL PERMISSIONS

Note: *You do not need to restart PopChart Server after changing path and URL permissions.*

This screen allows you to edit the path.xml file, which controls read and write permissions for PopChart Server.

More exactly, this file controls which server-side directories PopChart Server can read appearance files from and save images to. It also controls valid callback domains for appearance files and graph data. *In order for PopChart Server to read any appearance or data file, PopChart Server must have permission to read from the location where that file is stored.*

Note: *By default, the only URL you can read data or appearance files from is the machine that PopChart Server is running on. The only server-side directories that PopChart Server can read from are those in its document root* (usually chart_root).

For a more detailed explanation of PopChart Server security, refer to "Setting Path Permissions" on page 3-37 of the *PopChart Server User Guide*.

For the complete path.xml document specification, refer to Chapter 10, "Path Settings."

To add permission to read from a specified URL

1 **Copy the following text (which also appears in the Administration console) and paste it immediately above the last line:**

```
<Map Name="MyAppServer" Path="appservername.mycompany.
  com" Action="allowDomain"/>
```

2 **Replace MyAppServer (the value of the Name attribute) with the name you wish to give to this mapping.**

This name is for descriptive purposes only, and is entirely up to you. In fact, this step is entirely optional.

3 **Replace appservername.mycompany.com (the value of the Path attribute) with the name or IP address of the host that you want to allow PopChart Server to read from.**

This will allows PopChart Server to read any file that comes from the specified host. For example, if we specify www.mycoolstats.com, we could read from sources such as http://www.mycoolstats.com/data/110899.html, http://www.mycoolstats.com/renderer?name=bar&apfile=26, etc.

You can also use wildcards. For instance, *.corda.com would allow any host in the corda.com domain (www.corda.com, popchart.corda.com, etc.). Similarly, 10.0.*.* would allow PopChart Server to read from any IP address that begins with 10.0.

4 **Click Apply to apply your changes. You do not need to restart PopChart Server.**

To add permission to read from a specified local path

1 **Copy the following text and paste it immediately above the last line:**

```
<Map Name="Read" Path="./path" Action="Load"/>
```

Note: *This text is different from the text that appears in the Administration Console.*

2 **Replace Read (the value of the Name attribute) with the name you wish to give to this mapping.**

This name is for descriptive purposes only, and is entirely up to you. In fact, this step is entirely optional.

3 **Replace ./path (the value of the Path attribute) with the local path that you want to allow PopChart Server to read from.**

If you precede the path with ./, it will be assumed to be relative to the document root. Otherwise, it will be assumed to be an absolute path accessible to the machine that PopChart Server is running on. You can also put a wildcard at the end of a path to indicate that PopChart Server can also read form any of its subdirectories.

For example, if you set this value to F:\InetPub\mydata*, you are giving PopChart Server permission to read anything from the F:\InetPub\mydata directory, as well as any of its subdirectories. If you set this value to ./data, you are giving PopChart

Server permission to read from the data directory in the document root, but none of its subdirectories.

4 **Click `Apply` to apply your changes. You do not need to restart PopChart Server.**

CHANGING SERVER AND ADMINISTRATION PORT SETTINGS

The PopChart Administration Console uses two ports to communicate with PopChart Server. These ports, by default, are 2003 and 2004. Since PopChart Server runs on ports 2001 and 2002, this means you will need to have ports 2001 through 2004 free in order to run PopChart Server under its default settings.

You can change the ports for PopChart Server via the *Settings > Address / Port* screen in the Administration Console. However, if you cannot start the Administration Console, you will need to modify these ports externally.

The list below describes each port that the Administration Console uses and how to change it. You can change these ports by editing the PCAgent.cfg file in the config directory of your PopChart installation. You will need to restart PopChart Server after making any changes.

Note: *To learn more about the PCAgent.cfg file, refer to "Configuration File Format" on page 3-2.*

SERVER PORT (2001)

This is the port that the PopChart Server "runs" on. Clients will request images from PopChart Server over this port. This port is the only port that you have to open up if you want to clients to be able to view PopChart images from outside of your firewall.

You can change this port by modifying the following line in the PCAgent.cfg file:

```
-port 2001
```

You should replace the number 2001 with the number of the port that you want the Administration Console to run on.

If your computer has multiple IP addresses and you need to bind PopChart Server to a specific address, you could also enter it here. For example, if you were binding PopChart Server to port 2001 at the address 10.0.1.1, you would enter the following in the PCAgent.cfg file:

```
-port 10.0.1.1:2001
```

COMM PORT (2002)

This is the port that the PopChart Embedder (except the JavaScript version) uses to communicate with PopChart Server.

You can change this port by modifying the following line in the PCAgent.cfg file:

```
-commport 2002
```

You should replace the number 2002 with the number of the port that you want the Administration Console to run on.

If your computer has multiple IP addresses and you need to bind the comm port to a specific address, you could also enter it here. For example, if you were binding the comm port to port 2002 at the address 10.0.1.1, you would enter the following in the PCAgent.cfg file:

```
-port 10.0.1.1:2002
```

POPCHART AGENT PORT (2003)

This port facilitates communication between PopChart Server and the Administration Console.

You can change this port by modifying two lines in the PCAgent.cfg file. The first line is in the `[PCISStartupArgs]` section of the file, while these second line is in the `[CASStartupArgs]` section of the file. Both lines are the same:

```
-agentcommport 2003
```

You should replace the number 2003 with the number of the port that the agent should use.

ADMINISTRATION SERVER PORT (2004)

This is the port that the Administration Console "runs" on. In other words, you access the Administration Console via this port. This port is the only port that you have to open up if you want to administer your PopChart Server from outside of a firewall.

You can change this port by modifying the following line in the PCAgent.cfg file:

```
-casport 2004
```

You should replace the number 2004 with the number of the port that you want the Administration Console to run on.

If your computer has multiple IP addresses and you need to bind the Administration Console to a specific address, you could also enter it here. For example, if you were binding the Administration Console to port 2004 at the address 10.0.1.1, you would enter the following in the PCAgent.cfg file:

```
-port 10.0.1.1:2004
```

3

SERVER CONFIGURATION SETTINGS

This section discusses PopChart Server configuration settings. These settings control PopChart Server's operating environment, including the server port and password.

Configuration settings are controlled by the configuration file, PCAgent.cfg, which is located in the config directory. Under most circumstances, you should avoid modifying the PCAgent.cfg file directly. The PopChart Server Administration Console allows you to change all of the settings in this file without you having to edit the file manually. You should modify PCAgent.cfg directly only when you are having problems with the Administration Console.

Note: *Settings that are not available through a dialogue in the Administration Console can still be added through the Administration Console by listing them in the Extra Commands box on the Settings > Custom Server Settings page. Each setting should be listed on a separate line, using the syntax given in the "Configuration Settings Reference" section at the end of this chapter.*

CONFIGURATION FILE FORMAT

Example 3.1 shows a typical configuration file.

Example 3.1 Configuration File (PCAgent.cfg)

```
[PCISStartupArgs]
-port 2001
-commport 2002
-pw password
-agentcommport 2003
-noscreen

[CASStartupArgs]
-casport 2004
-agentcommport 2003
```

Note: *You can comment or uncomment a line in the configuration file by adding or removing a semi-colon in front it.*

As the example shows, the configuration file is divided into two sections.

The first section contains the main configuration settings. The first section begins with the line `[PCISStartupArgs]`. After that, each startup setting should be listed on a separate line, preceded by a hyphen. You can add a setting by placing it on a new line at the end of the section. The only required settings are `-port`, `-commport`, and `-agentcommport`.

The second section, beginning with `[CASStartupArgs]`, contains settings for the Administration Console. There are only a few settings that go in this section, and two of them are required: `-casport`, and `-agentcommport`.

CONFIGURATION SETTINGS LIST

Following is a list of PopChart Server startup settings. You can jump to a description of any setting by clicking on it.

CONFIGURATION SETTINGS REFERENCE

This section describes each configuration setting in detail.

agentcommport

Syntax `-agentcommport portnumber`

Description This setting specifies the PCAgent port. The PCAgent port facilitates communication between the Administration Console and PopChart Server.

The `-agentcommport` setting is followed by the number of the port on which the PCAgent will run. The PCAgent port must be set to a different port than the one specified in the `-port`, `-commport`, or `-casport` settings. The default setting is *2003*.

This setting should be listed under both the `[CASStartupArgs]` and `[PCISStartupArgs]` sections of the configuration file.

agenttimeout

Syntax `-agenttimeout seconds`

Description This setting specifies the number of seconds PopChart Agent will wait for PopChart Server to start.

If PopChart Server is unable to start in the specified time period, all PopChart processes will shut down and an error message will be sent to the screen or to the console. The PCAgent port facilitates communication between the Administration Console and PopChart Server. The default setting is *60*.

This setting should be listed under the `[CASStartupArgs]` section of the configuration file.

autoPNG

Syntax `-autoPNG { default | smallest | fastest }`

Description This setting tells PopChart Server to serve PNG images when possible. PopChart Server automatically checks to make sure that the client's browser supports PNG. If it does, it will send a PNG image. Otherwise, it sends a GIF image. This applies only when PopChart Server has been told to serve a GIF image, either by selecting GIF as the default image type (see the `-type` setting), or by specifically telling it to serve a GIF image in the Server Command String. Otherwise, it will send a Flash image, regardless of whether the `autoPNG` setting has been specified.

When specifying this setting, you must also include one of the following parameters: `default`, `smallest`, or `fastest`. These control the PNG compression method. The smallest parameter tells PopChart Server to make the file as small as possible, while the fastest parameter tells PopChart Server to compress the image as quickly as possible. Though actual performance may vary from system to system, these two methods usually offer only a small increase in performance, while demanding very slow image generation time and very large image sizes, respectively. The default parameter offers the best compromise between these two settings.

cache

Available only in PopChart Server Enterprise.

Syntax `-cache cachesize`

Description This setting enables PopChart Server's cache. When caching is enabled, PopChart Server stores server request strings and corresponding image in a cache. When the same request is received, PopChart Image Server Pro sends back the previously generated image.

Since image generation is the most time-consuming process for PopChart Server, caching can tremendously increase the speed with which PopChart Server serves images, especially if it receives many requests for the same image.

The `-cache` setting should be followed by a number, indicating the number of images that should be stored in the cache. For example, `-cache 100` allocates space for up to 100 images.

The cache size can be as large as you desire. However, keep in mind that the cache uses RAM and not disk space. Thus, if you keep a very large cache, you will need a lot of RAM. Cache sizes between 100 to 1000 images are typical.

casport

Syntax `-casport portnumber`

Description This setting specifies the Administration Console port. This port is the port that the Administration Console runs on.

The `-casport` setting is followed by the number of the port on which the Administration Console will run. The PCAgent port must be set to a different port than the one specified in the `-port`, `-commport`, or `-agentcommport` settings. The default setting is *2004*.

This setting and the `-agentcommport` should be listed under the `[CASStartupArgs]` of the configuration file.

commport

Syntax `-commport [ip:]portnumber`

Description This setting specifies the port that the PopChart Embedder will use to send data and commands to PopChart Server.

The `-commport` setting is followed by the number of the port on which the commport will run. The commport must be set to a different port than the one specified in the `-port`, `-commport`, or `-casport` settings. The default setting is *2002*.

You need to supply the commport number in the`internalCommPortAddress` attribute whenever you use the PopChart Embedder.

This command will also accept an IP address, which allows you to bind the comm port to a specific address if your system has more than one IP address. If this is the case, simply enter the address, followed by a colon, followed by the port number. For example, if you were binding the comm port to port 2002 at the address 10.0.1.1, you would enter the following in the PCAgent.cfg file:

```
-port 10.0.1.1:2002
```

By default, PopChart Server binds its comm port to localhost.

debug

Syntax `-debug`

Description This setting instructs PopChart Server to log debug information in the console or log file. You may find it useful to use the `-log` setting in conjuction with `-debug`.

debugverbose

Syntax `-debugverbose`

Description This setting works exactly like the `-debug` setting. Additionally, it records every request string exactly as it comes to PopChart Server. You may find it useful to use the `-log` setting in conjuction with `-debugverbose`.

Warning: Since server request strings are often larger than a kilobyte, the `-debugverbose` setting has the potential to create a very large Administration Console log file if you let PopChart Server run for a long time. If the Administration Console is running, run PopChart Server under the `-debugverbose` setting only for short stints while debugging.

keydirectory

Syntax `-keydirectory path`

Description Specifies the directory that will be used to store objects from the comm port key cache, should it run out of space in RAM. By default this directory is chart_root/keycache.

For more information, refer to "Embedder Session Cache" on page A-5 of the *PopChart Server User Guide*.

keymaxmegs

Syntax `-keymaxmegs megabytes`

Description Specifies the maximum amount of RAM, in megabytes, that the comm port key cache can use. The default value is *10*.

For more information, refer to "Embedder Session Cache" on page A-5 of the *PopChart Server User Guide*.

keysessiontime

Syntax `-keysessiontime minutes`

Description Specifies the session time for objects in the comm port key cache. This is the amount of time an object will remain in the key cache. The default value is *20*.

For more information, refer to "Embedder Session Cache" on page A-5 of the *PopChart Server User Guide*.

language

Syntax `-language languagecode`

Description Specifies the default language settings for all PopChart images and descriptive text. If this value is not set, *EN* (English) is assumed. Currently, *EN* (English) is the only valid setting.

log

Syntax `-log num_entries`

Description This setting tells PopChart Server to keep a log of the latest transactions. This log is different from the one kept by the PopChart Server Administration Console. It is also in addition to the hourly usage report that PopChart Server keeps by itself. You should investigate both of those settings before deciding to use this setting. For more information,

refer to "Using the Server Log and Console Output" on page 3-21 of the *PopChart Server User Guide*.

The `-log` setting should be followed by a number, `num_entries`, indicating how many transactions to keep track of. Only the `num_entries` most recent transactions will be kept in the log. All others will be removed. Remember when setting this size that, especially when `-debug` or `-debugverbose` is set, each transaction can be very large. You should therefore keep this number small. Between *40* and *100* is usually adequate.

The log can be accessed from a web browser by requesting the page: `http://yourserver/?@_LOG@_PWpassword`, where *yourserver* is the address and port that your PopChart Server is running on, and *password* is the password you specified in the Administration Console or with the `-pw` setting. For security reasons, you must supply a password in order to view the log.

You may find it useful to use the `-debug` or `-debugverbose` setting in conjuction with `-log`.

nosplash

Syntax `-nosplash`

Description This setting instructs PopChart Server not to show a splash screen as it launches. By default, this option is turned off for Linux and UNIX compatible systems, and turned on for all others.

nopwforsave

Syntax `-nopwforsave`

Description This setting instructs PopChart Server not to require a password for the `@_SAVE` server command. This setting is included for backwards compatibility only. Setting it may potentially compromise the security of your system.

pjafont

Syntax `-pjafont { platform | custom }`

Description Specifies Pure Java AWT fonts to use with PopChart Server. If set to *platform*, PopChart Server will use the system fonts. If set to *custom*, PopChart Server will use its own custom font set. Try the *custom* setting if fonts in GIF images look poor.

If you enable this setting, PopChart Server will automatically use the Pure Java AWT (refer to "Using the Pure Java AWT and Custom Fonts" on page A-2 of the *PopChart Server User Guide*).

pjawt

Syntax `-pjawt { normal | override }`

Description Specifies Pure Java AWT setup options. Unless PopChart Server has difficulty displaying GIF images, set this to normal. Otherwise, try *override*.

If you enable this setting, PopChart Server will automatically use the Pure Java AWT (refer to "Using the Pure Java AWT and Custom Fonts" on page A-2 of the *PopChart Server User Guide*).

port

Syntax `-port [ip:]portnumber`

Description This setting specifies the port that PopChart Server will use to serve PopChart images.

Unless this port number is *80*, all HTTP requests to PopChart Server will need to specify this port. For example, assuming the port is set to *2001* (default setting), and that you are running PopChart Server on your localhost, you would need to request images from http://localhost:2001.

The `-port` setting should be followed by a number that indicates the port on which PopChart Server will run. The setting is always required. If it is not specified, PopChart Server will not run correctly.

This command will also accept an IP address, which allows you to bind the server port to a specific address if your system has more than one IP address. If this is the case, simply enter the address, followed by a colon, followed by the port number. For example, if you were binding PopChart Server to port 2002 at the address 10.0.1.1, you would enter the following in the PCAgent.cfg file:

```
-port 10.0.1.1:2002
```

By default, PopChart Server is bound to localhost.

pw

Syntax `-pw password`

Description This setting specifies a password for PopChart Server.

For security reasons, certain server commands (e.g. `@_SAVE` and `@_LOG`) require a password. The Administration Console also requires a password.

The default password is *password*. You should change this password, either through the Administration Console or by changing the argument passed to the `-pw` setting, immediately upon installing PopChart Server.

report

Syntax `-report`

Description This setting instructs PopChart Server to report certain information to the console as it generates each image. This information includes the current date, the time required to create the image, and the image file size. You will see this information in the log for the Administration Console, or in a console window if PopChart Server is running from within one.

resroot

Syntax `-resroot path`

Description This setting tells PopChart Server where to look for certain resources, such as the folder that contains the font shape files or the license key.

You will usually not need to use this setting, since the resources are all located by default in the PopChart Server root directory. However, if you move these resources, you will need to use this setting.

The `-resroot` setting requires a `path` argument, which tells PopChart Server the path to its resources. This path is relative to the PopChart Server root directory, or you can use an absolute path. In this path, PopChart Server should find the lib and config folder. If logging is enabled, it will also need to find the logs folder.

root

Syntax `-root path`

Description This setting specifies the document root for PopChart Server. The document root is where PopChart Server looks for appearance files, command files, data, and images. If not specified, the default document root will be the chart_root folder.

The `-root` setting requires a `path` argument, which tells PopChart Server the path to the document root. This path is relative to the PopChart Server root directory, or you can use an absolute path.

type

Syntax
```
-imagetype { eps | gif | flash | pdf | png | svg |
     wbmp }
```

Description Specifies the default image format. The default format is assumed to be *flash* if this setting is not specified in the configuration file.

For an explanation of the image formats that PopChart Server can produce, refer to Chapter 14, “Image Formats.”

4

PopChart Embedder API

This chapter discusses the API for the PopChart Embedder.

The PopChart Embedder is a utility that simplifies the process of embedding a PopChart image into a web page. It is available in the following formats: Java, JavaBean, JavaScript, .NET, PHP, and COM (Component Object Model). It can be used in a variety of environments, including Java Server Pages, Active Server Pages, ASP.NET, Java Servlets, and ColdFusion.

The programming interface for the PopChart Embedder is essentially identical for all of the languages and environments mentioned above. Thus, except where there is an obvious difference in the way the PopChart Embedder behaves in a certain language or environment, this chapter makes no distinction between the various flavors of the PopChart Embedder.

Note: *In versions of the PopChart Server prior to the 4.0 release, the PopChart Embedder APIs were not consistent. In order to bring about this consistency, many methods have been deprecated. For more information, refer to the section entitled "Deprecated Methods" on page 4-31.*

To simplify the examples in this chapter, we rely mostly on Java coding conventions, but the equivalent C++ code should be fairly obvious. For example, most of the attributes are listed as type `String`, which, of course, should be type `string` (lower-case) in C++ or C#. You can also use a wide character array (`w_char*`) in place of `String`.

Likewise be aware that Active Server Pages use VBScript, which requires you to *not* have a semi-colon at the end of each line of code. Be sure to remove semi-colons from any example code that you use. It also requires any functions that do not return a value (`void` methods) to *not* have parenthesis around the parameters.

For example, we show the `addHTMLTable()` syntax to be:

```
myPopChart.addHTMLTable("graph","title");
```

This is perfectly valid in every language except VBScript. So in ASPs, you would need to use the following syntax:

```
myPopChart.addHTMLTable "graph", "title"
```

For most ASP developers this will be an obvious conversion process.

Finally, in PHP, you will always use pointer notation when calling methods or accessing attributes. Also, variable names must always begin with a dollar sign. For example, consider the following command in Java.

```
myPopChart.appearanceFile = "bar.pcxml";
```

The equivalent code in PHP would be as follows:

```
$myPopChart->appearanceFile = "bar.pcxml";
```

Again, for most PHP developers, this will be an obvious conversion process.

USING THE PopChart Embedder

Embedding a PopChart image is a four step process. Although the exact code that you use will vary depending on the language or environment that you are programming in, you will still follow this same process.

Table 4.1 shows you where you can find the PopChart Embedder utility for each of the languages and environments mentioned above.

TABLE 4.1 PopChart Embedder Locations

Language / Environment	File Name
Java	dev_tools/java_embedder/PopChartEmbedder.jar
JavaBean	dev_tools/java_embedder/PopChartEmbedder.jar
JavaScript	lib/fallback.js, lib/gifonly.js [a]
COM	dev_tools/COM_embedder/PCEmbedder.dll
.NET	dev_tools/dotnet_embedder/PCNetEmbedder.dll
PHP	dev_tools/php_embedder/PopChartEmbedder.php
C++	dev_tools/cpp_embedder[b]

a. These are the templates for the JavaScript PopChart Embedder. The JavaScript PopChart Embedder, itself, should be loaded from PopChart Server (refer to "Importing the PopChart Embedder Library" on page 4-5 in the *PopChart Server User Guide*).

b. The C++ PopChart Embedder is currently of beta quality and is undocumented. It requires an Enterprise or Pro license.

To embed a PopChart image with the PopChart Embedder

1 **Include or import the PopChart Embedder class into your operating environment.**

In Java, for example, you would do this by making sure that the PopChartEmbedder.jar file is in your classpath and including the statement:

```
import com.corda.pcis.PopChartEmbedder;
```

2 **Instantiate a PopChart Embedder object.**

In Java, for example, you would use this statement:

```
PopChartEmbedder myPopChart = new PopChartEmbedder();
```

3 **Tell the PopChart Embedder object where PopChart Server is located.**

You need to give PopChart Embedder two addresses—your external server address (the PopChart Server address that the client will use) and the internal comm port

address (the PopChart Server that the PopChart Embedder will use). For example, in Java you would say:

```
myPopChart.externalServerAddress =
    "http://localhost:2001";
myPopChart.internalCommPortAddress =
    "http://localhost:2002";
```

4 Send data and formatting options to your PopChart image.

You can do this by manipulating the attributes (e.g. `appearanceFile`, `imageType`, `width`) of your PopChart Embedder object. For example, if you have named your PopChart Embedder object `myPopChart`, you can specify an appearance file for your PopChart image with this Java statement:

```
myPopChart.appearanceFile = "apfiles/bar.pcxml";
```

5 Request and output the embedding HTML.

The `getEmbeddingHTML()` will return a string with the HTML necessary to embed your PopChart image, which you can then write to your web page. For example, you could write the embedding HTML in a Java Server Page with this statement:

```
<%= myPopChart.getEmbeddingHTML() %>
```

You can learn more about how to set up and use the PopChart Embedder in different web environments in Chapter 4 and Chapter 5 of the *PopChart Server User Guide*.

CLASS SUMMARY

The following attributes, methods and constants are defined for the PopChart Embedder object. Click on an attribute or method for a description of it.

ATTRIBUTES (PROPERTIES)

- String **appearanceFile**
- boolean **appendServerInfoSlash**
- String **bgColor**
- String **clusterMonitorAddress**
- String **externalServerAddress**
- String **extraHTMLAttributes**
- String **extraPCSCommands**
- String **fallback**
- int **height**
- String **htmlHeight**
- String **htmlWidth**
- String **imageType**
- String **internalCommPortAddress**
- String **language**
- boolean **makeFullRequest**
- int **maxRequestLength**
- String **password**
- String **pcScript**
- boolean **returnDescriptiveLink**
- String **svgTemplate**
- boolean **useCache**
- boolean **useLogData**
- String **userAgent**
- int **width**

METHODS

- void addHTMLTable(String,String)
- void addObjectParamTag(String, String)
- void addPCXML(String)
- String getEmbeddingHTML()
- byte[] getImageData()
- String getPCEmbedderVersion()
- void loadCommandFile(String)
- void loadData(String,String,String,String)
- void loadServerSideImage(String)
- void loadPCXML(String)
- void resetPCXML()
- void saveImageToAppServer(String, String)
- void saveImageToPopChartServer(String)
- void setData(String,String)
- void setDBQuery(String, String, String, String, String, String)
- void setResultSet(String,ResultSet)

CONSTANTS

IMAGE TYPES

- EPS = "eps"
- GIF = "gif"
- FLASH = "flash"
- PDF = "pdf"
- PNG = "png"
- SVG= "svg"
- WBMP=" wbmp"

FALLBACK METHODS

- NONE= "none"
- LOOSE= "loose"
- STRICT= "strict"

LANGUAGE CODES

- EN= "EN"

PopChart Embedder ATTRIBUTES

This section describes each of the attributes of PopChart Embedder. These attributes specify data and formatting settings for your PopChart image.

Note: *The code in the syntax section of these descriptions assumes that you are manipulating a PopChart Embedder object named* `myPopChart`.

String appearanceFile

Syntax
```
myPopChart.appearanceFile = "apfiles/bar.pcxml";
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="appearanceFile"
     value="apfiles/bar.pcxml" />
```

Description

Specifies the location of the appearance file for your PopChart image.

The appearance file should have a .pcxml extension. The file location should be relative to PopChart's document root. It can also be an absolute path or a URL. You may also need to make sure that your path.xml file gives PopChart Server permission to read appearance files from the location where your appearance file resides (see "Setting Path Permissions" on page 3-37 of the *PopChart Server User Guide*).

For more information about appearance files, refer to "About Appearance Files" on page 4-11 of the *PopChart Server User Guide*.

Note: *You can still use binary appearance files from PopChart Builder 3.x. You do not need to give these a .pcxml extension.*

boolean appendServerInfoSlash

Syntax
```
myPopChart.appendServerInfoSlash = false;
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="appendServerInfoSlash" value="false"
     />
```

Description

Specifies whether or not PopChart Embedder should automatically add a slash if there is not one at the end of the web address. By default, this is set to *true*.

Important: *You should not set this attribute to false unless you are having problems with a redirector. Most people will not need to use this attribute.*

String bgColor

Syntax
```
myPopChart.bgColor = "FFFFFF";
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="bgColor" value="FFFFFF" />
```

Description

Specifies the background color of your PopChart image.

The value of this string should be a six digit hexadecimal number, representing the RGB value of the desired color. For example, *"FFFFFF"* represents white, while *"C7C700"* represents the green-yellow color used on this page.

Note: *Do NOT prepend the color value with a pound # sign, as is often done on web pages.*

String clusterMonitorAddress

Available only in PopChart Server Enterprise.

Syntax
```
myPopChart.clusterMonitorAddress =
     "localhost/clusterMonitor";
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="clusterMonitorAddress"
     value="localhost/clusterMonitor" />
```

Description

Specifies the address and port of your PopChart cluster monitor.

The value of this string should be the location of the cluster monitor. Usually, your cluster monitor will run on your local machine as a servlet or web module.

This attribute only has meaning when you are using the clustering capabilities of PopChart Server Enterprise (see Chapter 13, "Clustering," in the *PopChart Server User Guide*).

Note: *When using a cluster monitor, do not set the `internalCommPortAddress` attribute.*

String externalServerAddress

Syntax
```
myPopChart.externalServerAddress =
     "is.mycompany.com:2001";
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="externalServerAddress"
     value="is.mycompany.com:2001" />
```

Description

Specifies the external address and port of your PopChart Server.

The value of this string should be an IP address or domain name, followed by your server port number (e.g. *10.0.0.1:2001*, *imageserver.mycompany.com:2001*). By default, your server port is *2001*.

Clients will retrieve the generated image from this address. The address must be accessible from the client, meaning that if you are serving images outside of your company intranet, you may need to configure your firewall so that the address or port is open.

If you don't want to open another port on your firewall, you can use an HTTP redirection module to get to PopChart Server (see Chapter 12, "HTTP Redirection," in the *PopChart Server User Guide*). In this case, you should set this attribute to the location of the redirector (e.g. *www.mycompany.com/popchart*). Note that you do not have to specify a port in this case, as port *80* is assumed.

The external server address is different from the internal comm port address, which is the address that your web application server uses to communicate with PopChart Server (refer to the `internalCommPortAddress` attribute). Both the external and internal addresses must be specified when generating a PopChart image (except when using the JavaScript embedder, which does not use a comm port, or when setting the `clusterMonitorAddress` while clustering).

For more information about configuring PopChart Server ports and addresses, refer to "Setting Server Address and Ports" on page 3-18 of the *PopChart Server User Guide*.

String extraHTMLAttributes

Syntax

```
myPopChart.extraHTMLAttributes = attributes;
```

JavaBean Syntax

```
<JSP:setProperty name="myPopChart"
     property="extraHTMLAttributes" value=attributes
     />
```

Description

Specifies additional attributes to place within the `<object>`, `<embed>`, or `<img>` tag that embeds the generated PopChart image.

The value of this string should be a list of attributes and styles, in the exact same format as the would appear within an HTML tag on a web page (i.e. *"attribute1='value1' attribute2='value2'"*).

Note: *If you need to use the double quotes character " inside of this string, be sure to escape it using the appropriate escape character for your language or scripting environment. For example, in most languages, you would need to use a back-slash (e.g.* `attribute1=\"value1\"`*). Some environments may require you to repeat the character twice instead (e.g.* `attribute1=""value1""`*).*

This attribute is useful when you need to format your PopChart image with HTML attributes that don't have an equivalent PopChart Embedder attribute. Some of the more common attributes that fall into this category are `style`, `align`, `name`, `vspace`, and `hspace`. Example 4.1 shows how you would set these attributes with `extraHTMLAttributes`:

Example 4.1 Setting Extra HTML Attributes

```
myPopChart.extraHTMLAttributes = "style='position:absolute;
              top:120' align='center' vspace='2' hspace='2'";
```

String extraPCSCommands

Syntax
```
myPopChart.extraPCSCommands = commandstring;
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="extraPCSCommands" value=commandstring
     />
```

Description

Specifies additional server commands to send to PopChart Server.

The value of this string should be a server command string (refer to "Server Command Strings" on page 6-2).

This attribute is useful when you need to send commands to PopChart Server that you do not have access to via the PopChart Embedder. Although there were more such commands in the past, currently, the only documented command that is inaccessible in the PopChart Embedder API is `@_FLUSH`. Example 4.2 shows how you would set this command with `extraPCSCommands`:

Example 4.2 Setting Extra Server Commands

```
myPopChart.extraPCSCommands = "@_FLUSH";
```

String fallback

Syntax
```
myPopChart.fallback = {"NONE" | "LOOSE" | "STRICT"};
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="fallback" value="{none|loose|strict}"
     />
```

Description

Specifies the fallback method for the generated PopChart image.

The value of this string should be one of the following three strings: `"NONE"`, `"LOOSE"`, or `"STRICT"`. These strings represent the different fallback methods—*none*, *loose*, and *strict*, respectively.

For an explanation of PopChart Server's best-image fallback feature and the various fallback methods, refer to "Best Image Fallback" on page 4-13 of the *PopChart Server User Guide*.

int height

Syntax
```
myPopChart.height = 400;
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart" property="height"
     value="400" />
```

Description

Specifies the height of the generated PopChart image.

This value should be set to an integer, which represents the height, in pixels, of the generated image.

This attribute sets the height of the image that is returned to the browser, but not necessarily the height at which the browser will display the image. This is because the `htmlHeight` attribute, if it is set, overrides the `height` attribute within the browser.

For more information on how PopChart Server handles the sizing of an image, refer to "Changing the Image Format" on page 4-13 of the *PopChart Server User Guide*.

String htmlHeight

Syntax
```
myPopChart.htmlHeight = "75%";
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="htmlHeight" value="75%" />
```

Description

Specifies the height at which a client will display a PopChart image.

This value should be set to a string which contains either the height of the image in pixels (number only), or the height of the image as a percentage of the web page (number followed by percentage % sign).

This attribute does not set the actual height of the image (the actual height is determined by either the appearance file or the PopChart Embedder `height` attribute). Rather, the `htmlHeight` attribute sets the value of the `height` attribute within the `<object>`, `<embed>`, or `<img>` tag that embeds the PopChart image into the web page. In other word, it is essentially like saying `<img height="75%">` in HTML.

If `htmlHeight` is not set, the value of `height` is used instead.

For more information on how PopChart Server handles the sizing of an image, refer to "Changing the Image Format" on page 4-13 of the *PopChart Server User Guide*.

String htmlWidth

Syntax
```
myPopChart.htmlWidth = "100%";
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="htmlHeight" value="100%" />
```

Description

Specifies the width at which a client will display a PopChart image.

This value should be set to a string which contains either the width of the image in pixels (number only), or the width of the image as a percentage of the web page (number followed by percentage % sign).

This attribute does not set the actual width of the image (the actual width is determined by either the appearance file or the PopChart Embedder `width` attribute). Rather, the `htmlWidth` attribute sets the value of the `width` attribute within the `<object>`, `<embed>`, or `<img>` tag that embeds the PopChart image into the web page. In other word, it is essentially like saying `<img width="100%">` in HTML.

If `htmlHeight` is not set, the value of `width` is used instead.

For more information on how PopChart Server handles the sizing of an image, refer to "Changing the Image Format" on page 4-13 of the *PopChart Server User Guide*.

String imageType

Syntax
```
myPopChart.imageType = { "EPS" | "GIF" | "FLASH" |
     "PDF" | "PNG" | "SVG" | "WBMP" };
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="imageType"
     value="{eps|gif|flash|pdf|png|svg|wbmp}" />
```

Description

Specifies the format for the generated PopChart image.

The value of this string should be a string containing the name of one of PopChart Server's supported image formats, such as `"GIF"`, `"FLASH"`, or `"SVG"`. A complete list of these formats is found under the topic "Image Types" on page 4-6.

You only need to set this attribute if you are requesting an image format other than the default format, which can be specified on the `Settings > Image Type` page of the Administration console, or using the `type` configuration setting.

This format is not necessarily the same format that will be returned to the client. If you have enabled best-image fallback (refer to "Best Image Fallback" on page 4-13 of the *PopChart Server User Guide*) or automatic PNG detection (see "Automatic PNG Detection" on page 3-27 of the *PopChart Server User Guide*) PopChart Server may return a different format, depending on which formats a browser can display.

For an explanation of the image formats that PopChart Server supports, refer to Chapter 14, "Image Formats."

String internalCommPortAddress

Syntax
```
myPopChart.internalCommPortAddress =
     "10.0.0.1:2002";
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="internalCommPortAddress"
     value="10.0.0.1:2002" />
```

Description

Specifies the internal address and comm port for your PopChart Server.

The value of this string should be an IP address or domain name, followed by your comm port number (e.g. *10.0.0.1:2002*, *imageserver.myintranet.com:2002*). By default, your comm port is *2002*.

This is the address that your web application will use to communicate with PopChart Server. The address needs only be accessible from the web application server. Often times, PopChart Server will run on the same machine as your web application server. In such cases, you can set `internalCommPortAddress` to *localhost:2002*.

The internal comm port address is different from the external server address, which is the address that clients will use to communicate with PopChart Server (refer to the `externalServerAddress` attribute). Both the external and internal addresses must be specified when generating a PopChart image (except when using the JavaScript embedder, which does not use a comm port, or when setting the `clusterMonitorAddress` while clustering).

For more information about configuring PopChart Server ports and addresses, refer to "Setting Server Address and Ports" on page 3-18 of the *PopChart Server User Guide*.

String language

Syntax
```
myPopChart.language = PopChartEmbedder.EN;
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="language" value="EN" />
```

Description

Specifies language settings for the generated PopChart image and descriptive text.

At this point in time, the only valid setting for this is *EN* (English). In the future, additional language settings will be supported.

boolean makeFullRequest

New in version 4.0.5. Not available in the JavaScript PopChart Embedder.

Syntax
```
myPopChart.makeFullRequest = true|false;
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="makeFullRequest" value="true|false"
     />
```

Description

Specifies whether the PopChart Embedder should embed an image within a web page using its full URL request, or just its embedder session ID. The default value is *false*.

You will probably only want to set this attribute when you want to disable the embedder session cache. For more information about this attribute, refer to "Embedder Session Cache" on page A-5 of the *PopChart Server User Guide*.

int maxRequestLength

New in version 4.0.5. Not available in the JavaScript PopChart Embedder.

Syntax
```
myPopChart.makeRequestLength = int;
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="makeRequestLength" value="int" />
```

Description

Specifies the maximum length for full URL requests embedded by PopChart Embedder within a web page. If an image's full URL request is longer than this length, PopChart Embedder will embed it using an embedder session ID instead of a full request. If this value is set to 0 (the default setting), there will be no maximum length.

You will probably only want to set this attribute when you want to disable the embedder session cache. For more information about this attribute, refer to "Embedder Session Cache" on page A-5 of the *PopChart Server User Guide*.

Note: *This attribute has no effect unless you set `makeFullRequest` to true.*

String password

Syntax
```
myPopChart.password = "password";
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="password" value="password" />
```

Description

Specifies the password for your PopChart Server.

You must use a password when using any of the following methods or attributes: `saveImageToAppServer(String, String)`, `saveImageToPopChartServer(String)`, `useLogData`.

Warning: You should not do anything that requires the use of your password with the JavaScript PopChart Embedder. Users will have access to your source files and be able read the password straight from the source.

String pcScript

Syntax
```
myPopChart.PCScript = pcscript;
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="PCScript" value="pcscript" />
```

Description

Specifies the PCScript command string.

PCScript (PopChart Script) is a language used to send data and graph formatting options to PopChart Server. Example 4.3 shows how you can give PCScript commands with the PCScript attribute.

To learn more about PCScript commands and the PCScript command string, refer to Chapter 5, "PCScript."

Example 4.3 Setting PCScript

```
String pcscript = "title.setText(2001 Statistics)";
pcscript += "graph.setSeries(January; 45,11,45,16; 90,13,34,56)";
pcscript += "graph.Description(Sales Stats)";

myPopChart.PCscript = pcscript;
```

boolean returnDescriptiveLink

Syntax
```
myPopChart.returnDescriptiveLink = true;
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="returnDescriptiveLink" value="true"
     />
```

Description

If set to true, this attribute instructs PopChart Server to return a link to a text description of the PopChart in addition to the image.

The text description link will be a blue underlined letter d. It will appear at the bottom right hand corner of the image. When a user clicks on it, the browser will jump to a separate page containing a description of the chart for the visually impaired.

PopChart Server will not generate descriptive text unless you set this attribute to *true*.

For more information about descriptive text, refer to Chapter 8, "Descriptive Text Settings," in the *PopChart Server User Guide*.

String svgTemplate

Syntax
```
myPopChart.svgTemplate = "svg_templates/grow.svg";
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="svgTemplate"
     value="svg_templates/grow.svg" />
```

Description

Specifies the location of an SVG template that should be applied to your PopChart image. This location is relative to your document root (usually chart_root). It can also be an absolute path or URL.

PopChart Server allows you to insert dynamically generated SVG PopChart images into an SVG document. You can use a tool such as Adobe® Illustrator® to create an SVG file, which you will use as a template. Then, simply insert a `<!--popchart-->` tag where you want PopChart Server to insert the dynamically generated SVG.

Several popular templates are located in the chart_root/svg_templates directory. They include special effects like fading, rotation, and animation. For consistency, we recommend that you also put any other SVG templates in this directory.

Important: *Make sure that your path.xml file gives PopChart Server permission to read from the location where your SVG Template is located. Otherwise, PopChart Server will not be able to read the file (see "Setting Path Permissions" on page 3-37 of the PopChart Server User Guide).*

boolean useCache

Syntax
```
myPopChart.useCache = false;
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="useCache" value="false" />
```

Description

Specifies whether or not the generated PopChart image should be cached.

This command has two effects. First of all, it tells the client's browser whether or not it should cache the image. Second of all, if you are running PopChart Server Enterprise with caching enabled, it tells PopChart Server whether or not it should cache the image.

By default, this value is set to *true*.

For more information about caching, refer to "Caching" on page A-3 of the *PopChart Server User Guide*.

boolean useLogData

Syntax
```
myPopChart.useLogData = true;
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="useLogData" value="true" />
```

Description

Instructs PopChart Server to import its data from the PopChart Server log files.

This command is only useful when you are trying to generate your own customized usage report for PopChart Server.

String userAgent

Not used in JavaScript PopChart Embedder.

Syntax
```
myPopChart.userAgent = userAgentString;
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="userAgent" value="<%=
     request.getHeader(\"user-agent\") %>" />
```

Description

Tells PopChart Server what the client's user agent is.

You should set this attribute to the value of the user agent string that is passed to your web application server (e.g. *"Mozilla/4.0 (compatible; MSIE 4.01; Windows NT)"* or *"Mozilla/4.7 (Macintosh; I; PPC)"*). Each environment has its own method for looking up the user agent string. The method for looking up the string in Java is shown in Example 4.4, while the method in JavaBeans is shown in the JavaBeans syntax description above.

Note: *It is not necessary to set this attribute, but doing so helps PopChart Server optimize its code for embedding PopChart Images.*

Example 4.4 Specifying the User Agent in Java

```
myPopChart.userAgent = context().request().headerForKey("user-
             agent");
```

int width

Syntax
```
myPopChart.width = 600;
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart" property="width"
     value="600" />
```

Description

Specifies the width of the generated PopChart image.

This value should be set to an integer, which represents the width, in pixels, of the generated image.

This attribute sets the height of the image that is returned to the browser, but not necessarily the height at which the browser will display the image. This is because the `htmlWidth` attribute overrides the `width` attribute within the browser.

For more information on how PopChart Server handles the sizing of an image, refer to "Changing the Image Format" on page 4-13 of the *PopChart Server User Guide*.

PopChart Embedder METHODS

This section describes each of the methods of PopChart Embedder. These methods perform a variety of tasks, such as loading data files or generating embedding HTML for your PopChart image.

Note: *The code in the syntax section of these descriptions assumes that you are manipulating a PopChart Embedder object named* `myPopChart`.

Note: *Where there is not a separate JavaBean syntax for PopChart Embedder methods, the syntax is the same as the Java syntax, except embedded inside of* `<% ... %>`. *This means that there is no native PopChart Embedder JavaBean property for that method.*

void addHTMLTable(String,String)

Available only in PopChart Server Enterprise.

Syntax

```
myPopChart.addHTMLTable([graph [,title]]);
```

Description

Instructs PopChart Server to return a HTML table containing the data for the specified graph. The HTML table will appear below the PopChart image.

You can call this method more than once. The HTML tables will appear below the image in the order that they are added. Or, if you call this method without any parameters, PopChart Server will add an HTML table for all graph objects in the PopChart.

For more information about HTML Tables, refer to "HTML Data Tables" on page 7-29 of the *PopChart Server User Guide*.

Parameters

The `addHTMLTable()` method accepts the following optional parameters.

graph *String*
The name of the graph object for which you want to return an HTML Table. If the specified graph object does not exist, the method will be ignored.

This parameter is optional. If it is not included, PopChart Server will return an HTML Table for all graphs.

title *String*
The title for the HTML Table. This parameter is used as the value for the `title` attribute of the `<table>` tag.

This parameter is optional (except with the COM PopChart Embedder). If it is not included, PopChart Server will try to give your table a title for you by looking for a textbox named *title*. If it cannot find one, your table will not have a title.

Example 4.5 **Adding an HTML Table**

```
myPopChart.addHTMLTable("graph","PopChart Data");
```

Important: *Due to limitations in the Component Object Model,* `addHTMLTable()` *always requires two arguments in the COM PopChart Embedder, even if you have no need to set the last parameter. If you don't need to set the title and you are using the COM PopChart Embedder, set the last argument to an empty string (e.g* `myPopChart.addHTMLTable "graph", ""`*).*

void addObjectParamTag(String, String)

New in PopChart Server 4.0.3. This method is not available in the JavaBean PopChart Embedder.

Syntax

```
myPopChart.addObjectParamTag(name, value)
```

Description

Adds a `<param>` tag to PopChart images that have been embedded using an `<object>` tag. Currently, this includes FLASH and PDF images in Internet Explorer only. This allows you to extend the functionality of the plug-ins that are used to display your PopChart images.

One common use for this method is to set the display mode of a FLASH image to transparent, so that dynamic HTML elements (such as menus) will overlap the image. The FLASH plug-in allows you to do this by adding the following `<param>` tag to the `<object>` element.

```
<param name="WMODE" value="transparent" />
```

To have the PopChart Embedder add this <param> tag to your FLASH PopChart image, you would make the following method call.

```
myPopChart.addObjectParamTag("WMODE","transparent")
```

Parameters

The `addObjectParamTag()` method accepts the following parameters.

name *String*
The value of the `name` attribute for the `<param>` tag that you are creating.

value *String*
The value of the `value` attribute for the `<param>` tag that you are creating.

void addPCXML(String)

Syntax

```
myPopChart.addPCXML(XMLString);
```

JavaBean Syntax

```
<JSP:setProperty name="myPopChart"
     property="addPCXML" value="XMLString" />
```

Description

Sends the specified PopChart XML statement to PopChart Server.

Note: *Any PopChart XML added through this method will be appended after all of the `loadPCXML(String)` methods have been processed. PCScript will be processed after all PopChart XML has been processed.*

For more information about PopChart XML, refer to Chapter 10, "Using PopChart XML," in the *PopChart Server User Guide*.

Parameters

The `addPCXML()` method accepts the following parameters.

XMLString *String*
The PopChart XML that should be sent to PopChart Server.

Example 4.6 Streaming PopChart XML to PopChart Server

```
myPopChart.addPCXML("<Chart> <GraphData Name='graph'> <Categories>
            <Category Name='Arrivals'/> <Category
            Name='Departures'/> <Category Name='Unused'/>
            <Category Name='Out of Commision'/> </Categories>
            <Series Name='Atlanta'> <Data Value='23.0'/> <Data
            Value='36.0'/> <Data Value='11.0'/> <Data
            Value='7.0'/> </Series><Series Name='Boston'> <Data
            Value='41.0'/> <Data Value='17.0'/><Data
            Value='25.0'/> <Data Value='9.0'/> </Series>
            </GraphData> </Chart>");
```

String getEmbeddingHTML()

Syntax

```
myPopChart.getEmbeddingHTML();
```

JavaBean Syntax

```
<JSP:getProperty name="myPopChart"
     property="embeddingHTML" />
```

Description

Returns a String containing the HTML necessary to embed the PopChart image.

This is one of PopChart Embedder's most important methods. If you are embedding your PopChart image on a web page, this will always be the last method you call, as it essentially "compiles" your PopChart image.

This function returns a String, but does not write it to the web page. That task is left to you, and how you do it will depend on your web environment. In the JavaScript Embedder, for instance, you could write this String using the `document.write()` command. Example 4.7 shows how to write the string to an Active Server Page.

To locate information on embedding a PopChart image in a particular environment, refer to *Table 4.1*

Example 4.7 **Writing the Embedding HTML in an Active Server Page**

```
Response.Write myPopChart.getEmbeddingHTML()
```

byte[] getImageData()

Not available in JavaScript or PHP Embedders.

Syntax

```
myPopChart.getImageData();
```

Description

Returns a byte array representation of the PopChart image, suitable for saving or for returning to a client.

To save this image in Java, you would create a new `DataOutputStream`, and write the results of this method using the `writebytes` method, as shown in Example 4.8.

Example 4.8 **Saving the Results of `getImageData()`**

```
DataOutputStream saveImage = new DataOutputStream(filename);
saveImage.writeBytes(myPopChart.getImageData());
saveImage.close();
```

You can also use this method in a Java Servlet to return an image to a browser. Be sure to tell the browser the length of the image and the image's mime type. "Table of MIME Types" on page 14-18 lists the mime types for each of PopChart Server's image formats.

Example 4.9 shows how to return an image using `getImageData()` in a Java servlet.

Example 4.9 **Returning the Results of `getImageData()` to a Browser**

```
response.setContentType("application/x-shockwave-flash");
response.setHeader("Expires", "-1");
response.setHeader("nPragma", "no-cache");
BufferedOutputStream returnImage = new
                BufferedOutputStream(response.getOutputStream());
byte[] imageData = myPopChart.getImageData();
response.setContentLength(imageData.length);
returnImage.write(imageData);
returnImage.flush();
returnImage.close();
```

String getPCEmbedderVersion()

Syntax
```
myPopChart.getPCEmbedderVersion();
```

Description

Returns the version of the PopChart Embedder that you are using.

This method is only useful when you are debugging and need to know what version you are using.

void loadCommandFile(String)

Syntax
```
myPopChart.loadCommandFile(commandfile);
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="loadCommandFile" value="commandfile"
     />
```

Description

Instructs PopChart Server to load server commands from a server command file.

Commands in a server command file are processed after all other server commands, meaning they will override commands specified outside of the server command file.

For more information on server commands and server command files, refer to Chapter 6, "Server Commands."

Parameters

The `loadCommandFile()` method accepts the following parameters.

commandfile *Filename | URL*

The path to and location of the server command file that you want PopChart Server to use. This file is relative to PopChart Server's document root. You can also specify an absolute path or URL.

Make sure that the file is accessible from the machine that PopChart Server runs on. Also make sure that your path.xml file permits PopChart Server to read from this path (see "Setting Path Permissions" on page 3-37 of the *PopChart Server User Guide*).

void loadData(String,String,String,String)

Syntax
```
myPopChart.loadData(graphname, datasrc [,method
     [,HTMLTable]]);
```

Description

Instructs PopChart Server to load data into the specified graph object from one of the following types of data sources: tab delimited data file, comma separated value file, xml database file, or a table in an HTML file.

By default, this command replaces all existing data in the graph. However, if you want to append the data from the file to the existing data, you can set the method attribute to *append*.

PopChart Server will automatically determine the data format. For more information on loading data and on these data formats, refer to "Importing Data from Data Files" on page 6-16 of the *PopChart Server User Guide*.

Important: *Your path.xml file must grant PopChart Server permission to read files from the location that you specify. Otherwise, it will be unable to load the data. For more information, see "Setting Path Permissions" on page 3-37 of the PopChart Server User Guide.*

Note: *This command provides the same functionality as the `LoadFile` PCScript command.*

Parameters

The `loadData()` method accepts the following parameters.

graphname *String*

The name of the graph object into which you want to load the data. If the specified graph object does not exist, the method will be ignored.

datasrc *Filename | URL*

The path to and location of the server command file that you want PopChart Server to use. This file is relative to PopChart Server's document root. You can also specify an absolute path or URL.

Make sure that the file is accessible from the machine that PopChart Server runs on. Also make sure that your path.xml file permits PopChart Server to read from this path (see "Setting Path Permissions" on page 3-37 of the *PopChart Server User Guide*).

method *"append" | "replace"*

Indicates the method for importing the data. If `method` is set to *replace*, all data currently in the graph will be replaced. If it is set to *append*, the imported data will be appended to any data already in the graph, including data from the appearance file.

If this parameter is not set, it is assumed to be *replace*. It must always be set when importing data from HTML tables. It must always be set when you use the COM PopChart Embedder.

HTMLTable *String*

The title or number of the table that should be loaded from the web page. If this parameter is a title, be sure that the title corresponds to the value of the `title` attribute of your `<table>` tag. If it is a number, PopChart Server will import data from the *n*th table in the web page, where *n* is the number you specify.

Example 4.11 illustrates how this parameter is used. The first line of code imports data from the table titled *pricing* on the web page stats.html. The second line of code imports data from the 11th table on the web page morestats.html.

This parameter must always be set when you import data from HTML tables. It also must always be set when you use the COM PopChart Embedder, but you should send an empty string if you are not importing from an HTML Table.

Important: *Due to limitations in the Component Object Model, `loadData()` always requires four arguments in the COM PopChart Embedder, even if you have no need to set the last two parameters. If you don't need to set the last two parameter and you are using the COM PopChart Embedder, set the third argument to "replace" and the fourth argument to an empty string (e.g `loadData("graph","prices.csv","replace","");`).*

Example 4.10 **Loading a Comma Separated Value File**

```
myPopChart.loadData("graph","data/020115.csv");
```

Example 4.11 **Loading an HTML Table from a Web Page**

```
myPopChart.loadData("graph", "http://www.myserver.com/stats.html",
                "replace", "pricing")
myPopChart.loadData("graph", "\\database\morestats.html",
                "append", "11")
```

void loadServerSideImage(String)

Syntax

```
myPopChart.loadServerSideImage(imagefile)
```

JavaBean Syntax

```
<JSP:setProperty name="myPopChart"
     property="loadServerSideImage"
     value="imagefile" />
```

Description

Instructs PopChart Server to load a server-side image.

When you use this command, PopChart Server will not generate a graph, even if graph-generating attributes or methods have been used. Instead, it will simply return the specified server-side image.

Important: *It is essential that you set the `imageType` attribute to the format of the image that you are loading; otherwise, PopChart Server may return the wrong mime type, causing the browser to be unable to display the image.*

This method is most often used in conjuction with the `saveImageToPopChartServer(String)` method.

Parameters

The `loadServerSideImage()` method accepts the following parameters.

imagefile *Filename*

The path to and location of the image that you want PopChart Server to load. This file is relative to PopChart Server's document root. You can also specify an absolute path.

Make sure that the file is accessible from the machine that PopChart Server runs on. Also make sure that your path.xml file permits PopChart Server to read from this path (see "Setting Path Permissions" on page 3-37 of the *PopChart Server User Guide*).

void loadPCXML(String)

Syntax
```
myPopChart.loadPCXML(xmlsrc)
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="loadPCXML" value="xmlsrc" />
```

Description

Instructs PopChart Server to load a PopChart XML file. The contents of this file will be appended to any other PopChart XML files that have been loaded into the graph, including the appearance file.

Note: *Any PopChart XML added through the `addPCXML(String)` method will be appended after all of the `loadPCXML()` methods have been processed.*

For more information about PopChart XML, refer to Chapter 10, "Using PopChart XML," in the *PopChart Server User Guide*.

Parameters

The `loadPCXML()` method accepts the following parameters.

xmlsrc *Filename | URL*

The path to and location of the PopChart XML file that you want PopChart Server to load. This file is relative to PopChart Server's document root. You can also specify an absolute path or a URL.

Make sure that the file is accessible from the machine that PopChart Server runs on. Also make sure that your path.xml file permits PopChart Server to read from this path (see "Setting Path Permissions" on page 3-37 of the *PopChart Server User Guide*).

void resetPCXML()

Syntax
```
myPopChart.resetPCXML()
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="resetPCXML" value="true" />
```

Description

Resets any PopChart XML that has been sent to PopChart Server. All PopChart XML commands and files previously sent to PopChart Server will be ignored.

For more information about PopChart XML, refer to Chapter 10, "Using PopChart XML," in the *PopChart Server User Guide*.

Parameters

The `resetPCXML()` method accepts no parameters.

void saveImageToAppServer(String, String)

Not implemented in JavaScript or PHP Embedders.

Syntax
```
myPopChart.saveImageToAppServer(path, name)
```

Description

Saves the generated PopChart image to the specified path and location on your Web Application Server.

Important: *A password is required (using the `password` attribute) to save images.*

Note: *If you want to save the file to PopChart Server so that you can load it later, you should use the `saveImageToPopChartServer(String)` method.*

Parameters The `saveImageToAppServer()` method accepts the following parameters.

path *Filepath*
The path to the local directory where you want to save your file. You should specify an absolute path.

name *Filename*
The file name for your saved image. PopChart Server will automatically write over that file if it already exists.

void saveImageToPopChartServer(String)

Syntax
```
myPopChart.saveImageToPopChartServer(name)
```

JavaBean Syntax
```
<JSP:setProperty name="myPopChart"
     property="saveImageToPopChartServer"
     value="name" />
```

Description Saves the generated PopChart image to the specified location on PopChart Server.

Important: *A password is required (using the `password` attribute) to save images. Also, you must have permission to save the file to the location that you save it to (see "Setting Path Permissions" on page 3-37 of the PopChart Server User Guide.).*

Saving the image to PopChart Server allows you to load it later using the *`loadServerSideImage(String)`* method. Under certain circumstances, this strategy can increase your server's capacity. For example, if you know that a particular PopChart image will be viewed often, you can save it to PopChart Server the first time that someone requests it, and then use the *`loadServerSideImage(String)`* to load it each subsequent time that someone requests it. The result is that PopChart Server generates only one image, no matter how often it is requested. Since image generation is the most demanding task for PopChart Server, this tremendously decreases the server load.

Note: *If you want to save the file to your web application server, you should use the `saveImageToAppServer(String, String)` method.*

Parameters The `saveImageToPopChartServer()` method accepts the following parameters.

name *Filename*
The path to and location that you want to save your image to. This path will be relative to PopChart Server's document root. Make sure that your path.xml file permits PopChart Server to read from this path (see "Setting Path Permissions" on page 3-37 of the *PopChart Server User Guide*). PopChart Server will automatically write over that file if it already exists.

void setData(String,String)

Syntax

```
myPopChart.setData(graphname, data);
```

Instructs PopChart Server to set the data for the specified graph object. This data must be in the standard XML format (refer to "XML Data Files" on page 6-17 of the *PopChart Server User Guide*).

Parameters

The `setData()` method accepts the following parameters.

graphname *String*
The name of the graph object for which you want to set the data. If the specified graph object does not exist, the method will be ignored.

data *String*
The actual data that should be sent to the graph. This data can be in one of the following formats: tab delimited, comma separated, xml, or HTML table (refer to "Data Formats that PopChart Server Imports" on page 6-16 of the *PopChart Server User Guide*).

void setDBQuery(String, String, String, String, String, String)

Available only for Java and COM versions of PopChart Embedder. This method is only implemented in COM for PopChart Embedder versions 4.0.1 and higher.

Syntax

```
myPopChart.setDBQuery(graphname, driver, database,
     user, password, query);
```

Description

Instructs the PopChart Embedder to make an SQL query to the specified database. The resulting data will be loaded into the specified graph, and will replace any existing data.

You can connect to any type of database, as long as that database has a JDBC driver (for Java), or has been set up as an ODBC data source on the computer running PopChart Server (for COM and C++).

Note: *For JDBC data sources, be sure that the driver is included in the classpath or header for your application.*

Example 4.12 illustrates how to call this method in Java. For more information about using databases with PopChart Embedder, refer to "Importing Data Directly from Databases" on page 6-25 of the *PopChart Server User Guide*.

Parameters

The `setDBQuery()` method accepts the following parameters.

graphname *String*
The name of the graph object into which you want to import the data. Usually, this is *graph*, but check the appearance file to be sure.

driver *String*
In Java, this is the JDBC driver name. Make sure this driver has been included in the classpath for your web application.

For COM, this parameter is not used and should be set to an empty String.

database *String*
In Java, this is the database URL (e.g. *jdbc:obdc:bcdemo*).

For COM, this is the DSN name of the ODBC database that you want to connect to. For DSN-less connections, you should instead set this to a string containing the information necessary to identify your database (e.g. *"Provider=SQLOLEDB.1;Password=MyPassword;Persist Security Info=True;User ID=MyUserID;Initial Catalog=Northwind;Data Source=MyDataServer"*).

user *String*
A valid user name for that database. If no user name is required, you should set this parameter to an empty String.

password *String*
The user's password. If no password is required, you should set this parameter to an empty String.

query *String*
The SQL query that you wish to make.

Example 4.12 Importing Data From a DataBase

```
myPopChart.setDBQuery("graph", "sun.jdbc.odbc.JdbcOdbcDriver",
                "jdbc:odbc:bcdemo", "popchart", "secretpassword",
                "Select Description, OnHand from parts order by
                Description");
```

void setResultSet(String,ResultSet)

Applies only to the Java PopChart Embedder.

Syntax
```
myPopChart.setResultSet(graphname, resultSet);
```

Description
Imports the data from a `ResultSet` object into the specified graph object. The new data will replace any existing data.

`ResultSet` is a class in the `java.sql` package. ResultSet objects are usually returned when you make an SQL query. See Example 4.13 for an example of how this works.

For more information about using databases with PopChart Server, refer to "Importing Data Directly from Databases" on page 6-25 of the *PopChart Server User Guide*.

Parameters
The `setResultSet()` method accepts the following parameters.

graphname *String*
The name of the graph object into which you want to import the data. Usually, this is *graph*, but check the appearance file to be sure.

resultSet A resultSet object.		*ResultSet*

Example 4.13 **Using `setResultSet()`**

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
theConnection = DriverManager.getConnection("jdbc:odbc:bcdemos","po
                pchart","secretpassword");
theStatement = theConnection.createStatement();
ResultSet results = theStatement.executeQuery("Select Description,
                OnHand from parts order by Description");
myPopChart.setResultSet("graph", results);
```

Note: *The example above accomplishes the same task that* Example 4.12 *accomplished.*

DEPRECATED METHODS

In versions of the PopChart Image Server prior to the 4.0 release, the PopChart Embedder was called the PCISEmbedder. To maintain backwards compatibility, you can still use the `PCISEmbedder` class, which is included in the PopChartEmbedder.jar or PCEmbedder.dll file. However, this class, as well as many methods which it used to contain, have been deprecated. Code written for the PCISEmbedder prior to the 4.0 release will still work in the 4.0 release, but the `PCISEmbedder` class and its deprecated methods will be discontinued in the 5.0 release.

Note: *Deprecated methods are not included in the `PopChartEmbedder` class.*

This section contains three tables, one for each of the 3.x versions of the PCISEmbedder. These tables list the deprecated methods and show their 4.0 equivalent. You can click on the 4.0 methods to jump to a description of the method.

DEPRECATED JAVA METHODS

TABLE 4.2 Deprecated Java Methods

Deprecated Method	4.0 Equivalent
`retrieveImageServerSide(String)`	`loadServerSideImage(String)`
`saveImageAsLocally(String, String)`	`saveImageToAppServer(String, String)`
`saveImageAsServerSide(String)`	`saveImageToPopChartServer(String)`
`serverInfoAppendSlash(boolean)`	`appendServerInfoSlash`
`setAppearanceFile`	`appearanceFile`
`setBGColorStr`	`bgColor`
`setCacheImage(boolean)`	`useCache`
`setExtraHTMLAttributes(String)`	`extraHTMLAttributes`
`setExtraPCISCommands(String)`	`extraPCSCommands`
`setHeight(int)`	`height`
`setHTMLHeight(String)`	`htmlHeight`
`setHTMLWidth(String)`	`htmlWidth`
`setReturnTextDescription(String)`	`returnDescriptiveLink = true; language = PopChartEmbedder.EN;`[a]
`setOutputToGIF()`	`imageType = "GIF";`
`setOutputToFLASH()`	`imageType = "FLASH";`
`setOutputToFLASHWithFallback(boolean)`	`imageType = "FLASH"; fallback = { "STRICT"|"LOOSE" }`[b]
`setOutputToPNG()`	`imageType = "PNG";`
`setOutputToSVG()`	`imageType = "SVG";`
`setOutputToSVGWithFallback(boolean)`	`imageType = "FLASH"; fallback = { "STRICT"|"LOOSE" }`[c]
`setOutputToWBMP()`	`imageType = "WBMP";`
`setPCScript(String)`	`pcScript`
`setReturningGraphicalLog(String)`	`useLogData = true; password = "password";`
`setServerInfo(String)`	`externalServerAddress = "www.myserver.com:2001"; internalCommPortAddress = "10.0.0.1:2002";`[d]
`setUseCommPortAlways()`	Obsolete (Comm Port always used)

TABLE 4.2 Deprecated Java Methods

Deprecated Method	4.0 Equivalent
`setUseCommPortIfOverLength(int)`	`makeFullRequest = true;`[e] `maxRequestLen = int;`
`setUseCommPortNever()`	`makeFullRequest = true;`[f]
`setUseLoadBalancing(boolean)`	Obsolete (use Clustering instead)
`setUserAgent(String)`	`userAgent`
`setUseSVGTemplate(String)`	`svgTemplate`
`setWidth(int)`	`width`

a. You may not need to set the `language` attribute if you have already specified a default language. PopChart Embedder will assume a default language of *EN* (English).

b. Whether you use `"STRICT"` or `"LOOSE"` depends on the fallback type.

c. Whether you use `"STRICT"` or `"LOOSE"` depends on the fallback type.

d. For a better understanding of how this method translates, you should examine the description of the new attributes.

e. The PopChart Embedder must still use the COM port to generate the embedding HTML, however it will embed a full URL request instead of an embedder session ID.

f. The PopChart Embedder must still use the COM port to generate the embedding HTML, however it will embed a full URL request instead of an embedder session ID.

DEPRECATED COM METHODS & ATTRIBUTES

TABLE 4.3 Deprecated COMObject Methods & Attributes

Deprecated Method	4.0 Equivalent
AlwaysUseCommPort	Obsolete (Comm Port always used)
AppearanceFile	appearanceFile
BGColorString	bgColor
CacheImage	useCache
GetEmbeddingHTML()	getEmbeddingHTML()
Height	height
ImageData	getImageData()
NeverUseCommPort	makeFullRequest = true;[a]
RetrieveImageServerSide(String)	loadServerSideImage(String)
SaveImageAsLocally(String, String)	saveImageToAppServer(String, String)
SaveImageAsServerSide(String)	saveImageToPopChartServer(String)
ServerInfoAppendSlash(boolean)	appendServerInfoSlash
SetExtraHTMLAttributes(String)	extraHTMLAttributes
SetExtraPCISCommands(String)	extraPCSCommands
SetReturnTextDescription(String)	returnDescriptiveLink = true language = "EN";[b]
SetOutputToGIF()	imageType = "GIF"
SetOutputToFLASH()	imageType = "FLASH"
SetOutputToFLASHWithFallback(boolean)	imageType = "FLASH" fallback = { "STRICT" | "LOOSE" }[c]
SetOutputToPNG()	imageType = "PNG";
SetOutputToSVG()	imageType = "SVG"
SetOutputToSVGWithFallback(boolean)	imageType = "FLASH"; fallback = { "STRICT" | "LOOSE" }[d]
SetOutputToWBMP()	imageType = "WBMP"
SetReturningGraphicalLog(String)	useLogData = true password = "password"
SetServerInfo(String)	externalServerAddress = "www.myserver.com:2001" internalCommPortAddress = "10.0.0.1:2002";[e]

TABLE 4.3 Deprecated COMObject Methods & Attributes

Deprecated Method	4.0 Equivalent
`SetUseCommPortIfOverLength(int)`	`makeFullRequest = true;`[f] `maxRequestLen = int;`
`SetUserAgent(String)`	`userAgent`
`SetUseSVGTemplateFile(String)`	`svgTemplate`
`UseLoadBalancing`	Obsolete (use Clustering instead)
`Width`	`width`

a. The PopChart Embedder must still use the COM port to generate the embedding HTML, however it will embed a full URL request instead of an embedder session ID.

b. You may not need to set the `language` attribute if you have already specified a default language. PopChart Embedder will assume a default language of *EN* (English).

c. Whether you use `"STRICT"` or `"LOOSE"` depends on the fallback type.

d. Whether you use `"STRICT"` or `"LOOSE"` depends on the fallback type.

e. For a better understanding of how this method translates, you should examine the description of the new attributes.

f. The PopChart Embedder must still use the COM port to generate the embedding HTML, however it will embed a full URL request instead of an embedder session ID.

DEPERECATED JAVABEAN PROPERTIES AND METHODS

TABLE 4.4 Deprecated JavaBean Properties & Methods

Deprecated Method	4.0 Equivalent
`bgColorStr`	`bgColor`
`cacheImage`	`useCache`
`outputToGIF()`	`<jsp:setProperty name="myPopChart" property="imageType" value="GIF"/>`
`outputToFLASH()`	`<jsp:setProperty name="myPopChart" property="imageType" value="FLASH"/>`
`outputToFLASHWithFallback(boolean)`	`<jsp:setProperty name="myPopChart" property="imageType" value="FLASH"/>` `<jsp:setProperty name="myPopChart" property="fallback" value="STRICT|LOOSE"/>`[a]

TABLE 4.4 Deprecated JavaBean Properties & Methods

Deprecated Method	4.0 Equivalent
`outputToPNG()`	`<jsp:setProperty name="myPopChart" property="imageType" value="PNG"/>`
`outputToSVG()`	`<jsp:setProperty name="myPopChart" property="imageType" value="SVG"/>`
`outputToSVGWithFallback(boolean)`	`<jsp:setProperty name="myPopChart" property="imageType" value="SVG"/>` `<jsp:setProperty name="myPopChart" property="fallback" value="STRICT|LOOSE"/>`[b]
`outputToWBMP()`	`imageType = PopChartEmbedder.WBMP;`
`pcScript`	`PCScript`
`pcisEmbedderVersion`	`getPCEmbedderVersion()`
`retrieveImageServerSide`	`<jsp:setProperty name="myPopChart" property="loadServerSideImage" value="filename"/>`
`returnTextDescription`	`<jsp:setProperty name="myPopChart" property="returnDescriptiveLink" value="true"/>` `<jsp:setProperty name="myPopChart" property="language" value="EN"/>`[c]
`returningGraphicalLog`	`<jsp:setProperty name="myPopChart" property="useLogData" value="true"/>` `<jsp:setProperty name="myPopChart" property="password" value="password"/>`
`saveImageAsLocally(String, String)`	`saveImageToAppServer(String, String)`
`saveImageAsServerSide`	`saveImageToPopChartServer(String)`

TABLE 4.4 Deprecated JavaBean Properties & Methods

Deprecated Method	4.0 Equivalent
`serverInfo`	`<jsp:setProperty name="myPopChart" property="externalServerAddress" value="www.myserver.com:2001"/>` `<jsp:setProperty name="myPopChart" property="internalCommPortAddress" value="10.0.0.1:2002"/>`[d]
`serverInfoAppendSlash`	`appendServerInfoSlash`
`useCommPortAlways()`	Obsolete (Comm Port always used)
`useCommPortIfOverLength(int)`	`<jsp:setProperty name="myPopChart" property="makeFullRequest" value="true"/>`[e] `<jsp:setProperty name="myPopChart" property="maxRequestLen" value="int";`
`useCommPortNever()`	`<jsp:setProperty name="myPopChart" property="makeFullRequest" value="true"/>`[f]
`useLoadBalancing`	Obsolete (use Clustering instead)
`useSVGTemplate`	`svgTemplate`

a. You may not need to set the `language` attribute if you have already specified a default language. PopChart Embedder will assume a default language of *EN* (English).

b. You may not need to set the `language` attribute if you have already specified a default language. PopChart Embedder will assume a default language of *EN* (English).

c. You may not need to set the `language` attribute if you have already specified a default language. PopChart Embedder will assume a default language of *EN* (English).

d. For a better understanding of how this method translates, you should examine the description of the new attributes.

e. The PopChart Embedder must still use the COM port to generate the embedding HTML, however it will embed a full URL request instead of an embedder session ID.

f. The PopChart Embedder must still use the COM port to generate the embedding HTML, however it will embed a full URL request instead of an embedder session ID.

5

PCSCRIPT

This chapter lists and describes PopChart Script (PCScript) commands for PopChart Server. PCScript is an object-oriented language used to manipulate objects in a PopChart. With it, you can send data to and customize the format of the graphs, legends, and text boxes inside your PopChart image.

Note: *The preferred method for customizing your PopChart is PopChart XML. Many 3.x users, however, are already familiar with PCScript. Additionally, PCScript can be convenient for quick customizations.*

HOW TO USE PCSCRIPT

This section describes how to use PopChart Script.

PCSCRIPT COMMAND FORMAT

The syntax for PCScript commands follows a convention similar to that of object-oriented languages such as C++ or Java. Each command consists of three parts—an object, method, and list of parameters. The code below shows the syntax for a PCScript command.

```
object.method(parameters)
```

Objects are created and named in the PopChart appearance file (see "About Appearance Files" on page 1-6 of the *PopChart Builder User Guide*). PCScript commands use the object names specified in the appearance file. For more info on object names, refer to "What is my Object Named?" on page 6-6 of the *PopChart Server User Guide*

The number of parameters for each command will depend on the method. Each parameter will be separated by the *parameters delimiter* (refer to the `Main.ParamDelimiter` method), which is usually a comma. For example, consider the `addPopupText` method for `Graph` objects. This method accepts three arguments—two numbers and a string. An `addPopupText` method call would appear as follows:

```
graph.addPopupText(1, 2, This is the PopUp Text)
```

Note: *Strings in PCScript do not have quotation marks around them. Because of this, if you have a comma or semi-colon in your string, PopChart Server will not be able to interpret your string correctly. You can get around this problem by changing the parameter or item delimiters using the* `Main.ParamDelimiter` *or* `Main.ItemDelimiter` *command.*

Note: *PCScript commands are not case sensitive.*

A complete list of object and method types is available in the "PCScript Reference" section at the end of this chapter.

LISTING MORE THAN ONE SET OF PARAMETERS PER PCSCRIPT COMMAND

To help shorten the length of the PCScript Command String, many methods accept more than one set of parameters per method call. In this case, each set of parameters is separated by the *item delimiter* (refer to the `Main.ItemDelimiter` method), which is usually a semi-colon. The syntax for such statements is:

```
object.method(parameter_list [;parameter_list]*)
```

For example, consider a PCScript command string that contains several `graph.addPopupText` method calls, as in the example below.

```
graph.addPopupText(1-4,1,Hello World)
  graph.addPopupText(1-4,2,Foo Bar)
  graph.addPopupText(1-4,3,Goodbye)
```

These PopUp text items could be specified in one method call instead of three:

```
graph.addPopupText(1-4,1,Hello World;1-4,2,Foo Bar;1-4
  ,3,Goodbye)
```

PCSCRIPT COMMAND STRINGS

PCScript commands are sent to PopChart Server via a PCScript command string. A PCScript command string is a string that contains all of the PCScript commands in the order that they are to be executed.

The commands can run together, so that nothing is separating them from each other, or they can be separated by white space. They should *not* be separated by a semi-colon or any other character.

Example 5.1 and Example 5.2 demonstrate two valid PCScript command strings, both of which contain the same commands.

Example 5.1 **PCScript Command String 1**

```
graph.setCategories(Apples)textbox.setText(Apple Graph)graph.setSe
              ries(Andy,10;Julie,14;Bryan,11)graph.addPopupText(1,
              1-3,Apples are Good)graph.DDEnable(1,1-3,http://www.
              applepies.com)
```

Example 5.2 **PCScript Command String 2**

```
graph.Categories(Apples)
textbox.setText(Apple Graph)      graph.Series(Andy,10;Julie,14;Bry
                an,11) graph.addPopupText(1,1-3,Apples are Good)
graph.DDEnable(1,1-3,http://www.applepies.com)
```

USING PCSCRIPT WITH THE PopChart Embedder

If you are requesting images from PopChart Server via the PopChart Embedder (refer to Chapter 4 and Chapter 5 of the *PopChart Server User Guide*), you will send your PCScript command string via the `pcScript` attribute.

This attribute should be set to your PCScript command string. The code below illustrates the use of the `pcScript` attribute in Java.

Example 5.3 Setting the PCScript Command String in PopChart Embedder

```
String pcscript="textbox.setText(Hello
                World)graph.setCategories(Apple)graph.setSeries(And
                y;10)setSeries(Joe;15)setSeries(Jenny;20)";
graphImage.pcScript = pcscript;
```

USING PCSCRIPT WITH THE @_PCSCRIPT SERVER COMMAND

If you are requesting images from PopChart Server via HTTP requests (refer to Chapter 11, "Getting PopChart Images with HTTP Requests," in the *PopChart Server User Guide*), you will send your PCScript command string to PopChart Server via the `@_PCSCRIPT` server command.

The code below shows a complete `@_PCSCRIPT` server command and PCScript command string.

```
@_PCSCRIPTtextbox.setText(Hello World)graph.setCategor
  ies(Apple)graph.setSeries(Andy;10)setSeries(Joe;15)s
  etSeries(Jenny;20)
```

Assuming PopChart Server is running on port 2001 on your local machine, this `@_PCSCRIPT` command could then be used to generate a PopChart image through the following HTTP request:

```
http://localhost:2001/?@_FILEexamples/apfiles/pie.bin@
  _PCSCRIPTtextbox.setText(Hello World)graph.Categorie
  s(Apple)graph.setSeries(Andy;10)setSeries(Joe;15)set
  Series(Jenny;20)
```

PCSCRIPT REFERENCE

This section is divided into five subsections, each of which deals with a separate class of PCScript objects. The five classes of objects are:

- **Main**
- **Graph**
- **Textbox**
- **Legend**
- **Bitmap**

MAIN OBJECT

Each PopChart appearance file contains one `Main` object. This object controls formatting options for the entire PopChart.

The `Main` object has the following methods:

- **Description**
- **Message**
- **ParamDelimiter**
- **ItemDelimiter**
- **SuppressAutoDescription**
- **Title**

These methods are described below.

Description

Available only in PopChart Server Enterprise.

Syntax `Main.Description(description)`

Description Specifies a description for the PopChart. This description will be displayed immediately under the PopChart's title in its text description (refer to Chapter 8, "Displaying 508 Compliant Descriptive Text," of the *PopChart Server User Guide*).

Parameters `Main.Description` accepts the following parameters.

description *String*
The description of the PopChart.

ItemDelimiter

Syntax `Main.ItemDelimiter(delimiter)`

Description

Sets the item delimiter to the specified character. This method affects PCScript commands for the current PopChart only.

The item delimiter divides data items from each other in the `Graph.SetSeries` method. Since a data series can have any number of data items, it uses two types of delimiters. The parameters delimiter delimits data values in each data item, while the item delimiter divides data items from each other (refer to the `Graph.SetSeries` method description for more details).

Similarly, the item delimiter is used to divide multiple Popup text or drill-down settings from each other in `Graph.AddPopupText` and `Graph.DDEnable` commands, respectively.

By default, the item delimiter for PCScript is a semi-colon *;* . This can cause problems when you are trying to pass strings containing semi-colons. The `Main.ItemDelimiter` method lets you change the delimiter to something else.

Example 5.4 illustrates the usefulness of this method. In this example, we want to be able to name a series *Fred; not Wilma*. If we tried to send this string with the default parameter delimiter, PopChart Server would think that the series was named *Fred*, and that the first data item in the series was *not Wilma*. However, since we set the delimiter to a carat ^, we are able to properly pass the data series. Each data item is divided from the others by a carat ^.

Important: *Unless you change the parameters delimiter using the `Main.ParamDelimiter` method, do not change the series delimiter to a comma , . This character is reserved for the delimiting parameters.*

Parameters

`Main.ItemDelimiter` accepts the following parameters.

delimiter *char*
The character that will delimit items in this PCScript command string.

Example 5.4 Changing the Item Delimiter

```
Main.ItemDelimiter(^)
graph.Series(Fred; not Wilma ^ 23,34 ^ 54,29)
graph.AddPopupText(1,1, Flintstones ^ 1,2, Vitamins)
```

Message

Syntax

`Main.Message(TextMessage)`

Description

Tells PopChart Server to generate a PopChart image that displays the specified message. All server commands, PCScript commands, and data will be ignored.

Parameters

`Main.Message` accepts the following parameters.

TextMessage *String*
The message that should be displayed in the PopChart image. As with all strings in PCScript, it should *not* be surrounded by quotation marks.

ParamDelimiter

Syntax `Main.ParamDelimiter(delimiter)`

Description Sets PCScript's parameter delimiter to the specified character. This method affects PCScript commands for the current PopChart only.

The parameter delimiter divides different parameters from each other inside of PCScript methods. By default, the parameter delimiter for PCScript is a comma , . This can cause problems when you are trying to pass strings that have commas in them. The `Main.ParamDelimiter` method lets you change the delimiter to something else.

The usefulness of this method is illustrated in Example 5.5. In this example, we attempt to name a category *Fred, Barney, and Wilma*. Of course, there would be no problem naming the category, since Categories are delimited with an item delimiter instead of a parameter delimiter. However, the problem arises when we try to create a drill-down item. Normally, PopChart Server would think that we were specifying three different parameters—*Fred*, *Barney*, and *and Wilma*. But since we set the delimiter to an asterisk *, we are able to properly pass the category name. Note that *Gus* and *Matilda*, the other category names, are delimited by the asterisk *.

Important: *Unless you change the item delimiter using the `Main.ItemDelimiter` method, do not change the parameter delimiter to a semi-colon ; . This character is reserved for delimiting data items.*

Parameters `Main.ParamDelimiter` accepts the following parameters.

delimiter *char*
The character that will delimit parameters for this PCScript command string.

Example 5.5 Changing the Parameter Delimiter

```
Main.ParamDelimiter(*)
graph.Categories(Fred, Barney, and Wilma * Gus * Matilda)
graph.Series(Apples, Oranges, and Pears * 67 * 89 * 93)
graph.DDEnable(Fred, Barney, and Wilma * 1-3 *
               http://fruitstocks.com)
```

SuppressAutoDescription

Available only in PopChart Server Enterprise.

Syntax `Main.SuppressAutoDescription()`

Description Suppresses automatic descriptive text generation for the entire PopChart. Only explicitly specified descriptions (via `Main.Description`, `Graph.Description`, or `TextBox.Description`) will appear in the text description.

Title

Available only in PopChart Server Enterprise.

Syntax `Main.Title(title)`

Description Specifies the title of the PopChart. This is used only in the text description of the PopChart (refer to Chapter 8, "Displaying 508 Compliant Descriptive Text," of the *PopChart Server User Guide*).

Parameters `Main.Title` accepts the following parameters.

title *String*
The title of the PopChart.

GRAPH OBJECTS

`Graph` objects represent each graph (i.e. Bar graph, Pie chart, Gauge, etc.) in a PopChart. There can be any number of `Graph` objects in an appearance file. Each `Graph` object has its own name, data, and formatting options.

Note: *For this section, our examples assume that the `Graph` object that we are using has the default name of graph. However, be aware that your `Graph` object is not always named graph. If such cases, you should use the object name in place of graph.*

`Graph` objects have the following methods:

- **AddNote**
- **AddPopupText**
- **AddScaleMarker**
- **AppendByRow**
- **DateInputFormat**
- **DDEnable**
- **DDPrefix**
- **Description**
- **EnableCategory**
- **EnableColumn**
- **EnableRow**
- **EnableSeries**
- **Hide**

- **LoadFile**
- **NumberOfLines**
- **ReverseDataOrder**
- **SetCategories**
- **SetDataLabelFormat**
- **SetGaugeRange**
- **SetGaugeValue**
- **SetScale**
- **SetSeries**
- **SetSeriesStyle**
- **SetSeriesToSecondScale**
- **SetStockHLOCOrder**
- **Show**
- **SuppressAutoDescription**
- **SuppressDescriptionItem**
- **Transposed**
- **TargetCategory**
- **UseColorPalette**

These methods are described below:

AddNote

Syntax

```
graph.AddNote(Category, Series, Text, [Textbox]
     [;Category, Series, Text, [Textbox]]*)
```

Description

Adds a PopChart Note to any data item(s) that are in the specified categor(ies) and series.

Notes are a combination of text boxes and PopUp text (in fact, another name for them is *sticky PopUp*). Like PopUp text, they are attached to a specific data item. However, like text boxes they are always visible. Notes "call-out" from the data item--there is a leader line from the data item to the note box.

You can change the format for notes generated dynamically inside of PopChart Builder. To do this, choose a graph and select Graph Properties > Notes. This dialog allows you to change the positioning, font, color, and other settings of dynamic notes.

Using an optional fourth parameter in the `addNote()` method, you can also attach any note to Textbox Objects. This gives you more control over the formatting of your note.

Each `Graph.AddNote` method can specify any number of Note boxes. The parameters for each different Note box should be separated from the parameters for other Note boxes by a semi-colon (or the item delimiter, which is specified in the `Main.ItemDelimiter` method).

Note: *If you are loading data via the* `graph.LoadFile` *command, be sure to call* `AddNote` ***after*** *you call* `graph.LoadFile`*. Otherwise,* PopChart Server *will be unable to associate your Note with the correct data item.*

Parameters

`Graph.AddNote` accepts the following parameters.

Category *{ int | int-int | Category Name }*

Specifies the category or categories of the data item(s) for which a Note will appear. This value can be an integer, a range of integers, or the name of a category that has been specified in a `Graph.SetCategories` method.

If the value of this parameter is an integer, it refers to the *nth* category listed in the `Graph.SetCategories` method, where *n* is the integer specified. So, for example, if *Fred* is the second category name listed in the `Graph.SetCategories` method, and we pass *2* as the value of the `categories` parameter, we are referring to the *Fred* category.

If you wish to access a category number that is the same as another category's name, you will need to use the pound # sign before the category number. This lets PopChart Server know that you are referring to a category number instead of a category name.

For example, if category number 5 is named *3*, then the statement `graph.AddNote(3,1,Hello)` will add a Note to category number 5, not 3, as it assumes you are referring to the category name. However, if you include a pound # sign before the number 3—`graph.AddNote(#3,1,Hello)`—then PopChart Server will override the category name and instead add a Note to category number 3.

You can denote a range of categories with an integer, followed by a hyphen, followed by another integer. The first integer represents the first category that should display a Note, while the last integer indicates the last category that should display a Note. All data items in categories between these two will also display the Note. Line 5 of Example 5.6 illustrates how to do this.

Because X-Y and Time graphs do not use categories, the value of this parameter for those graphs is the number of the data item in the specified data series to which you want to add PopUp text. Example 5.7 illustrates this difference.

Note: *Make sure you have Sort Data turned off in your appearance file (uncheck Properties > Graph Properties > General > Sort Data) if you are going to use Popup text with X-Y and Line graphs. Otherwise, the Note may appear at the wrong data point.*

Series *{ int | Series Name }*

Specifies the series of the data item(s) for which a Note will appear. This value can be an integer, a range of integers, or the name of a series that has been specified in a `Graph.SetSeries` method.

If the value of this parameter is an integer, it refers to the *nth* series listed in the `Graph.SetSeries` method(s), where *n* is the integer specified. For example, if *Fruit* is the name of the series described in the second `Graph.SetSeries` method, and we pass 2 as the value of the `series` parameter, we are referring to the *Fruit* series.

If you wish to access a series number that is the same as another series' name, you will need to use the pound # sign before the series number. This lets PopChart Server know that you are referring to a series number instead of a series name.

For example, if series number 5 is named *3*, then the statement `graph.AddNote(1,3,Hello)` will add PopUp text to series number 5, not 3, as it assumes you are referring to the series name. However, if you include a pound # sign before the number 3—`graph.AddNote(1,#3,Hello)`—then PopChart Server will override the series name and instead add a Note to series number 3.

A range of series is denoted by an integer, followed by a hyphen, followed by another integer. The first integer represents the first series that should display a Note, while the last integer indicates the last series that should display a Note. All data items in series between these two will also display a Note. This works similarly to the range of categories shown in Line 5 of Example 5.6.

Text *String*

The text that will appear in the Note box.

You can force a new line in the Note box text by insert a backslash and the letter *n* (*n*) where you want the new line to occur. However, when forcing a new line in JavaScript, *n* is interpreted as an escape code. You will need to use *n* in JavaScript to specify a new line.

Textbox *Textbox Object*

(Optional) This parameter allows you to specify the name of a text box object to which the note will be attached.

For example, if you have created a textbox (or notebox—for all practical purposes, they are the same) in PopChart Builder called notebox1, you can attach a Note to that box using the `addNote()` call on line 9 of Example 5.6.

Important: *If you are sending PCScript in an HTTP Request (i.e. not the PopChart Embedder), be sure to URL-encode (see "URL Encoding" on page 11-8 of the PopChart Server User Guide) this command, especially if the Note box text string contains spaces or other irregular characters.*

Example 5.6 **Adding Notes**

```
graph.setCategories(Barney; Wilma; Fred; Pebbles)
graph.setSeries(English;90;85;94;78)
graph.setSeries(Math;87;96;53;94)
graph.setSeries(Science;59;42;51;75)
graph.addNote(Fred, Math, Fred is failing Math)
graph.addNote(3, 2, Fred is failing Math)
graph.addNote(1-3, Science, Barney and Wilma and Fred are failing
              Science)
graph.addNote(Fred, 1, Fred has the best English score; 2, Math,
              Wilma has the best Math score)
graph.addNote(Wilma, Math, Wilma's math score was the highest score
             of all!, notebox1)
```

Lines 5 and 6 in Example 5.8 are the same command. The first just uses category and series names, while the second uses numbers. In both cases, the text *Fred is failing Math* will appear in a Note connected to the Fred category in the Math series.

Line 7 shows how to incorporate a range of data items. Data items that are in both the *Science* series and categories 1-3 display the PopUp text *Barney and Wilma and Fred are failing Science*.

Line 8 shows how more than one Note can be added with just one `Graph.AddNote` command.

Example 5.7 Adding Notes for an X-Y Line Graph

```
graph.setSeries(Blue; 7,10; 30,0; 23,30)
graph.setSeries(Red; 0,10; 5,0; 10,7)
graph.addNote(2, 2, Second Red Plot Point)
```

Example 5.9 shows how `Graph.AddNote` differs for X-Y and Time graphs. In this case, the text *Second Red Plot Point* will appear as a Note connected to the data point *(5,10)*.

AddPopupText

Syntax

```
graph.AddPopupText(Category, Series, Text
     [;Category, Series, Text]*)
```

Description

Adds PopUp text to any data item(s) that are in the specified categor(ies) and series.

PopUp text refers to a manually created box of text that appears as a user moves their mouse pointer over a specific data item. More information about PopUp text can be found in "PopUp Text" on page 7-12 of the *PopChart Server User Guide*.

Each `Graph.AddPopupText` method can specify any number of PopUp text boxes. The parameters for each different PopUp text item should be separated from the parameters for other PopUp text items by a semi-colon (or the item delimiter, which is specified in the `Main.ItemDelimiter` method).

Note: *If you are loading data via the `graph.LoadFile` command, be sure to call `AddPopUpText` **after** you call `graph.LoadFile`. Otherwise, PopChart Server will be unable to associate your PopUp text with the correct data item.*

Parameters

`Graph.AddPopupText` accepts the following parameters.

Category *{ int | int-int | Category Name }*
Specifies the category or categories of the data item(s) for which PopUp text will appear. This value can be an integer, a range of integers, or the name of a category that has been specified in a `Graph.SetCategories` method.

If the value of this parameter is an integer, it refers to the *nth* category listed in the `Graph.SetCategories` method, where *n* is the integer specified. So, for example, if *Fred* is the second category name listed in the `Graph.SetCategories` method, and we pass *2* as the value of the `categories` parameter, we are referring to the *Fred* category.

If you wish to access a category number that is the same as another category's name, you will need to use the pound # sign before the category number. This lets PopChart Server know that you are referring to a category number instead of a category name.

For example, if category number 5 is named *3*, then the statement `graph.AddPopupText(3,1,Hello)` will add PopUp text to category number 5, not 3, as it assumes you are referring to the category name. However, if you include a pound # sign before the number 3—`graph.AddPopUpText(#3,1,Hello)`—then PopChart Server will override the category name and instead add PopUp text to category number 3.

You can denote a range of categories with an integer, followed by a hyphen, followed by another integer. The first integer represents the first category that should display PopUp text, while the last integer indicates the last category that should display PopUp text. All data items in categories between these two will also display Popup text. Line 5 of Example 5.8 illustrates how to do this.

If you choose 0 as the category number, PopUp text will be added to the appropriate series name(s) in the legend.

Because X-Y and Time graphs do not use categories, the value of this parameter for those graphs is the number of the data item in the specified data series to which you want to add PopUp text. Example 5.9 illustrates this difference.

Note: *Make sure you have Sort Data turned off in your appearance file (uncheck Properties > Graph Properties > General > Sort Data) if you are going to use Popup text with X-Y and Line graphs. Otherwise, the PopUp text may appear at the wrong data point.*

Series *{ int | Series Name }*

Specifies the series of the data item(s) for which PopUp text will appear. This value can be an integer, a range of integers, or the name of a series that has been specified in a `Graph.SetSeries` method.

If the value of this parameter is an integer, it refers to the *nth* series listed in the `Graph.SetSeries` method(s), where *n* is the integer specified. For example, if *Fruit* is the name of the series described in the second `Graph.SetSeries` method, and we pass 2 as the value of the `series` parameter, we are referring to the *Fruit* series.

If you wish to access a series number that is the same as another series' name, you will need to use the pound # sign before the series number. This lets PopChart Server know that you are referring to a series number instead of a series name.

For example, if series number 5 is named *3*, then the statement `graph.AddPopupText(1,3,Hello)` will add PopUp text to series number 5, not 3, as it assumes you are referring to the series name. However, if you include a pound # sign before the number 3—`graph.AddPopUpText(1,#3,Hello)`—then PopChart Server will override the series name and instead add PopUp text to series number 3.

A range of series is denoted by an integer, followed by a hyphen, followed by another integer. The first integer represents the first series that should display PopUp text, while the last integer indicates the last series that should display PopUp text. All data items in series between these two will also display Popup text. This works similarly to the range of categories shown in Line 5 of Example 5.8.

Text *String*
The text that will appear as a user moves their mouse pointer over the specified data item(s).

You can force a new line in the PopUp text by insert a backslash and the letter *n* (\n) where you want the new line to occur. However, when forcing a new line in JavaScript, \n is interpreted as an escape code. You will need to use \\n in JavaScript to specify a new line.

Important: *If you are sending PCScript in an HTTP Request (i.e. not the PopChart Embedder), be sure to URL-encode (see "URL Encoding" on page 11-8 of the PopChart Server User Guide) this command, especially if the Popup text string contains spaces or other irregular characters.*

Example 5.8 **Adding PopUp Text**

```
graph.setCategories(Barney; Wilma; Fred; Pebbles)
graph.setSeries(English;90;85;94;78)
graph.setSeries(Math;87;96;53;94)
graph.setSeries(Science;59;42;51;75)
graph.addPopupText(Fred, Math, Fred is failing Math)
graph.addPopupText(3, 2, Fred is failing Math)
graph.addPopupText(1-3, Science, Barney and Wilma and Fred are
                failing Science)
graph.addPopupText(Fred, 1, Fred has the best English score; 2,
              Math, Wilma has the best Math score)
```

Lines 5 and 6 in Example 5.8 are the same command. The first just uses category and series names, while the second uses numbers. In both cases, the text *Fred is failing Math* will appear as the user moves the mouse pointer over the data item in the Fred category in the Math series.

Line 7 shows how to incorporate a range of data items. Data items that are in both the *Science* series and categories 1-3 display the PopUp text *Barney and Wilma and Fred are failing Science*.

Line 8 shows how more than one PopUp text box can be added with just one `Graph.AddPopupText` command.

Example 5.9 **Adding PopUp Text for an X-Y Line Graph**

```
graph.setSeries(Blue; 7,10; 30,0; 23,30)
graph.setSeries(Red; 0,10; 5,0; 10,7)
graph.addPopupText(2, 2, Second Red Plot Point)
```

Example 5.9 shows how `Graph.AddPopupText()` differs for X-Y and Time graphs. In this case, the Popup text *Second Red Plot Point* will appear as the user moves the mouse pointer over the data point *(5,10)*.

AddScaleMarker

Syntax

```
graph.AddScaleMarker(scale, type, low/width,
     high/position, color)
```

Description

Places a line over or highlighted range behind the graph at the specified scale index on the graph. This is helpful when you want to emphasize data items that are above or below a certain level, or emphasize data items that are in a certain range.

There are two types of scale markers: range and line. The line scale marker draws a line over the graph at a certain scale index, while the range scale marker highlights the area between two scale indices (highlighting will appear behind the graph).

Example 5.10 illustrates the difference between line and range scale markers. The first two commands create the green and red areas behind the graph. These are range scale markers. The last command creates a line scale marker, which is the red line that appears in front of the graph.

Pie graphs and Gauges are the only graphs that do not have scale markers.

Parameters

`Graph.AddScaleMarker` uses the following parameters.

scale *{ x | y | value | svalue | time }*

The scale that you want to put the value on. For most graphs, you will want to set this to *value* (value scale). For Line Bar Combo graphs, you can also choose *svalue* (secondary value scale—i.e. the scale on the right side of the graph). For Time Graphs, you can also choose the *time* scale (along the bottom). For X-Y and Time graphs, you should use either *x* (x-axis) or *y* (y-axis).

type *{ range | line }*

Sets the type of the scale markers—*line* or *range*.

low/width *int*

For line scale markers, this parameter sets the width of the line.

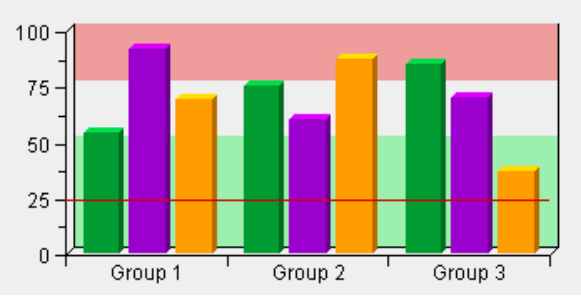
For range scale markers, this sets the value along the specified scale at which to begin highlighting.

high/position *int*
For line scale markers, this value is the position of the line along the scale specified with the `scale` parameter.

For range scale markers, this represents the value along the specified scale at which to end highlighting. This value must be higher than the value passed to the `low/width` parameter.

color *Color*
The color of the graph, using the standard six-digit hexadecimal representation (RRGGBB). There should *not* be a pound # sign before the color code.

Example 5.10 **Scale Markers in a Bar Graph**




```
graph.addScaleMarker(value, range,
   0, 50, 99eeaa)
graph.addScaleMarker(value, range,
   75, 100, ee9999)
graph.addScaleMarker(value, line, 1,
   25, cc0000)
```

AppendByRow

Syntax `graph.AppendByRow(true|false)`

Description Instructs PopChart Server on how to append data to the graph. This applies only to data appended by using the `append` switch on the `graph.LoadFile` command.

If the value is *true* PopChart Server will append the data as new rows (series) in the graph. If the value is *false*, PopChart Server will append the data as new columns (categories) in the graph. By default, it is set to *true*.

You must use this command *before* the `graph.LoadFile` command you want it to affect. This command can be called multiple times in the same PCScript command string.

Warning: If you call any `graph.Series` commands after setting this value to *false*, you will get very unpredictable results.

DateInputFormat

Syntax `graph.DateInputFormat(DateFormatString)`

Description Sets the date input format for Time Plot graphs.

The default input format for Time Plot graphs is *%m/%d/%Y* (month/day/year – e.g. *5/23/2000*). This method will override that format for this graph object only.

The base date for PopChart Server is the current date. For example, assume that today is Jan/15/2001 and your date input format is *%H:%M*. If you provide a date of *5:34*, then PopChart Server will treat the full date and time as *Jan/15/2001 05h:34m*.

The date input format can also be set in the appearance file using PopChart Builder by entering it in the *Properties > Graph Properties > General > Date Input Format* box.

Parameters `Graph.DateInputFormat` accepts the following parameters.

DateFormatString *String*
Sets the date input format. The string passed to this parameter can consist of text and any combination of macros listed in Table 5.5

TABLE 5.5 Date Formatting Macros

%Y	Four-digit year (e.g., 2001)
%y	* Two-digit year (e.g., 01)
%m	Month of year (1-12)
%d	Day of month (1-31)
%H	Hour (0-23)
%M	Minute (0-59)

Note: ** PopChart is Y2K compliant when using two-digit year values. Values 00 - 69 will be assigned to years 2000 - 2069. Values 70 - 99 will be assigned to years 1970 - 1999.*

Example 5.11 Setting the Date Input Format

```
graph.DateInputFormat(%Y-%m-%d %H.%M)
```

DDEnable

Syntax
```
graph.DDEnable(Category, Series, URL [,Target]
     [;Categories, Series, URL [,Target]]*)
```

Description Adds drill-down effects to any data item(s) that are in the specified categor(ies) and series.

Drill-down effects include linking to a different web page or executing a JavaScript function. The drill-down effect will occur whenever a user clicks his or her mouse over a specific data item. More information about drill-down effects is available in "Drill-down Effects" on page 7-14 of the *PopChart Server User Guide*.

You can use any combination of static text and data macros to create unique drill-down effects for every data item in your graph.

For example, suppose that whenever a user clicks on the graph, you want the graph to drill-down to a web page generated by your web application server at http://webapp.mycompany.com/drilldown. However, you also want to be able to tell the web application what data item the user just clicked on. One way of doing this would be to pass the following macro to the `Graph.DDEnable` method:

http://webapp.mycompany.com/drilldown?category=`%_CATEGORY_NAME`&series=`%_SERIES_NAME`.

So, for example, if a user clicks on the data item in the *South* category and the *Summer* series, the user would drill-down to this URL: http://webapp.mycompany.com/drilldown?category=South&series=Summer.

Example 5.14 illustrates how to make use of macros in a `Graph.DDEnable` statement. See the paragraph that follows it to learn what the example does.

Each `Graph.DDEnable` method can specify any number of drill-down effects. The parameters for each drill-down effect should be separated by a semi-colon (or the item delimiter, which is specified in the `Main.ItemDelimiter` method).

Note: *If you are loading data via the `graph.LoadFile` command, be sure to call `DDEnable` **after** you call `graph.LoadFile`. Otherwise, PopChart Server will be unable to associate your drill-down effect with the correct data item.*

Parameters

`Graph.DDEnable` accepts the following parameters.

Category *{ int | int-int | Category Name }*

Specifies the category or categories of the data item(s) for which drill-down effects should be enabled. This value can be an integer, a range of integers, or the name of a category that has been specified in a `Graph.AppendByRow` method.

If the value of this parameter is an integer, it refers to the *nth* category listed in the `Graph.SetCategories` method, where *n* is the integer specified. So, for example, if *Fred* is the second category name listed in the `Graph.SetCategories` method, and we pass *2* as the value of the `categories` parameter, we are referring to the *Fred* category.

If you wish to access a category number that is the same as another category's name, you will need to use the pound # sign before the category number. This lets PopChart Server know that you are referring to a category number instead of a category name.

For example, if category number 5 is named *3*, then the statement `graph.DDEnable(3,1,index.html)` will add a drill-down effect to category number 5, not 3, as it assumes you are referring to the category name. However, if you include a pound # sign before the number 3—

`graph.DDEnable(#3,1,Hello)`—then PopChart Server will override the category name and instead add the drill-down effect to category number 3.

A range of categories is denoted by an integer, followed by a hyphen, followed by another integer. The first integer represents the first category that should have this drill-down effect, while the last integer indicates the last category that should have the drill-down effect. All categories in between will also have the drill-down effect. Line 5 of Example 5.12 gives an example of a range of categories.

Because X-Y and Time graphs do not use categories, the value of this parameter for those graphs is the number of the data point in the specified data series to which you want to add a drill-down effect. Example 5.13 illustrates this difference.

If you choose 0 as the category number, the drill-down effect will be added to the appropriate series name(s) in the legend.

Note: *Make sure that you don't turn Sort Data for your appearance file (by checking Properties > Graph Properties > General > Sort Data) when you use drill-down effects with X-Y and Line graphs. Otherwise, the drill-down effects may occur at the wrong data point.*

Series *{ int | Series Name }*

Specifies the series of the data item(s) for which drill-down effects should be enabled. This value can be an integer, a range of integers, or the name of a series that has been specified in a `Graph.SetSeries` method.

If the value of this parameter is an integer, it refers to the *nth* series listed in the `Graph.SetSeries` method(s), where *n* is the integer specified. For example, if *Fruit* is the name of the series described in the second `Graph.SetSeries` method, and we pass 2 as the value of the `series` parameter, we are referring to the *Fruit* series.

If you wish to access a series number that is the same as another series' name, you will need to use the pound # sign before the series number. This lets PopChart Server know that you are referring to a series number instead of a series name.

For example, if series number 5 is named *3*, then the statement `graph.DDEnable(1,3,index.html)` will add a drill-down effect to series number 5, not 3, as it assumes you are referring to the series name. However, if you include a pound # sign before the number 3—`graph.DDEnable(1,#3,index.html)`—then PopChart Server will override the series name and instead add the drill-down effect to series number 3.

A range of series is denoted by an integer, followed by a hyphen, followed by another integer. The first integer represents the first series that should have this drill-down effect, while the last integer indicates the last series that should have the drill-down effect. All data series in between will also have the drill-down effect. A range of series looks exactly like the range of categories specified in Line 5 of Example 5.12.

URL *{ URL | JavaScript function }*

Either the web page that the data item(s) will drill-down to, or a JavaScript function.

If the `URL` parameter is a web page, the link will be relative to the location of the web page that contains the PopChart image. Or, by prefixing the URL with *http://*, you can specify an absolute path to a web page.

If the URL parameter is a JavaScript function, it must be prefixed with *javascript:*. The function must also have been declared within the web page that contains the PopChart image.

The URL can also contain macros, which represent values that are data item specific. In this manner, you can create a global drill-down destination for the graph (see Example 5.14). Table 5.6 lists the available drill-down macros.

TABLE 5.6 Drill-down Macros

Macro	Description	Graphs
%_CATEGORY_NAME	The name of the category that the data item belongs to.	All, except X-Y and Time Plot
%_CATEGORY_NUMBER	The number of the data series that the data item belongs to.	All, except X-Y and Time Plot
%_POINT_NUMBER	The number of the data item in the data series (e.g, the first plot point=1).	X-Y and Time Plot
%_SERIES_NAME	The name of the data series that the data item belongs to.	All
%_SERIES_NUMBER	The number of the data series that the data item belongs to.	All

Target *String*
The target within the web page (specified by the URL parameter) to which the data item(s) will drill-down. This parameter is optional and should not be used when drilling-down to a JavaScript function.

Example 5.12 Adding Drill-down Effects

```
graph.SetCategories(Produce; Poultry; Pastries)
graph.SetSeries(March;190;161;96)
graph.SetSeries(April;295;150;113)
graph.SetSeries(May;374;142;117)
graph.DDEnable(Pastries, May, pastry.html)
graph.DDEnable(3, 3, pastry.html)
graph.DDEnable(2-3, March, http://www.foodsales.com/march.html)
graph.AddPopupText(Poultry, 3, javascript:alert('Poultry sales are
            falling!'); 2, 1, poultry.html, march)
```

Lines 5 and 6 in Example 5.12 are the same command. The first just uses category and series names, while the second uses numbers. In both cases, the data item in the *Pastries* category and the *May* series will "drill-down" to the web page pastry.html when a user clicks on it.

Line 7 shows how to incorporate a range of data items. Data items in the *March* series and in categories 2-3 (*Poultry* and *Pastries*) drill-down to the web site http://www.foodsales.com/march.html.

Line 8 illustrates several drill-down features. First of all, it shows how more than one drill-down effect can be added with just one `Graph.DDEnable` command. Secondly, it shows how a JavaScript function may be used in place of a URL. In this case, the data item in the *Poultry* category and the *May* series will execute the function `alert('Poultry sales are falling!')` when the user clicks on it. Finally, the second drill-down effect shows how a target may be specified. In this case, the data item in the *Poultry* category and *March* series will drill-down to the target march in the web page poultry.html.

Example 5.13 Drill-down Effects for an X-Y Line Graph

```
graph.Series(Blue; 7,10; 30,0; 23,30)
graph.SetSeries(Red; 0,10; 5,0; 10,7)
graph.DDEnable(3, Blue, http://www.blue.com)
```

Example 5.13 shows how `Graph.DDEnable` differs for X-Y and Time graphs. In this case, the third data point in the *Blue* series *(23,30)* drills-down to the website http://www.blue.com.

Example 5.14 Adding a Global Drill-down Destination

```
graph.DDEnable(1-99,1-99,http://www.myappserver.com/newscript?Seri
                es=%_SERIES_NUMBER&Category=%_CATEGORY_NUMBER)
```

Example 5.14 shows how you can use `graph.DDEnable()` to create a global drill-down destination. We aren't sure how many series or categories are in the graph, so we simply set the range from 1 to 99, assuming that there will be no more than 99 series or categories. We could easily set this range higher if we are dealing with a larger data set.

The example also illustrates the use of macros in a `graph.DDEnable()` statement. Macros are textual keywords that represent values that are data item specific. Macros begin with a percentage mark and an underscore (%_). The macros in this statement, *%_CATEGORY_NUMBER* and *%_SERIES_NUMBER*, refer to the data item's category and series number, respectively. So, for example, if the a certain bar is in series 2 and category 9, the drill-down URL for the bar would be http://www.myappserver.com/newscript?Series=2&Category=9.

For a complete list of macros, refer to see Table 5.6 on page 5-29.

DDPrefix

Syntax `graph.DDPrefix(URL)`

Description Specifies a drill-down string for all data items in your graph.

This function can also be used to specify a prefix for all URL addresses or JavaScript functions passed to any subsequent `Graph.DDEnable` methods. If all of the web pages to which you drill-down share a common URL prefix, this method will help shorten your PCScript command string.

You should use this method only if *all* of the URLs have a common prefix, *including* the JavaScript functions. Otherwise, this prefix will be placed in front of links or functions that shouldn't have this prefix, rendering them invalid.

Note that this method effects only subsequent `Graph.DDEnable` method calls, meaning that any `Graph.DDEnable` method calls that appear before the `Graph.DDPrefix` method call will ignore the prefix. Also, you can use more than one `Graph.DDPrefix`, with the most recent prefix overriding any previous prefixes. One strategy that this allows for is breaking up your `Graph.DDEnable` method calls according to prefix, as demonstrated in Example 5.16.

Parameters `Graph.DDPrefix` accepts the following parameters.

URL *String*
Your drill-down string, or the prefix that PopChart Server should prepend to any URL addresses or JavaScript functions in subsequent `Graph.DDEnable` method calls.

Example 5.15 Creating a Drill-Down Prefix

```
graph.DDPrefix(http://webapp.mycompany.com/drilldown?)
```

Example 5.16 Using DDPrefix Multiple Time in the Same Graph

```
graph.DDPrefix(http://www.myserver.com/newgraphs/)
graph.DDEnable(8, 13, profits.html)
graph.DDPrefix(javascript:)
graph.DDEnable(1, 1-20, popupwindow(net); 2, 1-20,
               calculate(2,54,gross))
```

Example 5.16 shows two `Graph.DDPrefix` and `Graph.DDEnable` method calls. The effect of the first `graph.DDPrefix` method call is that the first `graph.DDEnable` drills-down to http://www.myserver.com/newgraphs/profits.html (not profits.html, as it would if there were no `graph.DDPrefix` command). The effect of the second `graph.DDPrefix` method call is that all drill-down URLs in the second

`graph.DDEnable` method call are preceded by *javascript:*, making them JavaScript functions.

Description

Syntax `graph.Description(description)`

Description Specifies a description for this graph. This description will be displayed at the beginning of this graph's section in the PopChart's descriptive text (refer to Chapter 8, "Displaying 508 Compliant Descriptive Text," of the *PopChart Server User Guide*).

Parameters `Graph.Description` accepts the following parameters.

description *String*
The description of this graph.

EnableCategory

Syntax
```
graph.EnableCategory(true|false, CategoryNumber,
     [,CategoryNumber]*)
```

Description This enables or disables a category of data. By default, all categories of data are enabled, so you will probably only use this command to filter data categories.

This command can be used to filter any number of categories. Each category should be separated by a comma. It can also filter ranges of categories, as illustrated in Example 5.17 below. A range of categories is denoted by a hyphen between the low and high value in the range.

Perhaps the main difference between this command and the `graph.EnableColumn()` command is that it is a post-processing command. It occurs after data from a spreadsheet is converted to series and categories. This means that `graph.Transposed()` applies to this command, while it doesn't apply to `graph.EnableColumn()`.

The usefulness of this method is illustrated with the following example—suppose you are importing data for a bar graph. When a user clicks on a certain category of bars, you want to drill-down to a line graph of just that category. By filtering out all of the other categories, you would be able to use the exact same data as you used from your bar graph, saving you from having to make another data query. For example, if a user drill-downed on the second column of data, you would add this PCScript command to filter out all of the other series:

Example 5.17 Hiding Data Categories

```
Graph.EnableCategories(false, 1, 3-6)
```

EnableColumn

Syntax

```
graph.EnableColumn(true|false, ColumnNumber,
     [,ColumnNumber]*)
```

Description

This enables or disables a column of data. By default, all columns of data are enabled, so you will probably only use this command to filter data columns.

This command is especially useful when importing HTML Tables or data files that you have no control over, as these data sources may have extraneous columns of information.

This command will filter the specified column(s) of data if the first parameter is set to false. Otherwise, it will unfilter it.

This command can filter any number of columns. Each column should be separated by a comma. It can also filter ranges of columns, as illustrated in Example 5.18 below. A range of columns is denoted by a hyphen between the low and high value in the range.

Example 5.18 Filtering Columns of Data

```
Graph.EnableColumn(false, 1, 6-99)
```

EnableRow

Syntax

```
graph.EnableRow(true|false, RowNumber,
     [,RowNumber]*)
```

Description

This enables or disables a row of data. By default, all rows of data are enabled, so you will probably only use this command to filter data rows.

This command is especially useful when importing HTML Tables or data files that you have no control over, as these data sources may have extraneous rows of information.

This command will filter the specified column(s) of data if the first parameter is set to false. Otherwise, it will unfilter it.

This command can filter any number of rows. Each row should be separated by a comma. It can also filter ranges of rows, as illustrated in Example 5.19 below. A range of rows is denoted by a hyphen between the low and high value in the range.

Example 5.19 Filtering Rows of Data

```
Graph.EnableColumn(false, 1, 3, 8-10)
```

EnableSeries

Syntax
```
graph.EnableSeries(true|false, SeriesNumber,
     [,SeriesNumber]*)
```

Description

This enables or disables a row of data. By default, all rows of data are enabled, so you will probably only use this command to filter data series.

This command can filter any number of series. Each series should be separated by a comma. It can also filter ranges of series, as illustrated in Example 5.20 below. A range of series is denoted by a hyphen between the low and high value in the range.

Perhaps the main difference between this command and the `graph.EnableRow()` command is that it is a post-processing command. It occurs after data from a spreadsheet is converted to series and categories. This means that `graph.Transposed()` applies to this command, while it doesn't apply to `graph.EnableRow()`.

The usefulness of this method is illustrated with the following example—suppose you are importing data for a bar graph. When a user clicks on a certain series of bars, you want to drill-down to a pareto graph, which only uses one series of data. By filtering out all of the other series, you would be able to use the exact same data as you used from your bar graph, saving you from having to make another data query. For example, if a user drill-downed on the fifth series of data, you would add this PCScript command to filter out all of the other series:

Example 5.20 Hiding Data Series

```
Graph.EnableSeries(false, 1-4, 6)
```

Hide

Syntax
```
graph.Hide()
```

Description

Hides this graph object.

LoadFile

Syntax
```
graph.LoadFile(FileName [,method [,HTMLTable]])
```

Description

Instructs PopChart Server to import data from a tab delimited data file, comma separated value file, xml database file, or a table in an HTML file. This file or web page must be accessible to the machine running PopChart Server.

By default, this command replaces all existing data in the graph. However, if you want to append the data from the file to the existing data, you can use the `append` parameter.

PopChart Server automatically detects the format of the file that you load. For more information on these data formats, refer to "Importing Data from Data Files" on page 6-16 of the *PopChart Server User Guide*.

Note: *You must add PopUp text, Notes, or Drill-down effects to your graph **after** you use this command, as otherwise PopChart Server will be unable to associate the special effect with the corresponding data item.*

Important: *Your path.xml file must grant PopChart Server permission to read files from the location that you specify. Otherwise, it will be unable to load the data. For more information, see "Setting Path Permissions" on page 3-37 of the PopChart Server User Guide.*

Note: *This command provides the same functionality as the `loadData(String,String,String,String)` PopChart Embedder method.*

Parameters

`Graph.LoadFile` accepts the following parameters.

FileName *String*

The path to and file name of the file or web page from which the data will be retrieved. This path will be relative to PopChart Server's document root. You can also use an absolute path or a URL. Make sure that your path.xml file permits PopChart Server to read from this path (see "Setting Path Permissions" on page 3-37 of the *PopChart Server User Guide*).

method *append | replace*

Indicates the method for importing the data. If `method` is set to `replace`, all data currently in the graph will be replaced. If it is set to `append`, the imported data will be appended to any data already in the graph, including data from the appearance file.

If this parameter is not set, it is assumed to be `replace`. It must always be set when importing data from HTML tables.

HTMLTable *{ int | title }*

The title or number of the table that should be loaded from the web page. If this parameter is a title, be sure that the title corresponds to the value of the `title` attribute of your `<table>` tag. If it is a number, PopChart Server will import data from the *n*th table in the web page, where *n* is the number you specify.

Example 5.22 illustrates how this parameter is used. The first line of code imports data from the table titled *pricing* on the web page stats.html. The second line of code imports data from the 11th table on the web page morestats.html.

This parameter must always be set when you import data from HTML tables.

Example 5.21 Loading a Comma Separated Value File

```
Graph.LoadFile(data/020115.csv)
```

Example 5.22 **Loading an HTML Table from a Web Page**

```
Graph.LoadFile(http://www.myserver.com/stats.html,replace,pricing)
Graph.LoadFile(\\database\morestats.html,append,11)
```

NumberOfLines

Syntax `graph.NumberOfLines(NumberOfLines)`

Description Specifies the number of lines in Line Bar Combo graphs.

For example, if you set this value to *2*, the first two data series will be displayed as lines. All remaining data series will be displayed as bars.

By default the Line Bar Combo graph has an equal number of lines and bars, and the Line/Stacked Bar Combo graph has one line. If you want to override the default setting, you can specify a new value using this command.

ReverseDataOrder

Syntax `graph.ReverseDataOrder()`

Description Reverses the order of the data items in the data series in the graph. For example, the last item in each data series that you sent to the graph will be the first data item to be displayed.

Note: *Does not work with X-Y or Time Plot graphs.*

Parameters `Graph.ReverseDataOrder()` does not accept any parameters.

SetCategories

Syntax `graph.SetCategories(CategoryName [;CategoryName]*)`

Description Sends the graph a set of category names separated by semi-colons (or the item delimiter, which is specified in the `Main.ItemDelimiter` method).

PopChart Server uses these category names to classify columns of data items. For example, the first data item in each `Graph.SetSeries` command is considered to be in the first column of data, and thus belongs to the first category name specified by the `Graph.SetCategories` command.

In most graphs, the category names become the values displayed along the x-axis (or category scale) of the graph. In some graphs, however, category names are used differently, or even ignored. Refer to Chapter 12, "Data Organization," for more information on how each graph type uses category names.

Note: *You must send the graph a set of categories before you can send it any series data. This means the* `Graph.SetCategories` *command must occur before any* `Graph.SetSeries` *commands. X-Y and Time Plot graphs are exceptions to this, as they do not use categories.*

This method can only be called once; if it is called again, it will erase all data in the graph and start anew.

The `Graph.SetCategories` also takes a special `CLR_` macro, which overrides the color for all data items in that category. This macro should be followed by a six or eight (RGB plus Alpha) digit hexadecimal color code, indicating the value of the overriding color. It is placed in parenthesis in front of the name of the category it should override, as illustrated in Example 5.24.

Parameters `Graph.SetCategories` accepts the following parameters.

CategoryName *String*
Specifies a category name. There can be any number of category names listed in a `Graph.SetCategories` method.

Example 5.23 Naming Your Categories

```
Graph.Categories(Fred; Barney; Gus; George)
```

Example 5.24 Changing Data Item Properties on the Fly

```
graph.SetSeries(Math; English; (CLR_00FF00)Science; History)
```

SetDataLabelFormat

Syntax `graph.SetDataLabelFormat(label)`

Description Specifies the format of the graph's data labels.

Data labels are small boxes of text and data that appear above each data item. If data labels are enabled, the string that you pass to the `setDataLabelFormat()` method will appear above every data label.

Note: *If you have text that you want to put above only one data item, you should use either PopUp text (see* `graph.AddPopupText`*) or Notes (see* `graph.AddNote`*).*

You can use macros to represent information that is data item specific, such as the value of the data item or the series name. For example, consider a data item in a series named *Houston* whose value is *38*. Using the command shown in Example 5.25, the label for this data item would be *Houston: 38 Pts.*.

If this command is not specified, the value from the appearance file will be used.

For more information on data labels, refer to "Data Labels" on page 7-5 of the *PopChart Server User Guide*.

Parameters `Graph.SetDataLabelFormat()` accepts the following parameters.

label *String*

Specifies the data label string. The string can contain both regular text and macros.

Macros will be replaced by information specific to each data item when PopChart Server generates the graph. A macro must always be proceeded by a %_, and all of the letters must be upper-case.

Table 5.7 shows the macros that you can use in this string.Some of these macros will not apply for all graph types-—for instance, `%_XVALUE` only applies to X-Y graphs, while `%_PERCENT_OF_TOTAL` only applies to bar, line, radar, and pie graphs.

TABLE 5.7 Data Label Macros

Macro	Description	Graphs
%_BUBBLE_VALUE	The value of the bubble in an X-Y or Time Bubble graph.	X-Y or Time Bubble
%_CATEGORY_NAME	The name of the category that the data item belongs to.	All, except X-Y and Time Plot
%_CATEGORY_TOTAL	The sum of all data values in the category to which the data item belongs (applies to	Area, Bar, Line, Pareto, Pie, Radar
%_CLOSE_VALUE	The close value for a high-low data item.	Stock
%_GRAPH_TOTAL	The sum of all data values in a bar, line, pie, or radar graph.	Area, Bar, Line, Pareto, Pie, Radar
%_HIGH_VALUE	The high value for a high-low data item.	Stock
%_LOW_VALUE	The low value for a high-low data item.	Stock
%_OPEN_VALUE	The open value for a high-low data item.	Stock
%_PERCENT_OF_CATEGORY	The data value represented as a percentage of the sum of all data values in its category.	Area, Bar, Line, Pareto, Pie, Radar
%_PERCENT_OF_TOTAL	The data value represented as a percentage of the sum of all data values in the graph.	Area, Bar, Line, Pareto, Pie, Radar
%_SERIES_NAME	The name of the data series that the data item belongs to.	All
%_TIME_VALUE	The time value for a Time Plot data item.	Time Plot
%_VALUE	The value of the data item.	Area, Bar, Line, Pareto, Pie, Radar

TABLE 5.7 Data Label Macros

Macro	Description	Graphs
%_XVALUE	The x value for an X-Y data item.	X-Y
%_YVALUE	The y value for an X-Y or Time Plot data item.	X-Y, Time Plot

Example 5.25 Customizing a Data Label

```
Graph.SetDataLabelFormat(%_SERIES_NAME: %_VALUE Pts.);
```

SetGaugeRange

Applies only to Gauges.

Syntax

```
gauge.SetGaugeRange(rangeName, color, min, max [;
     rangeName, color, min, max]*)
```

Description

Specifies color ranges for Gauges.

You can send as many color ranges as you want in a single `Graph.SetGaugeRange` call. The parameters for each range should be separated from each other by a semi-colon (or the series delimiter, which is specified in the `Main.ItemDelimiter` method). Example 5.26 illustrates how to do this.

For more information about gauges, refer to "Gauges" on page 13-35.

Parameters

`Graph.SetGaugeRange()` accepts the following parameters.

rangeName *String*
The name of the color range. This name will appear in the descriptive text for the gauge.

color *color*
The six-digit hexadecimal color code for the data range. The color should *not* be preceded by a pound # sign, as is done in HTML.

min *float*
The minimum value for the color range.

max *float*
The maximum value for the color range.

Example 5.26 **Setting Gauge Color Ranges**

```
gauge.setGaugeRange(Yellow, b2b200, 60, 80; Red, b20000, 80, 115)
```

SetGaugeValue

Applies only to Gauges.

Syntax
```
gauge.SetGaugeValue(value [, label [, min [,max]]])
```

Description
Specifies the value of a gauge. Can also specify the label of the gauge, as well as the minimum and maximum value of an LED Bar Gauge.

For more information about gauges, refer to "Gauges" on page 13-35.

Parameters
`Gauge.SetGaugeValue()` accepts the following parameters.

value *float*
The value of the gauge.

label *String*
The label for the gauge. This parameter is optional. If it is not specified, the value from the appearance file will be used.

min *float*
The minimum value of the range for a LED Bar Gauge. This parameter is optional. If it is not specified, the value from the appearance file will be used.

max *float*
The maximum value of the range for a LED Bar Gauge. This parameter is optional. If it is not specified, the value from the appearance file will be used.

Example 5.27 **Set the Value of a Gauge**

```
gauge.setGaugeValue(89, Server Load, 50, 110)
```

SetScale

Syntax
```
graph.SetScale(type, auto|manual, max|startdate
     [,min|enddate [,majorTicks]])
```

Description
Specifies scale settings, including automatic or manual scaling, scale range, and scale marker increments.

For information about scales, scale ranges, and scale markers, refer to refer to "Changing Scale Formatting Dynamically" on page 7-22 of the *PopChart Server User Guide*.

Parameters `Graph.SetScale()` accepts the following parameters.

type *{ value | svalue | x | y | y2 | time }*
Specifies which scale this method is changing the settings for.

For most graphs, you will only make changes to the *value* scale. Line Bar Combo graphs also have the *svalue* (secondary value) scale, which is on the right side of the graph. Time graphs have a *time* scale along the bottom of the graph, in addition to the *value* scale. X-Y graphs have both an *x* (x-axis) and *y* (y-axis) scale, but do not have a *value* scale. If a graph is a dual y scale graph, it will also have a y2 (right y-axis) scale.

scaling *{ auto | manual }*
Specifies whether to use automatic or manual scaling. When set to automatic, PopChart Server will try to adjust the scale range and the number of ticks and labels so that scale remains useful while the graph remains visibly appealing.

max | startdate *int*
Specifies the maximum value for the scale. For time scales, this value should be the starting date.

min | enddate *int*
Specifies the minimum value for the scale. This parameter is optional. For time scales, this value should be the ending date.

majorTicks *int*
Specifies the number of major increments in the scale. Major ticks are places where large tick marks and scale labels appear on the graph.

Example 5.28 Graph.SetScale Example Method Calls

```
Graph.SetScale(value,manual,100,0,5)
Graph.SetScale(x, auto, 78.5)
Graph.SetScale(y, manual, 500)
Graph.SetScale(time, auto, 5/1/2000, 5/31/2000)
```

SetSeries

Syntax `graph.SetSeries(SeriesName [;DataItem]*)`

Description Sends a data series to the graph.

Important: *PopChart Server 3.x users are probably accustomed to the `Graph.Series` method, which is now deprecated. The `Graph.SetSeries` method behaves somewhat differently than did `Graph.Series`. `Graph.Series` will still work in version 4.0.5, but support cannot be guaranteed for the 5.0 release.*

The first argument to this method should be the name of the series. Following this, you should have a semi-colon (or the item delimiter, which is specified in the `Main.ItemDelimiter` method). You should then list all data items in this series.

Data items can consist of anywhere from one to four values, depending on the graph type (refer to Chapter 12, "Data Organization," for details). If there is more than one value in the data item (e.g. an X-Y Plot point), the values should be separated by a comma (or the parameter delimiter, which is specified in the `Main.ParamDelimiter` method).

A data series can have as many data items as you want it to have. Each data item should be separated by an item delimiter.

The data values in a data item will be interpreted differently depending on what class of data the graph accepts. For details, refer to Chapter 12, "Data Organization."

The `Graph.SetSeries` method is unique in that it also accepts special color and symbol override commands. These commands dynamically change color or symbol properties for data items and/or series of data items. They are placed in parenthesis in front of the data items (for individual data items) or series names (for an entire series) that they are to effect, as demonstrated in Example 5.30.

The override commands are listed in Table 5.8 below.

TABLE 5.8 Color & Style Override Commands

Command	Description
CLR_	Overrides the color of a data item or series. The command should be followed by the six or eight digit (RGB plus Alpha) hexadecimal code of the overriding color.
FCLR_	Overrides the fill color of a data item or series. This command affects only Radar, Area, X-Y, and Time Plot graphs. It should be followed by the six digit hexadecimal code of the overriding color.
LCLR_	Overrides the color of a series line. This command can only precede a series name, and must be used on a series that is being displayed as a line (e.g. Line graph, Radar, X-Y Line). It should be followed by the six or eight digit (RGB plus Alpha) hexadecimal code of the overriding color
OCLR_	Overrides the outline color of a data item or series. This command affects only Bubble graphs. It should be followed by the six or eight digit (RGB plus Alpha) hexadecimal code of the overriding color
SCLR_	Overrides the symbol color of a data item or series. This command affects only Radar, X-Y, and Time Plot graphs. It should be followed by the six or eight digit (RGB plus Alpha) hexadecimal code of the overriding color
STYP_	Overrides the symbol type of a data item or series. This command affects only Radar, X-Y and Time Plot graphs. It should be followed by the symbol type. Table 5.9 on page 5-35 shows the symbol type enumeration.
DFCLR_	Overrides the fill color of Candlestick data items that closed down on the day. This command should be used only in front of a series name. It should be followed by the six or eight digit (RGB plus Alpha) hexadecimal code of the overriding color
DOCLR_	Overrides the outline color of Candlestick data items that closed down on the day. This command should be used only in front of a series name. It should be followed by the six or eight digit (RGB plus Alpha) hexadecimal code of the overriding color

TABLE 5.8 Color & Style Override Commands

Command	Description
UFCLR_	Overrides the fill color of Candlestick data items that closed up on the day. This command should be used only in front of a series name. It should be followed by the six or eight digit (RGB plus Alpha) hexadecimal code of the overriding color
UOCLR_	Overrides the outline color of Candlestick data items that closed up on the day. This command should be used only in front of a series name. It should be followed by the six or eight digit (RGB plus Alpha) hexadecimal code of the overriding color

Parameters `Graph.SetSeries` accepts the following parameters.

SeriesName *String*
The name of the series. This name will be shown in the legend.

DataItem *int [, int]**
A data item. Technically, a data item consists of any number of comma (or parameter delimiter) separated values, but the number of values that are actually used for the data item is between one and four, depending on the class of data that the graph accepts. For details, refer to Chapter 12, "Data Organization."

Each series can have an infinite number of data items. Each data item should be separated by a semi-colon (or the item delimiter, which is specified in the `Main.ItemDelimiter` method).

Example 5.29 Setting Data Series for a Standard Graph

```
graph.SetSeries(Math; 23; 95; 56)
graph.SetSeries(English; 65; 32; 66)
graph.SetSeries(History; 72; 58; 44)
graph.SetSeries(Science; 53; 69; 52)
```

Example 5.30 Changing Data Item Properties on the Fly

```
graph.SetSeries(Math; 23; (CLR_00FF00)95; 56)
graph.SetSeries((CLR_84EC90)English; 65; 32; 66)
xygraph.SetSeries((STYP_3)Plot Points; (SCLR_000099)10,15; 11,9;
             (SCLR_7180BB)20,4)
```

SetSeriesStyle

Syntax
```
graph.SetSeriesStyle(SeriesName, lineWidth,
     symbolType, areaFill, bubbleType)
```

Description

Specifies style settings, such as line width and symbol type, for Line, X-Y, and Time Plot graphs.

The first two parameters apply to Line graphs in addition to X-Y and Time Plot graphs. The last two only apply to X-Y and Time Plot graphs.

Note: *PopChart Server treats all X-Y and time Plot as Combo graphs. For example, if you set a bubble type in an X-Y Scatter graph, you will get a bubble, despite not being able to do this in the PopChart Builder interface.*

Parameters

`Graph.SetSeriesStyle` accepts the following parameters.

SeriesName *Series Name*
The name of the series for which you are setting the styles. This name must correspond to one that has been specified in a `Graph.SetSeries` command.

lineWidth *int*
The width of the line for this data series. Set this to 0 if you do not want the series to have a line.

symbolType *int*
Specifies the numeric code for the data series' symbol type.

Table 5.9 shows the symbol types with their corresponding enumeration.

TABLE 5.9 X-Y Symbol Types

Number	Description	Symbol	Number	Description	Symbol
0	None		7	Triangle	
1	Small Square		8	Triangle Outline	
2	Dot		9	Plus	
3	Square		10	Plus Outline	
4	Square Outline		11	Diamond	
5	Round		12	Diamond Outline	
6	Round Outline				

areaFill *{ true | false }*
Specifies whether or not the area underneath this graph's line should be filled in. Applies only to X-Y and Time Plot graphs.

bubbleType *{ 3D | FILL | NONE | OUTLINE | OUTLINEANDFILL }*
Specifies the type of bubble to display for these data items. This will automatically enable bubbles for the specified data series. Setting this to *NONE* will disable bubbles. Applies only to X-Y and Time Plot graphs.

Example 5.31 **Setting the Series Style**

```
graph.setSeriesStyle(Fred, 1, 7, no, 3D)
```

SetSeriesToSecondScale

Syntax `graph.SetSeriesToSecondScale(Series)`

Description Associates the specified series to the secondary scale for Dual Y Scale graphs (X-Y, Time).

Parameters

Series *{ int | Series Name }*

Specifies the series that should be associated with the secondary y scale. This value can be an integer, a range of integers, or the name of a series that has been specified in a `Graph.SetSeries` method.

If the value of this parameter is an integer, it refers to the *nth* series listed in the `Graph.SetSeries` method(s), where *n* is the integer specified. For example, if *Fruit* is the name of the series described in the second `Graph.SetSeries` method, and we pass 2 as the value of the `series` parameter, we are referring to the *Fruit* series.

If you wish to access a series number that is the same as another series' name, you will need to use the pound # sign before the series number. This lets PopChart Server know that you are referring to a series number instead of a series name.

For example, if series number 5 is named *3*, then the statement `graph.SetSeriesToSecondScale(3)` will set series number 5, not 3, to the secondary scale, as it assumes you are referring to the series name. However, if you include a pound # sign before the number 3—`graph.SetSeriesToSecondScale(#3)`—then PopChart Server will override the series name and set series number 3 to the secondary scale. You will need to URL-Encode (see "URL Encoding" on page 11-8 of the *PopChart Server User Guide*) your HTTP Request when you use a pound sign.

A range of series is denoted by an integer, followed by a hyphen, followed by another integer. The first integer represents the first series that should be added to the secondary scale, while the last integer indicates the last series that should be added to the secondary scale. All data series in between will also be added to the secondary scale. A range of series looks exactly like the range of categories specified in Line 5 of Example 5.12.

SetStockHLOCOrder

Syntax `graph.SetStockHLOCOrder(high, low, open, close)`

Description Sets the order of the data values in a Stock data item.

Stock data items contain four data values: high, low, open, and close. By default, when you send a stock data item to PopChart Server, it assumes that the values will be in this same order. However, if you want to input these data values in a different order, you can change that order with this command.

Example 5.32 illustrates how to use this method. This example tells PopChart Server, when reading a Stock data item, to expect the open value in the second column, then the close value, followed by the high and low values. When importing data from a table or file, column 1 is assumed to be the category name.

Note: *You must use this method before you import data to PopChart Server. Otherwise, PopChart Server will interpret the data items normally. You will probably only use this for imported data, particularly data from HTML Tables.*

For more information about the Stock data class, refer to "Stock Data Class" on page 12-13.

Parameters `Graph.SetStockHLOCOrder()` accepts the following parameters:

high *int*
Specifies which column of data will be used for a each data item's high value.

low *int*
Specifies which column of data will be used for a each data item's low value.

open *int*
Specifies which column of data will be used for a each data item's open value.

close *int*
Specifies which column of data will be used for a each data item's close value.

Example 5.32 Changing the Order of Data Values for a Stock Data Item

```
graph.setStockHLOCOrder(4, 5, 2, 3)
```

Show

Syntax `graph.Show()`

Description Makes this legend object visible.

By default, all objects are visible. You will only need to use this command if you need to make this object visible again after using the `legend.Hide()` command.

SuppressAutoDescription

Available only in PopChart Server Enterprise.

Syntax `graph.SuppressAutoDescription()`

Description Suppresses automatic descriptive text generation for this graph. Only the description specified via `Graph.Description` will appear in this graph's text description.

SuppressDescriptionItem

Available only in PopChart Server Enterprise.

Syntax `graph.SuppressDescriptionItem(itemtype)`

Description Suppresses descriptive text generation for the specified type of items in this graph object. Currently, you can suppress text descriptions for drill-down effects or PopUp text.

Parameters `Graph.SuppressDescriptionItem` accepts the following parameters.

itemtype *{ _CatDrilldown_ | _Drilldown_ | _PopUp_ | _SeriesDrillDown_}*
The item type for which you want to suppress descriptive text. Choosing *_CatDrilldown_* will suppress descriptive text for drill-down effects on category labels. Choosing *_Drilldown_* will suppress descriptive text for all drill-down effects. Choosing *_PopUp_* will suppress descriptive text for PopUp text. Choosing *_SeriesDrillDown_* will suppress descriptive text for drill-down effects on series labels.

Example 5.33 `Graph.SuppressDescriptionItem` Method

```
graph.SuppressDescriptionItem(_PopUp_)
```

Transposed

Syntax `graph.Transposed(true|false)`

Description Tells the graph whether or not to view the data transposed.

When a graph is transposed, data series become categories of data and categories of data become data series. In other words, the rows and columns of a graph are switched.

Because this command applies only to categorical data, you should only use it for data in the standard data class (refer to "Standard Data Class" on page 12-3). The effects of this command on non-standard data classes (such as x-y or stock) are unpredictable and, more than likely, not what you want.

TargetCategory

Syntax `graph.TargetCategory(CategoryNumber)`

Description Tells a pie graph which category to graph.

Since a pie graph only shows one category of data, you will need to tell PopChart Server which category should be graphed in a pie graph if you are sending it more than one category of data and you want to graph something other than the first category.

Parameters `Graph.TargetCategory` accepts the following parameters.

CategoryNumber *int*
The number of the category that should be graphed in the Pie graph.

Categories are numbered according to the order in which they appear in the `Graph.AppendByRow` method. Numbering begins with 1.

UseColorPalette

Syntax `graph.UseColorPalette(paletteName)`

Description Instructs PopChart Server to import a color palette from the config/PCColors.xml file. The graph's data series will be colored according to this palette.

For more information about color palettes and the PCColors.xml file, refer to Chapter 11, "Color Themes."

Parameters `Graph.UseColorPalette` accepts the following parameters.

paletteName *String*
The name of a color palette specified in the PCColors.xml file.

TEXTBOX OBJECTS

`Textbox` objects represent each textbox in a PopChart. There can be any number of `Textbox` objects in an appearance file. Each `Textbox` object has its own name, data, and formatting options.

Note: *For this section, our examples assume that the `Textbox` object that we are using has the default name of textbox. However, be aware that your `Textbox` object is not always named textbox. If such cases, you should use the object name in place of textbox.*

`Textbox` objects have the following methods:

- **DDEnable**
- **Description**
- **Hide**
- **SetText**
- **Show**
- **SuppressAutoDescription**
- **SuppressDescriptionItem**

These methods are described below.

DDEnable

Syntax `textbox.DDEnable(URL [,Target])`

Description Adds a drill-down effect to this textbox.

Drill-down effects include linking to a different web page or executing a JavaScript function. The drill-down effect will occur whenever a user clicks his or her mouse over a specific data item. More information about drill-down effects is available in "Drill-down Effects" on page 7-14 of the *PopChart Server User Guide*.

Parameters `Textbox.DDEnable` accepts the following parameters.

URL *{ URL | JavaScript function }*
Either the web page that the textbox will drill-down to, or a JavaScript function.

If the `URL` parameter is a web page, the link will be relative to the location of the web page that contains the PopChart image. Or, by prefixing the URL with *http://*, you can specify an absolute path to a web page.

If the `URL` parameter is a JavaScript function, it must be prefixed with *javascript:*. The function must also have been declared within the web page that contains the PopChart image.

Target *String*
The target within the web page (specified by the `URL` parameter) to which the data item(s) will drill-down. This parameter is optional and should not be used when drilling-down to a JavaScript function.

Example 5.34 `Textbox.DDEnable` Method Calls

```
textbox1.DDEnable(http://www.myserver.com/box.html)
textbox2.DDEnable(javascript:dostuff(true))
```

Description

Available only in PopChart Server Enterprise.

Syntax `textbox.Description(description)`

Description Specifies a description for this textbox object. This description will be displayed at the beginning of this textbox's section in the PopChart's descriptive text (refer to Chapter 8, "Displaying 508 Compliant Descriptive Text," of the *PopChart Server User Guide*).

Parameters `Textbox.Description` accepts the following parameters.

description *String*
The description of this textbox.

Hide

Syntax `textbox.Hide()`

Description Hides this text box.

SetText

Syntax `textbox.SetText(TextString)`

Description Sends the specified text to the text box

You can force a new line in the textbox by insert a backslash and the letter n (\n) where you want the new line to occur. Example 5.35 illustrates how you can do this and what the effect of it is.

Parameters Textbox.SetText accepts the following parameters.

TextString *String*
The text that should be displayed in the text box.

If it contains any spaces or irregular characters, you will probably want to URL-encode it (see "URL Encoding" on page 11-8 of the *PopChart Server User Guide*) when you are not using the PopChart Embedder.

Example 5.35 Setting the Text in a Text Box

```
textBox.SetText(Here's Line 1\nAnd Here's Line 2)
```

Note: *When forcing a new line in JavaScript, \n is interpreted as an escape code. You will need to use \\n in JavaScript to specify a new line.*

Show

Syntax `textbox.Show()`

Description Show the current text box.

By default, all objects are visible. You will only need to use this command if you need to make this object visible again after using the `textbox.Hide()` command.

SuppressAutoDescription

Available only in PopChart Server Enterprise.

Syntax `textbox.SuppressAutoDescription()`

Description Suppresses automatic descriptive text generation for this textbox. Only the description specified via `Textbox.Description` will appear in this textbox's text description.

SuppressDescriptionItem

Available only in PopChart Server Enterprise.

Syntax `textbox.SuppressDescriptionItem(itemtype)`

Description Suppresses descriptive text generation for the specified type of items in this textbox object. Currently, you can suppress text descriptions for drill-down effects.

Parameters `Textbox.SuppressDescriptionItem` accepts the following parameters.

itemtype *{ _Drilldown_ }*
The item type for which you want to suppress descriptive text. Choosing *_CatDrilldown_* will suppress descriptive text for drill-down effects on category labels. Choosing *_Drilldown_* will suppress descriptive text for all drill-down effects. Choosing *_PopUp_* will suppress descriptive text for PopUp text.

Example 5.36 `Textbox.SuppressDescriptionItem` Method

```
textbox.SuppressDescriptionItem(_Drilldown_)
```

LEGEND OBJECTS

`Legend` objects represent each legend in a PopChart. Though there is usually only one legend, there can be any number of `Legend` objects in an appearance file. Each `Legend` object has its own name, data, and formatting options.

Note: *For this section, our examples assume that the `Legend` object that we are using has the default name of* legend. *However, be aware that your `Legend` object is not always named* legend. *If such cases, you should use the object name in place of* legend.

`Legend` objects have the following methods:

- **Hide**
- **Show**

These methods are described below.

Hide

Syntax `legend.Hide()`

Description Hides this legend object.

Show

Syntax `legend.Show()`

Description Makes this legend object visible.

By default, all objects are visible. You will only need to use this command if you need to make this object visible again after using the `legend.Hide()` command.

BITMAP OBJECT

`Bitmap` objects represent graphics that have been imported into your PopChart. Each `Bitmap` object has its own name, data, and formatting options.

Note: *For this section, our examples assume that the `Bitmap` object that we are using has the default name of bitmap. However, be aware that your `Bitmap` object is not always named bitmap. If such cases, you should use the object name in place of bitmap.*

`Bitmap` objects have the following methods:

- **DDEnable**
- **Hide**
- **LoadFile**
- **Show**

These methods are described below.

DDEnable

Syntax `bitmap.DDEnable(URL [,Target])`

Description Adds a drill-down effect to this bitmap object.

Drill-down effects include linking to a different web page or executing a JavaScript function. The drill-down effect will occur whenever a user clicks his or her mouse over a specific data item. More information about drill-down effects is available in "Drill-down Effects" on page 7-14 of the *PopChart Server User Guide*.

Parameters `Bitmap.DDEnable` accepts the following parameters.

URL *{ URL | JavaScript function }*
Either the web page that the bitmap will drill-down to, or a JavaScript function.

If the `URL` parameter is a web page, the link will be relative to the location of the web page that contains the PopChart image. Or, by prefixing the URL with *http://*, you can specify an absolute path to a web page.

If the `URL` parameter is a JavaScript function, it must be prefixed with *javascript:*. The function must also have been declared within the web page that contains the PopChart image.

Target *String*
The target within the web page (specified by the `URL` parameter) to which the data item(s) will drill-down. This parameter is optional and should not be used when drilling-down to a JavaScript function.

Example 5.37 **`Bitmap.DDEnable` Method Calls**

```
bitmap1.DDEnable(http://www.myserver.com/logo.html)
bitmap2.DDEnable(javascript:dostuff(true))
```

Hide

Syntax `bitmap.Hide()`

Description Hides this bitmap object.

LoadFile

Syntax `bitmap.LoadFile(newimage)`

Description Loads a new image into the bitmap object.

This new image will replaces the image that was already in the bitmap object. Since the new image will be scaled to fit within the dimensions of the bitmap object, you should make sure that the new image is the same size (or at least maintains the same aspect ratio) as the old image.

This image file must be accessible to the machine running PopChart Server. It must be in either GIF, JPEG, or PNG format.

Important: *Your path.xml file must grant PopChart Server permission to read files from the location that you specify. Otherwise, it will be unable to load the data. For more information, see "Setting Path Permissions" on page 3-37 of the PopChart Server User Guide.*

Parameters `Bitmap.LoadFile` accepts the following parameters.

newimage *Filename*
The path to and location of the image that should be loaded into the bitmap. This path will be relative to PopChart Server's document root. You can also use an absolute path or a URL.

Show

Syntax `bitmap.Show()`

Description Makes this bitmap object visible.

By default, all objects are visible. You will only need to use this command if you need to make this object visible again after using the `bitmap.Hide()` command.

DEPRECATED METHODS

Several PCScript methods that existed prior to PopChart Server 4.0 have been deprecated in the 4.0 release. To maintain backwards compatibility, these methods will still work in 4.0.5. However, these deprecated methods will be discontinued in the 5.0 release.

Table 5.10 below lists the deprecated methods, along with their 4.0 equivalent. Please read the description of the new 4.0 methods, as they are likely to work differently than their 3.x equivalents. You can click on any 4.0 methods to jump to a description of that method.

TABLE 5.10 Deprecated PCScript Methods

Deprecated Method	4.0 Equivalent
`Main.SeriesDelimiter`	`Main.ItemDelimiter`
`Graph.Categories`	`Graph.SetCategories`
`Graph.FilterSeries`	`Graph.EnableCategory`
`Graph.FilterCategories`	`Graph.EnableSeries`
`Graph.Series`	`Graph.SetSeries`
`Graph.SetXYSeriesStyle`	`Graph.SetSeriesStyle`

6

SERVER COMMANDS

This chapter lists and describes server commands for PopChart Server. Server commands are one way to communicate with PopChart Server. They can be included in the query string of an HTTP request to PopChart Server, or in a separate server command file.

HOW TO USE SERVER COMMANDS

This section will discuss how to use server commands, including how to format them and how to send them to PopChart Server.

SERVER COMMAND FORMAT

All server commands begin with the characters `@_`. They are also all capitalized. Examples of server commands include `@_FILE`, `@_PCSCRIPT`, and `@_GIF`.

Each server command takes either zero or one arguments. If a command takes an argument, the argument should immediately follow the command. The argument should not be in parenthesis, nor should there be a space or any other character between the argument and the command.

For example, the `@_FILE` server command takes one argument, a file name. If the filename is apfles/graph1.bin, the full server command with the argument would be:

```
@_FILEapfiles/graph1.bin
```

SERVER COMMAND STRINGS

Server commands are issued to PopChart Server via a *server command string*.

In a URL or HTTP request, the server command string consists of any number of server commands, all joined together in one long and unbroken line. Each command should follow the previous without any sort of interruption (i.e. no semi-colon, period, line-break, etc.).

For example, if we wanted to issue the commands `@_FLASH`, `@_SAVEimage1.swf`, `@_DONTCACHE`, and `@_PWpass`, we could use the following command string:

```
@_PWpass@_SAVEimage1.swf@_DONTCACHE@_FLASH
```

In a server command file, white space is not a problem. You can put commands on separate lines or on the same line. The entire file will be considered to be a server command string.

Note: *The order of the commands does not matter.*

SERVER COMMANDS IN HTTP REQUESTS

When issuing server commands to PopChart Server from an HTTP request, you should first enter the URL of PopChart Server (including the address and port), followed by a question ? mark, and finally the server command string.

For example, let's assume that PopChart Server is running at http://pcis.mycompany.com:81. If our server command string is `@_GIF@_FILEexamples/apfiles/bar.pcxml@_PCSCRIPTtextbox.settext(Commodities)`, we would make the following HTTP request:

```
http://pcis.mycompany.com:2001/?@_GIF@_FILEexamples/ap
  files/bar.pcxml@_PCSCRIPTtextbox.settext(Commodities
  )
```

You can use this HTTP request as either the URL for a web page, in which case the browser will show only the PopChart image, or you can integrate it into a webpage by making it the source of an image or an object tag. In both cases, if your HTTP request contains a space or other special character, you will need to URL-encode it using JavaScript or some other method. This is because URLs cannot contain spaces or other special characters.

URL-encoding is discussed in "URL Encoding" on page 11-8 of the *PopChart Server User Guide*. Embedding a PopChart image generated by an HTTP request into a web page is discussed in Chapter 11, "Getting PopChart Images with HTTP Requests," in the *PopChart Server User Guide*.

SERVER COMMANDS IN THE PopChart Embedder

Most of the functionality of server commands is available through the various PopChart Embedder APIs.

Occasionally, you may find it necessary to use server command functionality that isn't available through the PopChart Embedder APIs. If this is the case, you can send server commands directly to PopChart Server through the PopChart Embedder's `extraPCSCommands` attribute. The only parameter to this method is a server command string. This server command string should be in the same format as that which was explained above for server command strings in HTTP requests, although you do not need to URL-encode this string.

For example, if you are using the Java PopChart Embedder and have an instance of the PopChart Embedder object named `graphImage`, the following method call would allow you to use the `@_LOADREQUEST` server command:

```
graphImage.setExtraPCSCommands
  ("@_LOADREQUESTcommands1.txt");
```

SERVER COMMAND LIST

This is a brief list of all available server commands. If you are viewing this documentation electronically, you can jump to a description of the command by clicking on it.

- **@_BGCOLOR**
- **@_DONTCACHE**
- **@_DONTGZIPSVG**
- **@_EPS**
- **@_FILE**
- **@_FLASH**
- **@_FLUSH**
- **@_GIF**
- **@_GRADCOLOR**
- **@_GRADDIRECTION**
- **@_HEIGHT**
- **@_LOAD**
- **@_LOADPCXML**
- **@_LOADREQUEST**
- **@_LOG**
- **@_LOGGRAPH**
- **@_PCSCRIPT**
- **@_PCXML**
- **@_PDF**
- **@_PNG**
- **@_PW**
- **@_SAVE**
- **@_SHOWTABLESFROMURL**
- **@_SVG**
- **@_TEXTDESCRIPTION**
- **@_USESVGTEMPLATE**
- **@_WBMP**
- **@_WIDTH**

SERVER COMMAND REFERENCE

The remainder of this chapter describes each of the server commands individually. The commands are described in alphabetical order.

@_BGCOLOR

Syntax `@_BGCOLORFFFFFF`

Description This command specifies the background color for the image. The argument should be a six-digit hexadecimal number using the typical `RRGGBB` convention. Do not include a pound `#` sign before this number. The default value is *FFFFFF* (white).

@_DONTCACHE

Syntax `@_DONTCACHE`

Description This command serves two purposes. First of all, it instructs the web browser not to cache the image, even if caching is enabled on the browser. This is useful when graphing data from a database where the data changes frequently.

Secondly, it instructs PopChart Server not to cache the image on the server side, even if caching is enabled. This latter purpose affects only PopChart Server Enterprise, as it is the only product that features server-side caching.

@_DONTGZIPSVG

Syntax `@_DONTGZIPSVG`

Description This command instructs PopChart Server not to gzip an SVG image. Doing so increases the size of the SVG image, but may be necessary in certain environments where you need to be able to parse or examine the contents of the SVG image that PopChart Server returns.

This command only affects SVG images.

@_EPS

Available only in PopChart Server Enterprise.

Syntax `@_EPS`

Description This command instructs PopChart Server to override the default image type and generate an EPS image.

@_FILE

Syntax `@_FILEexamples/apfiles/bar.pcxml`

Description This command specifies the path and name of your appearance file. It is required for all PopChart Server requests except those using the `@_LOAD` command.

The argument to this command should be the location of an appearance file. The path to the file is relative to PopChart Server's document root, however you can also use an absolute path. You can also enter a URL.

Important: *Your path.xml file must grant PopChart Server permission to read files from the location that you specify. Otherwise, it will be unable to load the data. For more information, see "Setting Path Permissions" on page 3-37 of the PopChart Server User Guide.*

@_FLASH

Available only in PopChart Server Pro and PopChart Server Enterprise.

Syntax `@_FLASH`

Description This command instructs PopChart Server to override the default image type and generate a FLASH image.

@_FLUSH

Available only in PopChart Server Enterprise.

Syntax `@_FLUSH@_PWpasswd`

Description This command deletes everything that is in the cache. You will need to specify a password using the `@_PW` command in order to execute `@_FLUSH`.

@_GIF

Syntax `@_GIF`

Description This command instructs PopChart Server to override the default image type and generate a GIF image (or a PNG image if Automatic PNG Detection (see "Automatic PNG Detection" on page 3-27 of the *PopChart Server User Guide*) is enabled and the browser supports the PNG format).

@_GRADCOLOR

Syntax `@_GRADCOLORFFFFFF`

Description This command specifies the gradient color for the image. If specified, this color will be used in conjuction with the background color (set in the appearance file or with `@_BGCOLOR`) to create a gradient background for the PopChart image.

The argument should be a six-digit hexadecimal number using the typical `RRGGBB` convention. Do not include a pound # sign before this number. The default value is *FFFFFF* (white). You can also specify `NONE` if you do not want a gradient.

@_GRADDIRECTION

Syntax `@_GRADDIRECTION{TB|BT|LR|RL}`

Description Allows you to specify the direction of the background gradient. You should specify your preference at the end of the command. The available settings are: `TB` (top-to-bottom), `BT` (bottom-to-top), `LR` (left-to-right), and `RL` (right-to-left).

These settings specify movement from the background color to the gradient color. For example, if you set this to TB (top-to-bottom), the top of the background will be painted in the background color, while the bottom will be painted in the gradient color.

This command will have no effect if a gradient color has not been specified in the appearance file or with the `@_GRADCOLOR` command.

@_HEIGHT

Syntax `@_HEIGHT300`

Description This command specifies the height of the image to be generated. If none is provided, PopChart Server will use the height of the appearance file.

This command takes as its argument an integer, representing the height of the generated image in pixels.

@_LOAD

Syntax `@_LOADimages/myimage.gif`

Description This command instructs PopChart Server to return the specified image. It will not generate any images, and any commands that pertain to image generation will be ignored.

The argument to the command should be the path to and name of the image file that should be returned. The path is relative to the document root. Alternatively you can specify an absolute path.

Important: *Because of the way PopChart Server returns the image, you must make sure that PopChart Server knows what format the image is in. PopChart Server does not look at a file's extension and therefore cannot discern the file format by itself.*

If the image is of the default format, you won't need to use any other commands. If it is not of the default format, however, you should use an appropriate command to override the default format. For example, if the default format is SVG and you want to load a GIF image, be sure to specify the `@_GIF` command in addition to the `@_LOAD` command.

Important: *Your path.xml file must grant PopChart Server permission to read files from the location that you specify. Otherwise, it will be unable to load the data. For more information, see "Setting Path Permissions" on page 3-37 of the PopChart Server User Guide.*

@_LOADPCXML

Syntax `@_LOADPCXMLpath`

Description This command loads the specified PopChart XML file from a local path (relative to the document root), an absolute path, or from a URL.

The PopChart XML in this file will be appended to the PopChart XML from the appearance file and from any previous `@_LOADPCXML` commands.

Important: *Your path.xml file must grant PopChart Server permission to read files from the location that you specify. Otherwise, it will be unable to load the data. For more information, see "Setting Path Permissions" on page 3-37 of the PopChart Server User Guide.*

PopChart XML is described in greater detail in Chapter 10, "Using PopChart XML," in the *PopChart Server User Guide*.

@_LOADREQUEST

Syntax
```
@_LOADREQUESTcommands/request.txt
@_LOADREQUESThttp://myserver.com/pccommands?month=2
```

Description This command loads the specified server command file from a local path (relative to the document root), an absolute path, or from a URL. The commands in this file will be appended to the command string. The file should consist of one line—a command string in the format specified at the beginning of this chapter.

If one of the following commands is included both in the HTTP request to PopChart Server and in the server command file, the command in the server command file will be ignored.

- **@_FILE**
- **@_SAVE**
- **@_BGCOLOR**
- **@_GRADCOLOR**
- **@_GRADDIRECTION**
- **@_WIDTH**
- **@_HEIGHT**

- **@_GIF**
- **@_FLASH**
- **@_WBMP**
- **@_SVG**
- **@_PNG**

Otherwise, any commands in the HTTP request to PopChart Server will be ignored.

Important: *Your path.xml file must grant PopChart Server permission to read files from the location that you specify. Otherwise, it will be unable to load the data. For more information, see "Setting Path Permissions" on page 3-37 of the PopChart Server User Guide.*

Server command files are described in greater detail in "Server Command Files" on page 6-13 of the *PopChart Server User Guide*.

@_LOG

Syntax `@_LOG@_PWpasswd`

Description This command will return a text-only web page displaying PopChart Server's log.

For security reasons, you will need to specify a password using the `@_PW` command in order to view the log.

This log is exactly what you see when you look at the Server > Statistics page in the Administration Console. It keeps track of several statistics, such as how many hits PopChart Server has received. If you enable transaction logging (refer to "Using the Server Log and Console Output" on page 3-21 of the *PopChart Server User Guide*), it will also display the PopChart Server's most recent transactions.

@_LOGGRAPH

Syntax `@_LOGGRAPH@_PWpasswd`

Description This command feeds hourly or daily usage data to PopChart Server so that you can graph it. The Administration Console automatically graphs this data for you in its Performance Charts, but you may use this command if you don't want to use the Administration Console.

For security reasons, you will need to specify a password using the `@_PW` command when trying to access the `@_LOGGRAPH` data.

PopChart Server keeps statistics on hourly hits over the past seven days. It also tracks of daily hits for the past 90 days. This data is kept in the logs folder of the PopChart Server root directory. Deleting the logs folder will effectively disable usage tracking.

In order for PopChart Server to graph this data, you will need to specify an appearance file using the `@_FILE` command in your server command string. The data is formatted for Time

Line graphs, so the appearance file must contain at least one Time Line graph. You will also need to properly specify PCScript commands for the graph and data.

The data is available inside of PCScript via three macros:

- `PCIS.totalHits` - total number of hits.
- `PCIS.hitsByHourSeries` - list of data indicating hourly hits.
- `PCIS.hitsByDaySeries` - list of data indicating daily hits.

The data lists are formatted for Time Line graphs (i.e. in the format `time, hits, time, hits,` etc.). These macros can be included anywhere in the PCScript, but you will probably want to place the data lists as the second argument to a `graph.series()` (link) command. For example, the following PCScript statement will send the hourly usage data to the series called *Hour* for the graph named *graph*:

```
graph.series(Daily Hits, PCSCript.hitsByDaySeries)
```

Time will be in the format `%Y/%m/%d:%H` (e.g. *1999/06/23:6*) for both `PCIS.hitsByHourSeries` and `PCIS.hitsByDaySeries`. Thus, you must set the date input format appropriately. This can be done with the following `graph.DateInputFormat` (link) PCScript command:

```
graph.DateInputFormat(%Y/%m/%d:%H)
```

@_PCSCRIPT

Syntax `@_PCSCRIPTpcscriptCommandString`

Description This command specifies the PCScript commands for the graph(s) you want to generate. PCScript is used to send information such as data and formatting options to the graphs, legends, and textbox objects within a PopChart image. Chapter 5, "PCScript," contains a complete list of PCScript commands.

The argument to the `@_PCSCRIPT` command is a string containing all of the PCScript commands that you wish to use.

In a URL, there should be nothing separating the PCScript commands from each other. The example below illustrates the proper format for a `@_PCSCRIPT` command in a URL.

```
@_PCSCRIPTtextbox.settext("Hello
  World")graph.AddPopupText(#3,#2,High)graph.
  AddPopupText(#1,#4,Low)
```

In other situations (e.g. server command files), there can be any amount of white space between PCScript commands.

@_PCXML

Syntax `@_PCXMLpcxmlString`

Note: *This command can only be used within a server command file, loaded by the `@_LOADREQUEST` server command or the `loadCommandFile(String)` PopChart Embedder method.*

This command specifies PCXML to send to PopChart Server. This PCXML will be appended to the appearance file, overriding any values that appear both in this command and in the appearance file. It works much like PopChart Embedder's `addPCXML(String)` method.

You can use multiple `@_PCXML` statements. Example 6.1 below illustrates several ways you can use `@_PCXML`.

Example 6.1 **Using `@_PCXML`**

```
@_PCXML<?xml ... ?><Chart>xml data</Chart>
@_PCXML<Graph name="graph" Type="Bar" SubType="Basic"></Graph>
@_BGCOLORCCCC00
@_PCXML<Textbox name="title"><Text>Bar Chart Textbox
                Text</Text></Textbox>
@_PCSCRIPTgraph.setseries(...)
@_END
```

@_PDF

Available only in PopChart Server Enterprise.

Syntax `@_PDF`

Description This command instructs PopChart Server to override the default image type and generate a PDF image.

@_PNG

Syntax `@_PNG`

Description This command instructs PopChart Server to override the default image type and generate a PNG image.

@_PW

Syntax `@_PWpasswd`

Description This command specifies your PopChart Server password. For security reasons, certain server commands always require a password. These include:

- **@_FLUSH**
- **@_LOG**
- **@_LOGGRAPH**
- **@_SAVE**

You should enter your password as the argument to the `@_PW` command.

If you wish to turn off this security feature, you can do so using the `-nopwforsave` configuration setting, which you can enter either in the configuration file.

Warning: Do not include this command in anything that can be seen by an end user, such as a link or an image tag in an HTML document.

@_SAVE

Syntax `@_SAVEimages/myimage.gif@_PWpassword`

Description This command directs PopChart Server to save the generated image to the specified location. The image will still be returned to the browser.

After an image has been saved, you can use `@_LOAD` command to load it. This keeps you from having to generate it again. Using `@_SAVE` and `@_LOAD` can greatly improve PopChart Server's performance if it is serving the same image over and over again.

For security reasons, you will need to specify a password using the `@_PW` command in order to use the `@_SAVE` command. Or, if you don't want to specify a password, you can set the `-nopwforsave` configuration setting.

Important: *Make sure that your path.xml file gives PopChart Server permission to save to the location you specify. Otherwise, PopChart Server will not be able to save the file. By default, PopChart Server can only save to the images folder of its document root (see "Setting Path Permissions" on page 3-37 of the PopChart Server User Guide).*

The argument to this command is the file to which you want to save the image. The file name is relative to PopChart Server's document root directory. Alternatively, you can specify an absolute path.

@_SHOWTABLESFROMURL

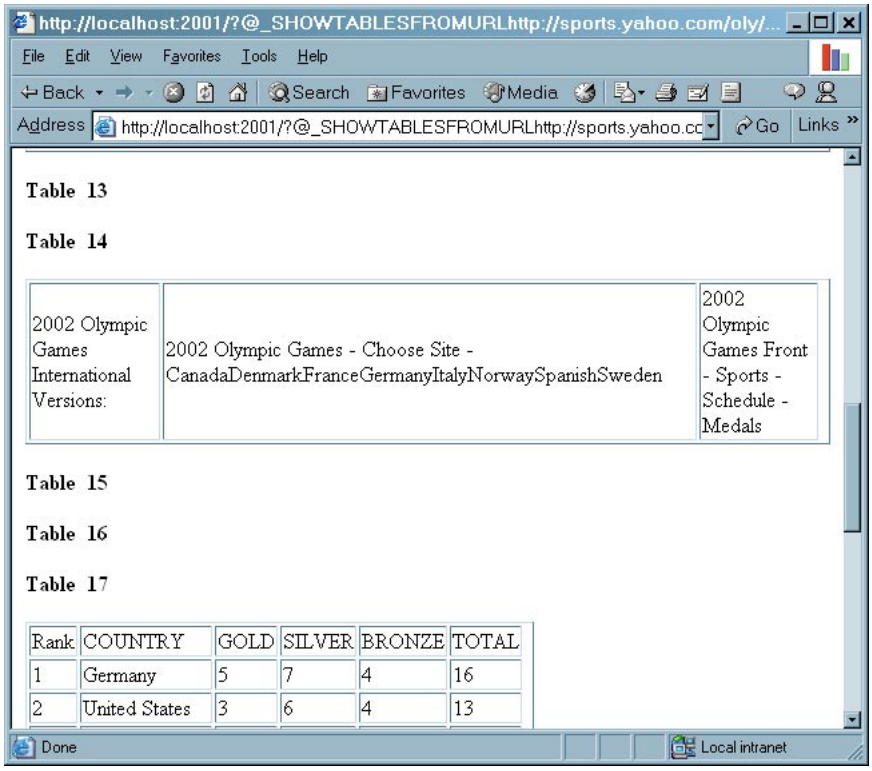
Syntax `@_SHOWTABLESFROMURLhttp://sports.yahoo.com/oly/medals.html`

Description

This command outputs a list of all tables in a web page with their corresponding table numbers. The output will help you determine what table number you should send to PopChart Server when you import data from an HTML Table using the PopChart Embedder `loadData()` method or the PCScript `LoadFile()` method.

Note: *When you use this command it will override all other server commands, so for obvious reasons you will only want to use this command while you are setting up your PopCharts.*

When you use the `@_SHOWTABLESFROMURL` command, PopChart Server will return a web page similar to the following.



In this situation, table *17* is the only one that contains useful data.

Important: *In order for PopChart Server to list the tables in a web page, it must be given permission to read data from the specified path or domain. For more information, see "Setting Path Permissions" on page 3-37 of the PopChart Server User Guide.*

@_SVG

Available only in PopChart Server Pro and PopChart Server Enterprise.

Syntax `@_SVG`

Description This command instructs PopChart Server to override the default image type and generate an SVG image.

@_TEXTDESCRIPTION

Syntax `@_TEXTDESCRIPTIONEN`

Description This command instructs PopChart Server to return a text description of the PopChart image (refer to "Displaying 508 Compliant Descriptive Text" on page 8-1 of the *PopChart Server User Guide*). It accepts as it's argument a two character language code, which tells PopChart Server which language settings to use.

Currently, the only language available in the standard PopChart Server installation is English (*EN*).

A text description is not an image—you should treat it as its own separate web page.

@_USESVGTEMPLATE

Available only in PopChart Server Pro and PopChart Server Enterprise.

Syntax `@_USESVGTEMPLATEsvg_templates/grow.svg`

Description Specifies the location of an SVG template that should be applied to your PopChart image. This location is relative to your document root (usually chart_root). It can also be an absolute path or URL.

SVG templates are SVG files that specify a transformation or special effect for a PopChart image. Examples of such effects include fading, rotating, and sliding. The effect or transformation is applied *after* the PopChart image has been generated.

Several popular templates are located in the chart_root/svg_template directory. For consistency, we recommend that you also put any other SVG templates in this directory.

Important: *Make sure that your path.xml file gives PopChart Server permission to read from the location where your SVG Template is located. Otherwise, PopChart Server will not be able to read the file (see "Setting Path Permissions" on page 3-37 of the PopChart Server User Guide).*

@_WBMP

Available only in PopChart Server Pro and PopChart Server Enterprise.

Syntax `@_WBMP`

Description This command instructs PopChart Server to override the default image type and generate a WBMP image.

@_WIDTH

Syntax `@_WIDTH500`

Description This command specifies the width of the image to be generated. If none is provided, PopChart Server will use the width of the appearance file.

This command takes as its argument an integer, representing the width of the generated image in pixels.

7

PopChart XML

This chapter contains a complete list of all PopChart XML (PCXML) elements and attributes.

Note: *This section assumes you have a basic understanding of XML. If this is not the case, you may want to read "XML Notations and Conventions" on page 1-7.*

At the beginning of this chapter, we have provided Enumerated Set Definitions for PCXML.

The top level element in PCXML is `PopChart`.

These are the second-level elements:

- `NumberFormat`
- `Project`
- `Image`
- `Legend`
- `Textbox`
- `Graph`
- `GraphData`

COMPATIBILITY BETWEEN VERSIONS

Many elements and attributes have been added or renamed since the 4.0 version of PCXML. Changes and additions are indicated in paranthesis next to the appropriate element or attribute.

To maintain backwards compatibility, PopChart Builder saves appearance files in the PCXML 4.0 format by default. Unless you change this behavior, you will be unable to take advantage of many of the latest PCXML features.

You can change the PCXML format that PopChart Builder uses to save its appearance files by selecting Edit > Preferences > Save and checking the Save Using Latest PCXML Format box. However, be aware that you cannot use appearance files saved in the latest PCXML format with previous versions of PopChart Server. The current version of PopChart Server supports previous versions of PCXML.

If you are manually creating your PCXML documents, be sure to specify the version of PCXML you are using by setting the `Version` attribute of the `Chart` tag. This ensures that PopChart Server will properly parse your PCXML.

ENUMERATED SET DEFINITIONS

Set Name	Possible Values
3DEffect	2D, 3D
Anchor	TopLeft, TopCenter, TopRight, MiddleLeft, MiddleCenter, MiddleRight, BottomLeft, BottomCenter, BottomRight
Boolean	True, Yes, False, No
BorderType	None, Thin, DoubleThin,ThinOutsideRegularInside, Regular, RegularOutsideThinInside, Thick
BubbleType	None, 3D, Fill, Outline, OutlineAndFill
BubbleValueAs (4.0.3+)	Area, Width
CategoryLabelAdjustment	AsNeeded, Always
Color	Black, Blue, Cyan, Darkgray, Gray, Green, Lightgray, Magenta, Orange, Pink, Red, White, Yellow, {#rrggbb}, {#rrggbbaa}
DataImportMethod	Replace, Append, LinkByRow, LinkByColumn, AddFilters
DDHighlightMode	BoldOutline, Outline, ColorChange
DivideBy	Auto, Unchanged, Percentage, Thousands, Millions, Billions, Trillions
DLBackgroundType	Transparent, Color
DLPosition (4.0 only)	InsideTop, OutsideTop, InsideBottom, Inside, Outside, OutsideWithLeader, OutsideWithLeaderOnSides
DLPosition (4.0.1+)	Inside, InsideLeft, Inside Right, InsideTop, InsideBottom, Out, OutsideTop, OutsideRight, OutsideWithLeader, OutsideWithLeaderOnSides, Above, Below
FontName	Courier, Helvetica, TimesRoman, {OtherName}
FontStyle	Plain, Bold, Italic, BoldItalic/ItalicBold
Gauge LabelPosition	LeftRight, TopBottom
Gauge LabelSizing	Automatically, Manually
GradientType	None, LeftRight, RightLeft, TopBottom, BottomTop
GraphSubType	Basic, Horizontal, Stacked, Horizontal Stacked, Scatter, HighLowOpenClose, CandleStick, Combo, Dual Y, Bulb, LED Bar, Filled Bar, 3D Bulb
GraphType	Bar, Area, Line, XY, Pie, Stock, Line Bar, Time, Radar, Pareto, Gauge

HJustification	Left, Center, Right
ImageType (4.0.1+)	PopChartLossLess,JPEG
MarkerPosition	X, Y, Value, SValue, Time, Y2 (4.0.1+)
MarkerType	Range, Line
NegativeDisplayType	MinusSign, Paranthesis, AbsoluteValue
NoteLayout	Default, OutsideInVertical, InsideOutVertical
NumberFormat	General, Currency, Percent, Date, HourM, HourMS
OutlineType	None, Color, Beveled, Darker, Lighter
OutputType	FlashWithFallback, SVGWithFallback, Flash, SVG, GIF
PercentageType	None, Category, Total
ScalePosition	Left, Right, Bottom, Top
SeriesTextType (4.0 only)	TextOnly, TextAndValue, TextOutValueIn, TextAboveVOG, TextBelowVOG, TextAVAbove, TextAVBelow, NotSet
SeriesTextType (4.0.1+)	ValueOnly, NameOnly, NameAndValue, NameOutsideValueInside, NameAboveValueOnGraph, NameBelowValueOnGraph, NameAndValueAbove, NameAndValueBelow
ShadowType	None, 1, 2, 3, 4, 5, 6
SymbolType	None, SmallSquare, Dot, Square, SquareOutliine, Circle, CircleOutline, Triangle, TriangelOutline, Plus, PlusOutline, Diamond, DiamondOutline
TickDirection	Outside, Inside, Cross
TickSize	Small, Medium, Large
ValueScaleSizing	Automatically, Manually
ValueScaleType	Line, Bar, Left, Right
WeekDay	Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday

POPCHART

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
PopChart	1			Yes	
		Version	String	Yes	
		BGColor	None/Color	No	White
		BorderType	BorderType	No	Thin
		BorderColor	Color	No	Black
		AutoResize	Boolean	No	True
		FitInBounds	Boolean	No	True
		Antialias	Boolean	No	False
		Width	Short	No	540
		Height	Short	No	330
		CollisionProtection	Boolean	No	True
		BGGradientColor	Color	No	Black
		GradientType	GradientType	No	None

NUMBERFORMAT

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
NumberFormat	2			No	
		CurrencySymbol	String	No	$
		CurrencySymbolBefore	Boolean	No	True
		DecimalSeparator	Character	No	.
		ThousandsCharacter	Character	No	,
		AbbreviateString	String	No	kmbt

PROJECT

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
Project	2			No	
		OutputType	OutputType	No	Flash
		OutputDLink	Boolean	No	True
		HTMLHeight	Short	No	0
		HTMLWidth	Short	No	0
		HTMLSameAsDesignSize	Boolean	No	True
		URLPrefix	String	No	

IMAGE

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
Image	2			No	
		Name	String	Yes	
		Top	Integer	Yes	
		Left	Integer	Yes	
		Height	Integer	Yes	
		Width	Integer	Yes	
		Anchor	Anchor	No	TopLeft
		Visible	Boolean	No	True
		Zindex (4.0.1+)	Short	No	0

DESCRIPTIVE TEXT

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
DescriptiveText	3			No	
Subelement of Image		SuppressAutoDescription	Boolean	No	False
		Description	String	No	

PROPERTIES

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
Properties	3			No	
Subelement of Image		BorderType	BorderType	No	None
		BorderColor	Color	No	Black
		KeepProportions	Boolean	No	True

POPUPTEXT

Element Tag	Level	Attribute Name		Variable Type	Required	Default Value
PopupText	3				No	
Subelement of Image		HJustification		HJustification	No	Left
		AutoWidth		Boolean	No	True
		AutoHeight		Boolean	No	True
		FillOn		Boolean	No	True
		BorderType		BorderType	No	Thin
		WrapWidth		Disabled/Short	No	Disabled
		TopMargin		Short	No	2
		BottomMargin		Short	No	2
		LeftMargin		Short	No	3
		RightMargin		Short	No	3
		FillColor		Color	No	#ffffc8
		BorderColor		Color	No	Black
		Font			No	
			Name	FontName	No	Helvetica
			Size	Short	No	10
			Style	FontStyle	No	Plain
			Color	Color	No	Black

DRILLDOWN

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
Drilldown	3			No	
Subelement of Image		Color	Color	No	Blue
		HighlightMode	DDHighlightMode	No	BoldOutline

DATA

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
Data	3			Yes	
Subelement of Image		NativeWidth	Integer	Yes	
		NativeHeight	Integer	Yes	
		Type (4.0.1+)	ImageType	No	
		DrilldownURL	String	No	
		Target	String	No	
		Popup	String	No	
		AlphaValue	Short	No	100

LEGEND

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
Legend	3 (2)			No	
Either Subelement of Graph or		Name	String	Yes	
Can be used independently for a previously created Graph.		GraphName	String	Sometimes	
		Top	Integer	Yes	
		Left	Integer	Yes	
		Height	Integer	No	110
		Width	Integer	No	80
		Anchor	Anchor	No	TopLeft
		Visible	Boolean	No	True
		Zindex (4.0.1+)	Short	No	0

PROPERTIES

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
Properties	4 (3)				
Subelement of Legend		AutoSizeMarkers	Boolean	No	True
		MarkerWidth	Short	No	9
		MarkerHeight	Short	No	9
		AbbreviateOn	Boolean	No	False
		AbbreviateIfOver	Byte	No	20
		LabelHighlightColor (4.0 only)	Color	No	Blue
		ShadowColor	Color	No	#646464
		ShadowType	ShadowType	No	None
		BorderType	BorderType	No	None
		BorderColor	Color	No	Black

	UseTransparentFill		Boolean	No	True
	FillColor		Color	No	White
	TopMargin		Short	No	0
	BottomMargin		Short	No	0
	LeftMargin		Short	No	0
	RightMargin		Short	No	0
	LayoutItems		LeftRight/TopBottom	No	LeftRight
	GrowVerticalMaxHeight		Disabled/Short	No	Disabled
	GrowHorizontalMaxWidth		Disabled/Short	No	Disabled
	NumberOfColumns		Disabled/Short	No	Disabled
	MinimumFontSize		Short	No	8
	TruncateDownTo		Short	No	20
	Font			No	
		Name	FontName	No	Helvetica
		Size	Short	No	12
		Style	FontStyle	No	Plain
		Color	Color	No	Black

DRILLDOWN

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
Drilldown (4.0.1+)	4 (3)			No	
Subelement of Legend		Color	Color	No	Blue

TEXTBOX

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
Textbox	2			No	
		Name	String	Yes	
		Top	Integer	Yes	
		Left	Integer	Yes	
		Height	Integer	No	19
		Width	Integer	No	76
		Anchor	Anchor	No	TopLeft
		Visible	Boolean	No	True
		Zindex (4.0.1+)	Short	No	0

DESCRIPTIVE TEXT

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
DescriptiveText	3				
Subelement of Textbox		SuppressAutoDescription	Boolean	No	False
		Description	String	No	

PROPERTIES

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
Properties	3				
Subelement of Textbox		AutoWidth	Boolean	No	True
		AutoHeight	Boolean	No	True
		FillOn	Boolean	No	False
		ShadowType	ShadowType	No	None

	HJustification		HJustification	No	Center
	BorderType		BorderType	No	Thin
	WrapWidth		Short	No	600
	TopMargin		Short	No	2
	BottomMargin		Short	No	2
	LeftMargin		Short	No	3
	RightMargin		Short	No	3
	TransparentFill		Boolean	No	False
	FillColor		Color	No	#ffffc8
	BorderColor		Color	No	Black
	ShadowColor		Color	No	#646464
	LabelHighlightColor (4.0 only)		Color	No	Blue
	Rotation		Short	No	0
	IsNoteTextbox		Boolean	No	False
	Font			No	
		Name	FontName	No	Helvetica
		Size	Short	No	10
		Style	FontStyle	No	Plain
		Color	Color	No	Black

DRILLDOWN

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
Drilldown (4.0.1+)	4 (3)			No	
Subelement of Textbox		Color	Color	No	Blue

TEXT

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
Text	3			Yes	
Subelement of Textbox		DrilldownURL	String	No	
		Target	String	No	

GRAPH

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
Graph	2				
		Name	String	Yes	
		Top	Integer	Yes	
		Left	Integer	Yes	
		Height	Integer	No	222 (Non-Gauge) 50 (Gauge)
		Width	Integer	No	300 (Non-Gauge) 350 (Gauge)
		Anchor	Anchor	No	TopLeft
		Visible	Boolean	No	True
		Zindex (4.0.1+)	Short	No	0
		Type	GraphType	No	Bar
		SubType	GraphSubType	No	Basic

PROPERTIES

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
Properties	3			No	
Subelement of Graph		Effect	3DEffect	No	3D
Non-Gauge Only		EffectPercentX	Float	No	8
		EffectPercentY	Float	No	8
		SingleSeriesMulticolor	Boolean	No	False
		TextTicksInBounds (4.0 only)	Boolean	No	False
		KeepTextTicksInBounds (4.0.1+)	Boolean	No	True
		EliminateWhiteSpace	Boolean	No	False
		OutlineType	OutlineType	No	None

	OutlineColor	Color	No	Black
	ShowPieGap	Boolean	No	True
	PieCategory	Short	No	1
	ComboLineNumber	Short	No	1
	Show3DLines	Boolean	No	True
	PercentSpaceBetweenBars	Float	No	0
	PercentBarWidth	Float	No	80
	PercentBarDepth	Float	No	60
	FloatingBars	Boolean	No	False
	YearQuartersPreferred	Boolean	No	False
	LimitTimeMajorIncs	Boolean	No	False
	BubblePercentSize (4.0.3+)	Integer	No	100
	BubbleValueAs (4.0.3+)	BubbleValueAs	No	Width
Gauges Only	SizeLabels	Gauge LabelSizing	No	Automatically
	BGColor	Color	No	Gray
	ShowBorder	Boolean	No	False
	ShowValueAsPercentage	Boolean	No	False
	ShowColorRange	Boolean	No	True
	ShowLabelsOn	Gauge LabelPosition	No	LeftRight

DESCRIPTIVETEXT

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
DescriptiveText	3			No	
Subelement of Graph		SuppressAutoDescription	Boolean	No	False
		Description	String	No	
		SuppressValues	String	No	

VALUES

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
Values	3			No	
Subelement of Graph		Value	Double	No	79
Gauges Only		Text	String	No	“Label”
		DrilldownURL	String	No	
		Target	String	No	
		Popup	String	No	

DRILLDOWN

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
Drilldown	3				
Subelement of Graph		HighlightMode	DDHighlightMode	No	BoldOutline
		MetaString	String	No	
		Color	Color	No	Blue

GAUGELABEL

Element Tag	Level	Attribute Name		Variable Type	Required	Default Value
GaugeLabel	3					
Subelement of Graph		Position		GaugePosition	No	Middle
Gauges Only		Visible		Boolean	No	True
		ManualWidth		Short	No	86
		Font				
			Name	FontName	No	Helvetica
			Size	Short	No	18

		Style	FontStyle	No	Plain
		Color	Color	No	White

VALUELABEL

Element Tag	Level	Attribute Name		Variable Type	Required	Default Value
ValueLabel	3				No	
Subelement of Graph		Position		GaugePosition	No	Middle
Gauges Only		Visible		Boolean	No	True
		ManualWidth		Short	No	86
		Font				
			Name	FontName	No	Helvetica
			Size	Short	No	18
			Style	FontStyle	No	Plain
			Color	Color	No	White

COLORRANGE

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
ColorRange	3			No	
Subelement of Graph		Name	String	Yes	
Gauges Only		Color	Color	Yes	
		Minimum	Double	Yes	
		Maximum	Double	Yes	
		NoColorRanges	Boolean	No	False

DATALABELS

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
DataLabels	3			No	
Subelement of Graph		Visible	Boolean	No	True
Non-Gauge Only		Position	DLPosition	No	OutsideTop
		ShowOnRollover	Boolean	No	True
		ShowSeriesTextType	SeriesTextType	No	NotSet
		TransparentBackground	Boolean	No	False
		BGColor	Color	No	#ccffff
		BorderType	BorderType	No	Thin
		BorderColor	Color	No	Black
		AlwaysShowDecimalPlaces	Boolean	No	False
		NumberType	NumberFormat	No	General
		UseThousandsSeparator	Boolean	No	True
		DecimalPlaces	Byte	No	2
		NegativeDisplayType	NegativeDisplayType	No	MinusSign
		ShowAsPercentageType	PercentageType	No	False
		ShowBarZeroValues (4.0.3+)	Boolean	No	False
		ShowOneValueOnStacked	Boolean	No	False
		ShowPercentDecimalPlaces	Boolean	No	False
		PercentDecimalPlaces	Byte	No	2
		WrapWidth	Short	No	400
		FormatString	String	No	%_Value
		TimeFormatString	String	No	%m/%d/%y %T
		Font		No	
		Name	FontName	No	Helvetica

		Size	Short	No	10
		Style	FontStyle	No	Plain
		Color	Color	No	Black

DATEINPUTFORMAT

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
DateInputFormat	3			No	
Subelement of Graph (Time Only)		%m/%d/%Y			

DATEMAJORTICKS

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
DateMajorTicks	3			No	
Subelement of Graph (Time Only)		Year (4.0.1+)	String	No	%Y
		Quarter (4.0.1+)	String	No	Q%Q %y
		Month (4.0.1+)	String	No	%b
		Day (4.0.1+)	String	No	%a
		Hour (4.0.1+)	String	No	%I%p
		Minute (4.0.1+)	String	No	%I:%M%p
		%Y^Q%Q %y^%b^%a^%I%p^%I:%M%p (4.0 only)			

DATEMINORTICKS

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
DateMinorTicks	3			No	
Subelement of Graph (Time Only)		Quarter (4.0.1+)	String	No	Q%Q
		Month (4.0.1+)	String	No	%b

	Day (4.0.1+)	String	No	%d
	Hour (4.0.1+)	String	No	%I%p
	Minute (4.0.1+)	String	No	%M
	Q%Q^%b^%d^%I%p^%M (4.0 only)			

MONTHNAMES

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
MonthNames	3			No	
Subelement of Graph (Time Only)	Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec, (4.0 only)				
	Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec (4.0.1+)				

DAYNAMES

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
DayNames	3			No	
Subelement of Graph (Time Only)	Sun,Mon,Tue,Wed,Thu,Fri,Sat, (4.0 only)				
	Sun,Mon,Tue,Wed,Thu,Fri,Sat (4.0.1+)				

FIRSTDAYOFWEEK

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
FirstDayOfWeek	3		WeekDay	No	
Subelement of Graph (Time Only)	Sunday				

GRIDCOLORS

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
GridColors	3			No	
Subelement of Graphs		BackColorTransparent	Boolean	No	True
Non-Gauge Only		BackColor	Color	No	White
		BottomColorTransparent	Boolean	No	True
		BottomColor	Color	No	White
		SideColorTransparent	Boolean	No	True
		SideColor	Color	No	White

COLORPALETTE

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
ColorPalette	3			No	
Subelement of Graphs		Name (4.0.1+)	String	No	Default
Non-Gauge Only		Color1	Color	No	#009933
		Color2	Color	No	#9900cc
		Color3	Color	No	#ff9900
		Color4	Color	No	#993333
		Color5	Color	No	#1e1edc
		Color6	Color	No	#666600
		Color7	Color	No	#ff6600
		Color8	Color	No	#0066cc
		Color9	Color	No	#cccc00
		Color10	Color	No	#006e00
		Color11	Color	No	#6699ff
		Color12	Color	No	#c81e1e

	Color13	Color	No	#cc99ff
	Color14	Color	No	#875400
	Color15	Color	No	#dc0a82
	Color16	Color	No	#005500

POPUPTEXT

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
PopupText	3			No	
Subelement of Graph		HJustification	HJustification	No	Left
		AutoWidth	Boolean	No	True
		AutoHeight	Boolean	No	True
		FillOn	Boolean	No	True
		BorderType	BorderType	No	Thin
		WrapWidth	Disabled / Short	No	Disabled
		TopMargin	Short	No	2
		BottomMargin	Short	No	2
		LeftMargin	Short	No	3
		RightMargin	Short	No	3
		FillColor	Color	No	#ffffc8
		BorderColor	Color	No	Black
		Font		No	
		Name	FontName	No	Helvetica
		Size	Short	No	10
		Style	FontStyle	No	Plain
		Color	Color	No	Black

NOTEDEFINITION

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
NoteDefinition	3			No	
Subelement of Graphs		HJustification	HJustification	No	Left
Non-Gauge Only		AutoWidth	Boolean	No	True
		AutoHeight	Boolean	No	True
		FillOn	Boolean	No	True
		BorderType	BorderType	No	Thin
		WrapWidth	Disabled / Short	No	Disabled
		TopMargin	Short	No	2
		BottomMargin	Short	No	2
		LeftMargin	Short	No	3
		RightMargin	Short	No	3
		FillColor	Color	No	#ffffc8
		BorderColor	Color	No	Black
		LeaderColor	Color	No	Black
		LeaderWidth	Byte	No	1
		NoteLayout	NoteLayout	No	Default
		Font		No	
		Name	FontName	No	Helvetica
		Size	Short	No	10
		Style	FontStyle	No	Plain
		Color	Color	No	Black

SERIESDEFINITION

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
SeriesDefinition	3			No	
Subelement of Graphs		Number	Short	Yes	
Non-Gauge Only		Scale	ValueScaleType	No	
		Color	Color	Yes	
		DownColor (4.0.1+)	Color	No	
		FillColor	Color	No	
		FillDownColor (4.0.1+)	Color	No	
		LineColor	Color	No	
		OutlineDownColor (4.0.1+)	Color	No	
		OutlineColor	Color	No	
		OutlineUpColor (4.0.1+)	Color	No	
		UpColor (4.0.1+)	Color	No	
		SymbolColor	Color	No	
		FillUpColor (4.0.1+)	Color	No	
		BubbleType	BubbleType	No	3D
		BubbleColor (4.0.1+)	Color	No	
		LineWidth	Float	No	4.0
		SymbolType	SymbolType	No	SmallSquare
		ShowArea	Boolean	No	False
		FormatString	String	No	

PIELABEL

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
PieLabel	3				

	Visible		Boolean	No	True
	Font			No	
		Name	FontName	No	Courier
		Size	Short	No	12
		Style	FontStyle	No	Plain
		Color	Color	No	Black

SCALE

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
Scale	3			No	
Subelement of Graph		Visible	Boolean	No	True
Gauges Only		Minimum	Double	No	0
		Maximum	Double	No	100
		Color	Color	No	White
		BGColor	Color	No	Black
		SizeManually	Boolean	No	True
		ManualMaxHeight	Integer	No	40
		ShowMajorTicks	Boolean	No	True
		NumberMajorTicks	Short	No	5
		MajorTickColor	Color	No	White
		ShowMinorTicks	Boolean	No	True
		NumberMinorTicks	Short	No	4
		MinorTickColor	Color	No	White

VALUESCALE

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
ValueScale	3			No	
Subelement of Graphs		Position	ScalePosition	No	Left
Non-Gauge Only		Type	ValueScaleType	No	Line (Left)
		SetScaleValues	ValueScaleSizing	No	Automatically
		UseLogScale	Boolean	No	False
		SyncWithPrimaryScale	Boolean	No	False
		ShowAbbreviationCharacter	Boolean	No	False
		DivideScaleBy	DivideBy	No	Unchanged
		AutoAbbreviate	Boolean	No	False
		AlwaysShowDecimalPlaces	Boolean	No	False
		DecimalPlaces	Byte	No	2
		NumberFormat	NumberFormat	No	General
		ShowThousandsSeparator	Boolean	No	True
		NegativeDisplayType	NegativeDisplayType	No	MinusSign
		RotateLabels	Short	No	0
		PercentUnderMinValue	Short	No	10
		SetScaleBaseToZero	Boolean	No	False
		MaximumMajorTicks	Short	No	6
		NumberMinorTicks	Short	No	1
		PercentOverMaxValue	Short	No	0
		NumberMajorTicks	Short	No	6
		NumberMinorTicks	Short	No	1
		MinimumValue (4.0 only)	Double	No	0.0
		MaximumValue (4.0 only)	Double	No	100.0

	Minimum (4.0.1+)		Double	No	0.0
	Maximum (4.0.1+)		Double	No	100.0
	DisplayScaleLabels		Boolean	No	True
	MinorDisplayScaleLabels		Boolean	No	True
	Font			No	
		Name	FontName	No	Helvetica
		Size	Short	No	12
		Style	FontStyle	No	Plain
		Color	Color	No	Black
	MinorFont			No	
		Name	FontName	No	Helvetica
		Size	Short	No	10
		Style	FontStyle	No	Plain
		Color	Color	No	Black
	MajorTick			No	
		Visible	Boolean	No	True
		LineWidth	Float	No	1.0
		Color	Color	No	Black
		Direction	TickDirection	No	Outside
		Length (4.0 only)	TickSize	No	Medium
		Size (4.0.1+)	TickSize	No	Medium
	MinorTick			No	
		Visible	Boolean	No	True
		LineWidth	Float	No	1.0
		Color	Color	No	#cccccc
		Direction	TickDirection	No	Out
		Length (4.0 only)	TickSize	No	Medium

		Size (4.0.1+)	TickSize	No	Medium
	LowOuterLine			No	
		Visible	Boolean	No	True
		LineWidth	Float	No	1.0
		Color	Color	No	Black
	HighOuterLine			No	
		Visible	Boolean	No	True
		LineWidth	Float	No	1.0
		Color	Color	No	Black
	ZeroLine			No	
		Visible	Boolean	No	True
		LineWidth	Float	No	1.0
		Color	Color	No	Black
	MajorGrid			No	
		Visible	Boolean	No	True
		LineWidth	Float	No	1.0
		Color	Color	No	Black
	MinorGrid			No	
		Visible	Boolean	No	True
		LineWidth	Float	No	1.0
		Color	Color	No	#cccccc

CATEGORYSCALE

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
CategoryScale	3			No	
Subelement of Graphs		PerformLabelAdjustments	Category LabelAdjustment	No	AsNeeded

Non-Gauge Only	LimitLabelLength		Boolean	No	False
	MaxLengthRotatedText		Short	No	15
	MaxLengthHorizontalText		Short	No	20
	FirstLabelToBeDisplayed		Short	No	1
	WrapTextToMaxLines		Short	No	2
	ShrinkFontSize		Short	No	9
	StaggerLabels		Boolean	No	True
	RotateLabels		Short	No	60
	SkipLabelsAtATime		Disabled / Short	No	1
	WrapWidthPixels		Short	No	200
	DisplayScaleLabels		Boolean	No	True
	Font			No	
		Name	FontName	No	Helvetica
		Size	Short	No	12
		Style	FontStyle	No	Plain
		Color	Color	No	Black
	MinorFont			No	
		Name	FontName	No	Helvetica
		Size	Short	No	10
		Style	FontStyle	No	Plain
		Color	Color	No	Black
	MajorTick			No	
		Visible	Boolean	No	True
		LineWidth	Float	No	1.0
		Color	Color	No	Black
		Direction	TickDirection	No	Outside
		Length (4.0 only)	TickSize	No	Medium

		Size (4.0.1+)	TickSize	No	Medium
	MinorTick			No	
		Visible	Boolean	No	True
		LineWidth	Float	No	1.0
		Color	Color	No	Black
		Direction	TickDirection	No	Out
		Length (4.0 only)	TickSize	No	Medium
		Size (4.0.1+)	TickSize	No	Medium
	LowOuterLine			No	
		Visible	Boolean	No	True
		LineWidth	Float	No	1.0
		Color	Color	No	Black
	HighOuterLine			No	
		Visible	Boolean	No	True
		LineWidth	Float	No	1.0
		Color	Color	No	Black
	ZeroLine			No	
		Visible	Boolean	No	True
		LineWidth	Float	No	1.0
		Color	Color	No	Black
	MajorGrid			No	
		Visible	Boolean	No	True
		LineWidth	Float	No	1.0
		Color	Color	No	Black
	MinorGrid			No	
		Visible	Boolean	No	True

		LineWidth	Float	No	1.0
		Color	Color	No	Black

SCALEMARKER

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
ScaleMarker	3			No	
Subelement of Graphs		Position	MarkerPosition	Yes	
Non-Gauge Only		Type	MarkerType	No	Line
		Low	String	No	
		High	String	No	
		LineWidth	Double	No	
		Value	Double	No	
		Color	Color	No	

GRAPHDATA

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
GraphData	2				
		Name	String	Yes	
		Method	DataImportMethod	Yes	Replace
		Source	String	No	
		SeriesIn	Rows/Columns	No	
		SortData	Boolean	No	

CATEGORIES

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
Categories	3				
Subelement of GraphData					

CATEGORY

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
Category	4				
Subelement of Categories		Name	String	No	
		DrilldownURL	String	No	
		Target	String	No	
		Popup	String	No	
		Note	String	No	
		NoteTarget	String	No	
		Color (4.0.1+)	Color	Yes	
		FillColor (4.0.1+)	Color	No	
		FillDownColor (4.0.1+)	Color	No	
		BubbleColor (4.0.1+)	Color	No	

	LineColor (4.0.1+)	Color	No	
	OutlineDownColor (4.0.1+)	Color	Yes	
	DownColor (4.0.1+)	Color	No	
	OutlineColor (4.0.1+)	Color	No	
	OutlineUpColor (4.0.1+)	Color	No	
	UpColor (4.0.1+)	Color	No	
	SymbolColor (4.0.1+)	Color	Yes	
	FillUpColor (4.0.1+)	Color	No	
	SymbolType (4.0.1+)	SymbolType	No	

SERIES

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
Series	3				
Subelement of GraphData		Name	String	No	
		FCValue	String	No	
		DrilldownURL	String	No	
		Target	String	No	
		Popup	String	No	
		Note	String	No	
		NoteTarget	String	No	
		Color (4.0.1+)	Color	Yes	
		FillColor (4.0.1+)	Color	No	
		FillDownColor (4.0.1+)	Color	No	
		BubbleColor (4.0.1+)	Color	No	
		LineColor (4.0.1+)	Color	No	
		OutlineDownColor (4.0.1+)	Color	Yes	
		DownColor (4.0.1+)	Color	No	

	OutlineColor (4.0.1+)	Color	No	
	OutlineUpColor (4.0.1+)	Color	No	
	UpColor (4.0.1+)	Color	No	
	SymbolColor (4.0.1+)	Color	Yes	
	FillUpColor (4.0.1+)	Color	No	
	SymbolType (4.0.1+)	SymbolType	No	

APPENDTOSERIES

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
AppendToSeries	3				
Subelement of GraphData					

DATA

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
Data	4				
Subelement of Series or AppendToSeries		Name	String	No	
		Value	Float	No	
		X	Float	No	
		Y	Float	No	
		Bubble	Float	No	
		Date	Date	No	
		High	Float	No	
		Low	Float	No	
		Open	Float	No	
		Close	Float	No	
		DrilldownURL	String	No	
		Target	String	No	

	Popup	String	No	
	Note	String	No	
	NoteTarget	String	No	
	Color	Color	Yes	
	FillColor	Color	No	
	FillDownColor (4.0.1+)	Color	No	
	BubbleColor (4.0.1+)	Color	No	
	LineColor	Color	No	
	OutlineDownColor (4.0.1+)	Color	No	
	DownColor (4.0.1+)	Color	No	
	OutlineColor (4.0.1+)	Color	No	
	OutlineUpColor (4.0.1+)	Color	No	
	UpColor (4.0.1+)	Color	No	
	SymbolColor	Color	No	
	FillUpColor (4.0.1+)	Color	No	
	SymbolType	SymbolType	No	

ROW

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
Row	3				
Subelement of GraphData					

APPENDTOROW

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
AppendToRow	3				
Subelement of GraphData					

CELL

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
Cell	4				
Subelement of Row or AppendToRow		Column	Short	No	
(Actual data goes between begin and end Cell element tags)		Row	Short	No	
		DrilldownURL	String	No	
		Target	String	No	
		Popup	String	No	
		Note	String	No	
		NoteTarget	String	No	
		Color	Color	Yes	
		FillColor	Color	No	
		FillDownColor (4.0.1+)	Color	No	
		BubbleColor (4.0.1+)	Color	No	
		LineColor	Color	No	
		OutlineDownColor (4.0.1+)	Color	No	
		DownColor (4.0.1+)	Color	No	
		OutlineColor (4.0.1+)	Color	No	
		OutlineUpColor (4.0.1+)	Color	No	
		UpColor (4.0.1+)	Color	No	
		SymbolColor	Color	No	
		FillUpColor (4.0.1+)	Color	No	
		SymbolType	SymbolType	No	

COLUMNFILTER

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
ColumnFilter	3				
Subelement of GraphData		Column	Short	Yes	
		Enabled	Boolean	No	True

ROWFILTER

Element Tag	Level	Attribute Name	Variable Type	Required	Default Value
RowFilter	3				
Subelement of GraphData		Row	Short	Yes	
		Enabled	Boolean	No	True

8

DESCRIPTIVE TEXT SETTINGS

This chapter documents the dlink.xml file, located in the config folder, which controls the formatting of descriptive text. For more information about descriptive text, refer to Chapter 8, "Displaying 508 Compliant Descriptive Text," in the *PopChart Server User Guide*.

Note: *This section assumes you have a basic understanding of XML. If this is not the case, you may want to read "XML Notations and Conventions" on page 1-7.*

CHANGING THE DLINK.XML DESCRIPTIVE TEXT TEMPLATE

A more powerful way of formatting your descriptive text is to customize the descriptive text template file. This single XML file controls how the descriptive text is generated. To correctly modify this file, you will probably need to know a little about XML (eXtensible Markup Language).

Inside the config folder of the PopChart Server root directory you will find the dlink.xml file, which controls all the formatting for descriptive text. Because it is so complicated, we have not fully documented it here. However, we have given you some pointers on how to customize it. You may also want to experiment with it on your own.

You can use any text editor to modify the dlink.xml file. However, it is much easier to use an XML file editor to make changes. Microsoft has a beta version of its XML Notepad available at: http://msdn.microsoft.com/xml/notepad/intro.asp. Other XML editors are available from a variety of sources.

Warning: Backup your dlink.xml file before making any changes.

The top element in the file is `DLinkTemplates`. There should be only one of these elements in the file. Under `DLinkTemplates` are the following tags and elements:

Version The D-link version. This should never be changed.

LanguageSettings You can change what words will be used for numbers, such as thousands or millions. You should not change the `LangCode` attribute unless you are creating a new language section (refer to "Adding Support for Additional Languages" on page 8-7).

AppearanceAssociations The most important element(s). There can be several of these elements, each associated with a single appearance file, or with a directory of appearance files. These elements describe the descriptive text template for any appearance files that are associated with the element.

The `Filename` attribute determines which appearance files are associated with an `AppearanceAssociation` element. When PopChart Server tries to match an appearance file to an `AppearanceAssociation`, it first tries to match the appearance file's name exactly with the `Filename` attribute of an `AppearanceAssociation`. If it cannot do that, it looks for the `AppearanceAssociation` element whose `Filename` most closely matches the path to the appearance file. if there are no matching paths, the first `AppearanceAssociation` element (the one whose `Filename` tag is `Default`) will be used.

For example, consider the following table:

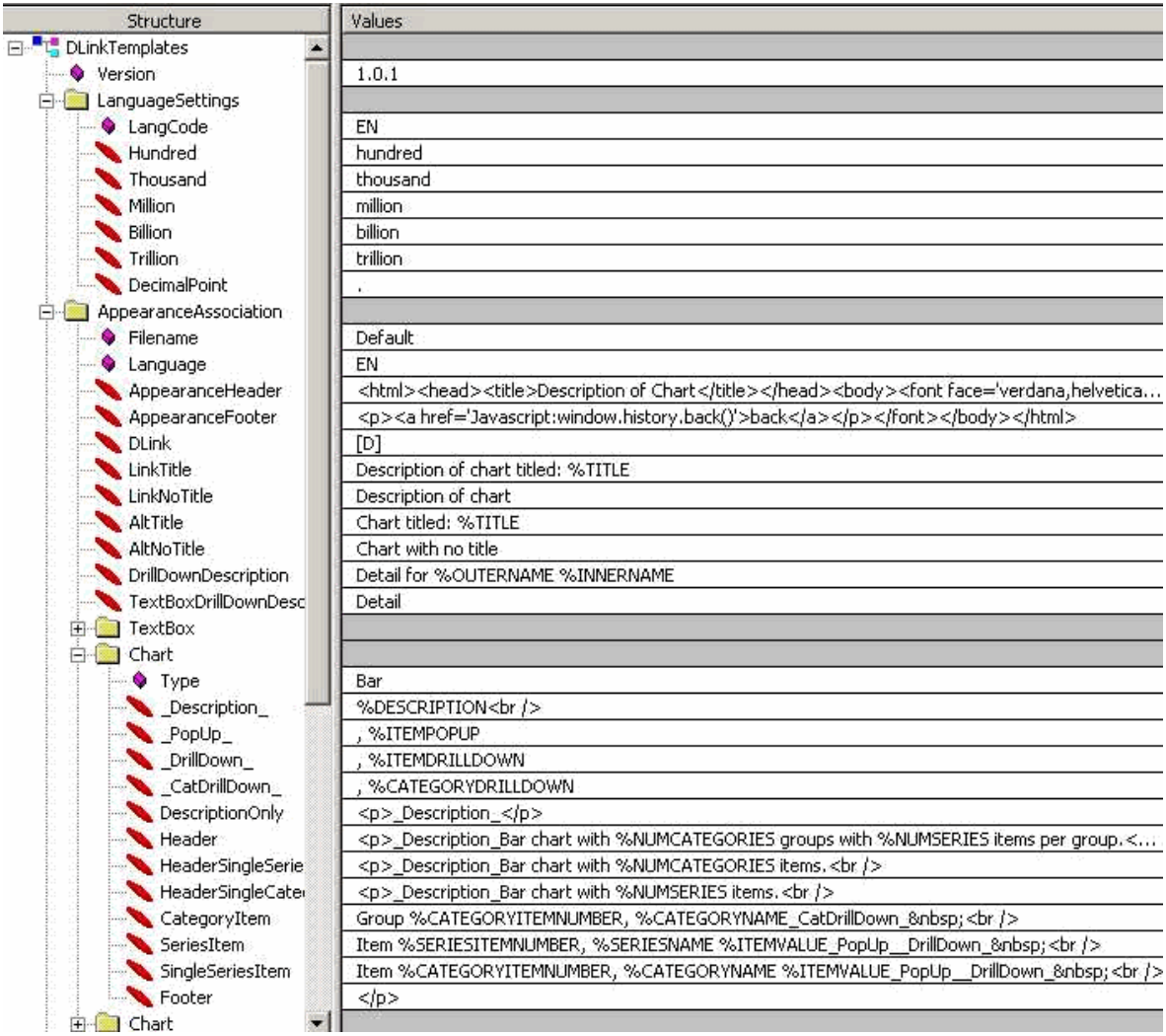
Appearance File	Resolved Appearance Association Filename	Notes
apfiles/bar.pcxml	*Default*	No exact match on the filename and no match on apfiles path.
apfiles/examples.pcxml	*apfiles/examples.pcxml*	Exact match found.
apfiles/demo/bar.pcxml	*apfiles/demo*	No exact match, but the directory apfiles/demo matches.
apfiles/demo/test/bar.pcxml	*apfiles/demo/test*	No exact match, but there are two directory matches: apfiles/demo, and apfiles/demo/test. The latter is a more exact match.

Note: *The `Filename` attribute is case sensitive.*

The `AppearanceAssociation` element contains many subelements that you may want to modify, including `AppearanceHeader` and `AppearanceFooter` (which control the HTML header and footer of the descriptive text file), `Dlink` (which specifies the format of the link that appears next to the PopChart image), and `DrillDownDescription` (which specifies the format for drill-down links).

The most important subelements, though, are the `TextBox` and `Chart` subelements. There is one `TextBox` element, which describes the appearance of text box descriptive text, and many `Chart` elements, each of which describes the appearance of descriptive text for the graph type listed in its `Type` tag. Both of these elements are described in the next section.

Here is an example of the DLink.xml file.



Structure	Values
DLinkTemplates	
Version	1.0.1
LanguageSettings	
LangCode	EN
Hundred	hundred
Thousand	thousand
Million	million
Billion	billion
Trillion	trillion
DecimalPoint	.
AppearanceAssociation	
Filename	Default
Language	EN
AppearanceHeader	<html><head><title>Description of Chart</title></head><body><font face='verdana,helvetica...
AppearanceFooter	back </body></html>
DLink	[D]
LinkTitle	Description of chart titled: %TITLE
LinkNoTitle	Description of chart
AltTitle	Chart titled: %TITLE
AltNoTitle	Chart with no title
DrillDownDescription	Detail for %OUTERNAME %INNERNAME
TextBoxDrillDownDesc	Detail
TextBox	
Chart	
Type	Bar
Description	%DESCRIPTION
PopUp	, %ITEMPOPUP
DrillDown	, %ITEMDRILLDOWN
CatDrillDown	, %CATEGORYDRILLDOWN
DescriptionOnly	_Description_
Header	_Description_Bar chart with %NUMCATEGORIES groups with %NUMSERIES items per group.<...
HeaderSingleSerie	_Description_Bar chart with %NUMCATEGORIES items.
HeaderSingleCate	_Description_Bar chart with %NUMSERIES items.
CategoryItem	Group %CATEGORYITEMNUMBER, %CATEGORYNAME_CatDrillDown_
SeriesItem	Item %SERIESITEMNUMBER, %SERIESNAME %ITEMVALUE_PopUp__DrillDown_
SingleSeriesItem	Item %CATEGORYITEMNUMBER, %CATEGORYNAME %ITEMVALUE_PopUp__DrillDown_
Footer	
Chart	

MODIFYING VALUES IN THE CHART AND TEXTBOX ELEMENTS

The tags within the `Textbox` and `Chart` elements are very similar. HTML is used within the values, such as `<p>` for a new paragraph and `<br />` for a line break. Variables, which

are all in upper case letters and start with %, represent settings within the PopChart project. For example, the %SERIESNAME variables represents the name of the series being described.

Note: *You are not limited to putting HTML between the tags. You can put any kind of text between the tags, allowing you to output data in almost any format.*

There are many tags within the Textbox and Chart elements, and you will have to experiment with most of them on your own. Here we explain one of the most important tags, SeriesItem. The SeriesItem tag is used for each item (bar, wedge, plot point, etc....) of a graph. Below is a typical value for a SeriesItem tag.

```
Item %SERIESITEMNUMBER, %SERIESNAME
  %ITEMVALUE_PopUp__DrillDown_ <br />
```

As an item is processed, the text description for that item is generated using the value above as a template. Text from the template is copied to the description, while variables are replaced with the appropriate text.

For example, the descriptive text for a bar in a bar chart is generated this way:

How Descriptive Text for a Bar is Generated

This assumes we are using the value for SeriesItem that is shown above.

1. **The first word--*Item* , with a space-- is copied to the description.**
2. **Next, the variable %SERIESITEMNUMBER is replaced with the number of the bar (first bar in a group is 1, second is 2, etc.).**
3. **A comma and space are copied from the template to the description.**
4. **The %SERIESNAME variable is replaced with the name of the series.**
5. **The space is copied.**
6. **%ITEMVALUE is replaced with the value of this item (i.e. the length of the bar)**
7. **The next two items, _PopUp_ and _DrillDown_ , are macros. These are names of tags within the Chart element. The value of these tags are substituted for _PopUp_ and _DrillDown_. In this example, let us assume that _PopUp_ is set to , %SERIESPOPUP.**
8. **If %SERIESPOPUP, which is the variable for the bar's PopUp text, is empty, then the _PopUp_ macro is replaced with nothing.**
9. **If there is a value for %SERIESPOPUP, then _PopUp_ is replaced with a comma, space, and the value of %SERIESPOPUP.**
10. **After the _PopUp_ and _DrillDown_ macros are processed then the
 is copied (this is HTML for a non-breaking space and a line break, which helps screen readers to pause between items).**

This is a basic description of how to interact with the DLink.xml file. While other elements use different macros, they work basically the same way as described above.

ADDING SUPPORT FOR ADDITIONAL LANGUAGES

If your chart text is in a language other than English, you will probably want the descriptive text to be generated in that same language. The dlink.xml file allows you to customize it for other languages.

The `LanguageSettings` subelement contains information used by all chart types and textboxes for a given language. It has an attribute `LangCode` whose value is the uppercase two letter code for the language. Currently, there is only one language code in this version of dlink.xml (*EN* for English). If you are adding a new language, you will need to:

To add a new language

1. **Open dlink.xml in a text or XML editor.**
2. **Duplicate the `LanguageSettings` element (copy and paste).**
3. **Modify the value of the `LangCode` attribute to be the two-letter code for the desired language. For example, to create a German `LanguageSettings` subtree, you would set the LangCode to *DE*. See ISO 639 for recommended language codes.**
4. **Translate the values of any `LanguageSettings` subelements that need translating (e.g. *Hundred*, *Thousand*).**
5. **Duplicate all of your `ApperanceAssociation` elements.**
6. **Modify the `Language` attribute for your duplicate `AppearanceAssociation` elements to match the two letter code from step 2.**
7. **Translate the text in the each of the duplicate `AppearanceAssociation` elements to the desired language. Be sure to not change any macros or HTML tags.**
8. **Save the file and restart PopChart Server.**
9. **In your PopChart Embedder code, set the language attribute to the `language` code from step 2. (e.g. `language="DE"`). Or, if you use the `@_TEXTDESCRIPTION` server command, place the new language code after the command (e.g. `@_TEXTDESCRIPTIONDE`).**

GENERATING TEXT-ONLY DESCRIPTIVE TEXT

New in PopChart Server 4.0.1

In certain circumstances, it may be desirable to generate descriptive text in a textual format instead of HTML. To do this, simply create a new `AppearanceAssociation` (refer to "AppearanceAssociations" on page 8-2) for your text-only descriptive text.

You will need to set the `TextOutput` attribute of the `AppearanceAssociation` tag to *true*. The example tag below shows how an text-only `AppearanceAssociation` tag would look if it were the default Appearance Association.

```
<AppearanceAssociation Filename="Default"
  Language="EN" TextOutput="true">
```

This instructs PopChart Server not to process HTML entities or encode ascii characters larger than 128, as it would for HTML output. For example, *>* would usually be converted to the greater-than > sign, but would be left alone for text-only output.

This also instructs PopChart Server to convert escaped characters to ascii text. PopChart Server will convert the following escaped characters:

`\n`	linefeed
`\t`	tab
`\b`	backspace
`\r`	carriage return
`\f`	form feed
`\\`	backslash (\ character)

For you convenience, a partially completed `AppearanceAssociation` element has been included at the end of the descriptive text (dlink.xml) template. To make this example useful, you would have to add `Chart` elements for all of the graph types.

9

HTML Table Settings

This chapter documents the table.xml file, located in the config folder, which controls the formatting of HTML data tables. For more information about HTML data tables, refer to "HTML Data Tables" on page 7-29 of the *PopChart Server User Guide*.

Note: *This section assumes you have a basic understanding of XML. If this is not the case, you may want to read "XML Notations and Conventions" on page 1-7.*

The table.xml file is processed by the same code as the dlink.xml file, which controls descriptive text. The two files have very similar structures. The same concepts described for the dlink.xml file, described in Chapter 8, "Descriptive Text Settings," also apply for table.xml.

9 HTML TABLE SETTINGS

10

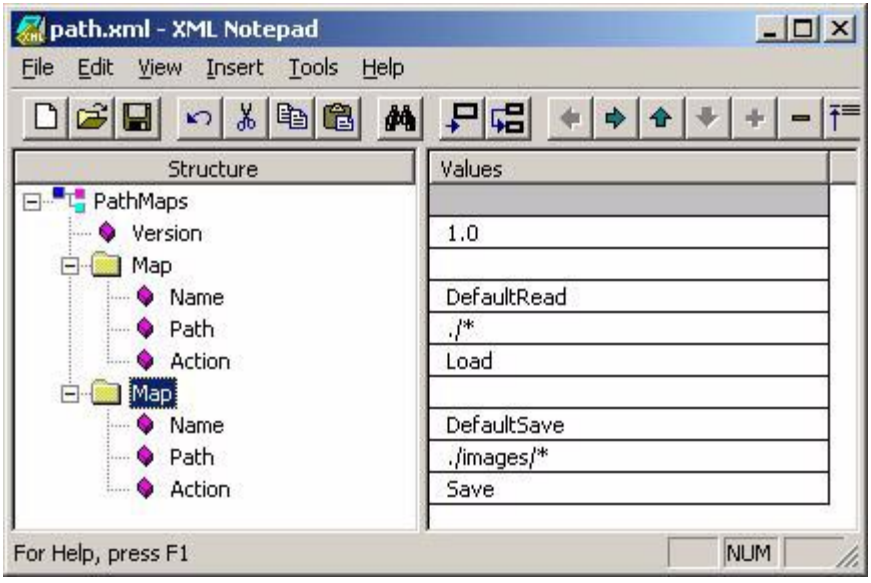
PATH SETTINGS

This chapter documents the path.xml file, located in the config folder, which controls security permissions for PopChart Server. For general information about security permissions text, refer to "Setting Path Permissions" on page 3-37 of the *PopChart Server User Guide*.

Note: *This section assumes you have a basic understanding of XML. If this is not the case, you may want to read "XML Notations and Conventions" on page 1-7.*

PATH.XML DOCUMENT SPECIFICATION

Below are the contents of the default path.xml file.



Or, if you view the file through a text editor, you will see the following:

Example 10.1 Typical path.xml File

```
<PathMaps Version="1.0">
   <Map Name="DefaultRead" Path="./*" Action="Load"/>
   <Map Name="DefaultSave" Path="./images/*" Action="Save"/>
   <Map Name="ValidDomain" Path="127.0.0.1" Action="allowDomain"/>
   <Map Name="ValidDomain" Path="localhost" Action="allowDomain"/>
</PathMaps>
```

The top element of the path.xml file is a `PathMaps`. The attribute in `PathMaps` is `Version`, which at this time can only be set to *1.0*. The `<PathMaps>` tag should not be modified.

Maps ELEMENT

The most important part of the path.xml file is the list of <map> tags, or mappings, each of which specifies a directory, host, or domain that PopChart Image Server has permission to access. Map, which is a subelement of PathMaps, consists of three attributes: Name, Path, and Action.

NAME

This specifies the name for the mapping. Except for the DefaultRead and the DefaultSave names (which specify the default directory for any files that are loaded and any files that are saved, respectively), this attribute has no other significance. In future releases, it may become an optional alias for that path, so you may want to keep that in mind as you name each mapping.

PATH

This specifies the path that this mapping allows access to. You should set the value of this attribute to either a local path, host name, or IP address.

A path can be absolute or relative. A dot slash ./ before the path makes it relative to PopChart Server's document root directory (usually chart_root). You *cannot* use two dots ../ to go up a directory.

Wild cards are allowed. In local paths, they allow PopChart Server to access all of the subdirectories of the specified path. For example, in the DefaultRead mapping of the default path.xml, ./* means that PopChart Server can access any file in the document root directory, or in any of its subdirectories.

In URL, wildcards can specify that access permissions should be given to an entire domain or subnet. For example, *.mycompany.com gives access permissions to every host in the mycompany.com domain. 12.1.*.* gives access permissions to every computer in the 12.1.x.x subnet.

ACTION

This specifies the action that the mapping allows. Currently, there are three mappings:

Load Allows PopChart Server to load files from the path specified by the Path attribute.

Save Allows PopChart Server to save to the path specified by the Path attribute.

AllowDomain Allows PopChart Server to load files from the host, domain, IP Address, or subnet specified by the Path attribute.

ADDING / MODIFYING PERMISSIONS

To add permissions for a directory which will be used for loading files, you simply add another mapping. Adding the following <map> tag allows PopChart Server to load files from the apfiles directory on the local Z drive (Windows Example).

```
<Map Name="myApfiles" Path="Z:\apfiles" Action="Load"/>
```

The following example tag adds saving permission for the /usr/home/myname/images/ directory and all of its subdirectories (UNIX Example).

```
<Map Name="mySaveFiles"
  Path="/usr/home/myname/images/*" Action="Save"/>
```

PopChart Server also supports UNC (Universal Naming Convention) in a Microsoft Network. This allows you to specify a file location on a different machine. For example \\WebServer\MyGraphs\apfiles\test.bin would refer to the file test.bin in the apfiles directory which is located in the share named MyGraphs on the machine WebServer.

To add load permissions from this directory, create the following mapping.

```
<Map Name="myApfiles"
  Path="\\WebServer\MyGraphs\apfiles" Action="Load"/>
```

Finally, PopChart Server also supports loading files from URLs. For instance, you could load a file from http://appserver.mycompany.com. To allow PopChart Server to load files from this host, you would need to add the following mapping.

```
<Map Name="ValidDomain" Path="appserver.mycompany.com"
  Action="allowDomain"/>
```

Alternatively, you could use wildcards to allow PopChart Server to load files from any computer in a domain. For example, to allow any server in the mycompany.com domain, use the following mapping.

```
<Map Name="ValidDomain" Path="*.mycompany.com"
  Action="allowDomain"/>
```

Likewise, you can use wildcards to specify a subnet for PopChart Server to load from. For example, to allow PopChart Server to load anything from the 10.0.x.x subnet, you would add the following mapping.

```
<Map Name="ValidDomain" Path="10.0.*.*"
  Action="allowDomain"/>
```

Note: *PopChart Server can only load files from HTTP Requests (i.e. not FTP support). PopChart Server cannot save files to a URL.*

11

COLOR THEMES

This chapter documents the pccolors.xml file, located in the config folder, which controls color themes for PopChart Builder and PopChart Server.

Note: *This section assumes you have a basic understanding of XML. If this is not the case, you may want to read "XML Notations and Conventions" on page 1-7.*

Color themes are a convenient way to change the look and feel of your graph. Instead of having to change the colors of your bars, lines, symbols, or pie wedges individually, you can change them all at once by selecting a pre-designed set of colors. Color themes do not apply to background, gridline, or text colors.

USING COLOR THEMES

Color themes specify the colors for up to sixteen data series. These colors will be used for any bars, lines, symbols, pie wedges, and filled areas in the series. PopChart 4.0.5 comes with several predefined color themes, including:

- Default
- Standard 16
- Pastel
- Southwest
- Everglades
- Ocean
- Subdued
- Standard Translucent

You can change a color theme for a graph in PopChart Builder by going to the Graph Properties > Colors dialog pane and selecting a new color theme from the Select a Color Theme pull-down menu.

You can change the color theme for a graph in PCScript with the `graph.UseColorPalette()` method. This method accepts as an argument the name of a color theme. For example, to use the *Ocean* color theme, you would add the following PCScript command:

```
graph.useColorPalette(Ocean)
```

You can change the color theme for a graph in PopChart XML by setting the `Name` attribute of the graph's `ColorPalette` subelement to the name of the color theme that you want to use. For example, the following subelement specifies that the graph should use the *Subdued* color theme.

```
<Graph Name="graph"><ColorPalette
  Name="Subdued" /></Graph>
```

CREATING NEW COLOR THEMES

PopChart Server comes with several color themes, but you may find it useful to develop color themes for yourself or for your company. You can add a new color theme simply by modifying an XML file.

Important: *You should backup any files that you modify before modifying them.*

To Create a New Color Theme

1. **Locate your PCColors.xml file, located in the config directory.**

2. **Open the file using an XML or Text editor.**

   You can use any text editor to modify the PCColors.xml file. However, it is much easier to use an XML file editor to make changes. Microsoft has a beta version of its XML Notepad available at: http://msdn.microsoft.com/xml/notepad/download.asp. Other XML editors are available from a variety of sources.

   The file will look something like this:

Example 11.1 **Color Themes XML File**

```
<ChartColorSchemes Version="1.0">
    <Scheme Name="Default">
        <Color1 Red="0" Green="153" Blue="51" />
        <Color2 Red="153" Green="0" Blue="204" />
        <Color3 Red="255" Green="153" Blue="0" />
        <Color4 Red="153" Green="51" Blue="51" />
        <Color5 Red="30" Green="30" Blue="220" />
        <Color6 Red="102" Green="102" Blue="0" />
        <Color7 Red="255" Green="102" Blue="0" />
        <Color8 Red="0" Green="102" Blue="204" />
        <Color9 Red="204" Green="204" Blue="0" />
        <Color10 Red="0" Green="110" Blue="0" />
        <Color11 Red="102" Green="153" Blue="255" />
        <Color12 Red="200" Green="30" Blue="30" />
        <Color13 Red="204" Green="153" Blue="255" />
        <Color14 Red="135" Green="84" Blue="0" />
        <Color15 Red="220" Green="10" Blue="130" />
        <Color16 Red="0" Green="85" Blue="0" />
    </Scheme>
    ...
</ChartColorSchemes>
```

3 Add a new `<Scheme>` element.

The PCColors.xml file consists of one `<ChartColorSchemes>` element, which can contain any number of `<Scheme>` elements. Each `<Scheme>` element consists of a `Name` attribute, which defines the scheme name, and sixteen tags, `<Color1>` through `<Color16>`, each of which represent one color in the color scheme. These tags all have the attributes `Red`, `Green`, `Blue`, and `Alpha` which can be set to any number between *0* and *255*. The values you give the first three attributes will be used to compute the RGB value of the respective color. The value that you give the `Alpha` attribute, which is optional, defines the transparency of the color.

To add a new `<Scheme>` element in a text editor, simply scroll to the end of the file and, right before the `</ChartColorSchemes>` tag, insert the following tag:

```
<Scheme Name="MyColorTheme">
```

MyColorTheme should be the name of your color theme. Then, insert sixteen of the following tags:

```
<ColorXX Red="???" Green="???" Blue="???"">
```

XX should be the numbers between *1* and *16*, sequentially, and *???* should be a number between *0* and *255*, representing the intensity of the `Red`, `Green`, or `Blue` attribute of the color.

If you want to make a color translucent, you can also add an `Alpha` attribute. Setting the alpha value to 0 will make the color solid, while 255 will make the color invisible. Set the color somewhere in between for translucency. For example, the `Standard Translucent` theme sets the `Alpha` value of its first color to 80:

```
<Color1 Red="255" Green="0" Blue="0" Alpha="80"/>
```

Finally, insert the following tag:

```
</Scheme>
```

4 Save the file.

5 Restart PopChart Server and/or PopChart Builder.

You can also modify existing themes in the PCColors.xml file. Just change the `Red`, `Green`, and `Blue` attributes for the appropriate `<ColorXX>` tags in the desired `<Scheme>` element.

12

DATA ORGANIZATION

This chapter explains how to organize data that you send to PopChart Server. It provides examples of how to organize data in a standard spreadsheet format (used by data files and databases), as well in PopChart XML and PCScript.

Note: *For information on how to send this data to PopChart Server, you should refer to Chapter 6, "Sending Dynamic Data to PopChart Server," in the PopChart Server User Guide.*

The graphs that PopChart Server displays can be divided into four *data classes*, based on how they expect your data to be organized. The data classes, each of which will be discussed in this chapter, are as follows:

STANDARD DATA CLASS Area Graphs, Bar Graphs, Line Graphs, Line Bar Combo Graphs, Pareto Graphs, Pie Graphs, Radar Graphs, Stacked Bar Graphs.

PLOT DATA CLASS Bubble Graphs, Time Plot Graphs, X-Y Plot Graphs.

STOCK DATA CLASS Stock Graphs (High-Low/Open-Close, Candlestick).

GAUGE DATA CLASS Gauges.

This chapter just discusses the format of the data for these data classes. For a discussion of the graph types themselves, refer to Chapter 13, "Graph Types."

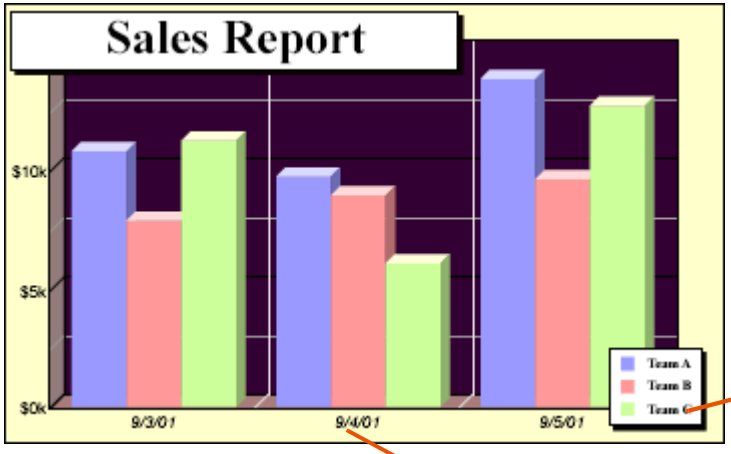
TERMINOLOGY

The building block of every graph in PopChart Server is a *data item*. Exactly what this data item is will depend on the graph. For example, a data item in a bar graph is simply a single data value represented as a bar. A data item in an X-Y Bubble graph, however, consists of three data values—an X-value, a Y-value, and Bubble value. Grouped together, all of the data items in a graph can be referred to as a *data set*.

PopChart Server groups data items into *series* and *categories*. It is easiest to think of series and categories as corresponding to rows and columns on a spreadsheet, respectively. Be aware, though, that this does not necessarily mean that your data will be organized in this fashion when you send it to PopChart Server.

All data items in a specific data *series* are displayed in the same color. In line graphs, they are also connected by a line. Each item in a series of data belongs to a different *category*. All items in the same category will be grouped together in the graph. Example 12.1 illustrates the roles of series and categories of data in creating a graph.

Example 12.1 **Series and Categories**



	9/3/01	9/4/01	9/5/01
Team A	10789	9745	13841
Team B	7861	8916	9612
Team C	11263	6047	12743

Each value on this row is a data item in the **Team C** *series* of data. The data items will appear as green bars.

Each value in this column is considered to be a data item in the **9/4/01** *category*. They are grouped together in the graph.

This example illustrates a graph that uses the Standard data class. The next section of this chapter discusses the Standard data class in detail. The rest of this chapter discusses other data classes. Though these classes organize their data differently, the concepts of data items, data values, series, and categories remain the same.

STANDARD DATA CLASS

This section discusses the standard data class. It first of all offers a Conceptual Overview of the data class. Then, it discusses how to organize your data in the following formats:

- Spreadsheet
- PopChart XML
- PCScript

CONCEPTUAL OVERVIEW

The *Standard* data class includes the following graph types:

- Bar Graphs
- Stacked Bar Graphs
- Pie Graphs
- Line Graphs
- Line Bar Combo Graphs
- Area Graphs
- Pie Graphs
- Radar Graphs
- Pareto Graphs

Because these graphs are all in the same data class, any data set that can be used for one of these graph types can be used for all of these graph types.

The Standard data class is the most intuitive data class. Graphs in the Standard data class organize data by *series* and *categories*, which correspond to rows and columns, respectively (although you can switch the series and categories by transposing the graph). Each data item consists of only one data value, which indicates the size or height of the data item. Each data item belongs to exactly one category and one series.

Data items are grouped together according to their category. Unless you override a data item's color, each data item in a series will be shown in the same color and symbol. In line graphs, data items in a data series will be connected by a line. There will be one legend item for each series.

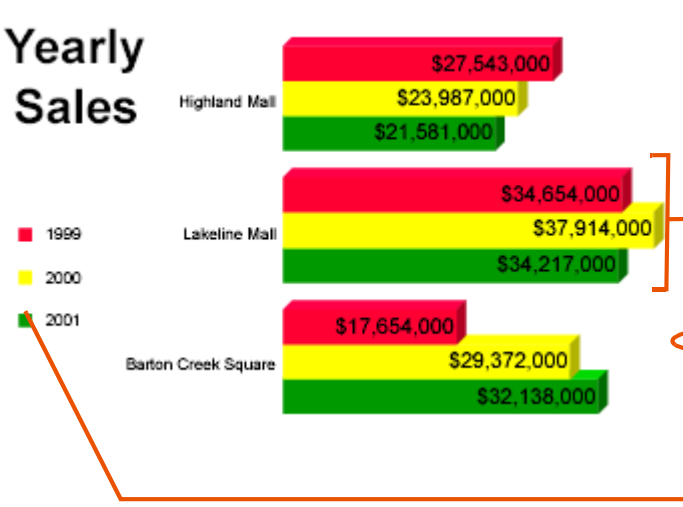
Pie graphs accept only one category of data, while Pareto graphs accept only one series of data.

SPREADSHEET

When organizing a Standard data set in a spreadsheet format, the first row will be interpreted as category names and the first column will be interpreted as series names. The top left cell of the spreadsheet should be left blank. All remaining data will be treated as individual data items, with each row corresponding to a single series and each column corresponding to a single category.

The spreadsheet Example 12.2 displays a typical data set for a graph that uses the Standard data class.

Example 12.2 **Standard Data Set**



Each value in this column is considered to be a data item in the **Lakeline Mall** *category*. They are grouped together in the graph.

	Highland Mall	Lakeline Mall	Barton Creek Square
1999	27,543,000	34,654,000	17,654,000
2000	23,987,000	37,914,000	29,372,000
2001	21,581,000	34,217,000	32,138,000

Each value on this row is a data item in the **2001** *series* of data. The data items will appear as green bars.

POPCHART XML

For the Standard data class, PopChart XML `Data` elements require the `Value` attribute, which specifies the value of the data item.

Example 12.3 illustrates how a data set for a graph in the Standard data class would look in PopChart XML. this example assumes that the graph is named *graph*. Note that the data set is the same one used in Example 12.2.

Example 12.3 **Standard Data Set in PopChart XML**

```
<GraphData Name='graph'>
    <Categories>
        <Category Name='Highland Mall'/>
        <Category Name='Lakeline Mall'/>
        <Category Name='Barton Creek Square'/>
    </Categories>
    <Series Name='1999'>
        <Data Value='27,543,000'/>
        <Data Value='34,654,000'/>
        <Data Value='17,654,000'/>
    </Series>
    <Series Name='2000'>
        <Data Value='23,987,000'/>
        <Data Value='37,914,000'/>
        <Data Value='29,372,000'/>
    </Series>
    <Series Name='2001'>
        <Data Value='21,581,000'/>
        <Data Value='34,217,000'/>
        <Data Value='32,138,000'/>
    </Series>
</GraphData>
```

For an explanation of the PopChart XML data format, refer to *"Sending Data with PopChart XML"* on page 6-5 of the *PopChart Server User Guide*.

PCSCRIPT

To send a standard data set via PCScript, first specify your category names by calling the `Graph.SetCategories()` method. Each category name should be separated by a semi-colon. Then, send each series to the graph via separate `Graph.SetSeries()` commands. The first parameter in this command should be the series name. Then you should list each data item in the series, separating them with a semi-colon.

Assuming that your graph object is named *graph*, you could send the standard data set of Example 12.2 to a graph with the PCScript in Example 12.4:

Example 12.4 **Standard Data Set in PCScript**

```
graph.SetCategories(Highland Mall; Lakeline Mall; Barton Creek
                    Square)
graph.SetSeries(1999; 27543000; 34654000; 17654000)
graph.SetSeries(2000; 23987000; 37914000; 29372000)
graph.SetSeries(2001; 21581000; 34217000; 32138000)
```

Warning: Do not use commas to separate the thousand and million positions of numbers in PCScript. PopChart Server will think you are specifying multiple data values.

To learn more about sending data via PCScript, refer to *"Sending Data with PopChart Script"* on page 6-10 of the *PopChart Server User Guide*.

PLOT DATA CLASS

This section discusses the Plot data class. It first of all offers a Conceptual Overview of the data class. Then, it discusses how to organize your data in the following formats:

- Spreadsheet
- PopChart XML
- PCScript

CONCEPTUAL OVERVIEW

The *Plot* data class includes the following graph types:

- Bubble Graphs
- Time Plot Graphs
- X-Y Plot Graphs

Note: *Unlike graphs in other data sets, graphs in the Plot data set are not interchangeable. In other words, a Time Plot data set cannot be used as an X-Y Plot data set.*

The Plot data class is used by graphs whose data items are plotted along two axes, like the X-Y graph. Each data item will have at least two data values—an `x`-value and a `y`-value (for time graphs, these are called the `data` and `value` attributes, respectively, but the concept remains the same). These values indicate the position of the data item along the x-axis and y-axis, respectively. Additionally, Plot data items can have a `bubble` value, which is used to determine the radius of bubbles in bubble graphs.

Like graphs in the Standard data class, Plot graphs organize data by *series*. However, they do not organize data by categories. Unless you override a data item's color or symbol, each data item in a series will be shown in the same color and with the same symbol. If the series is being displayed as a line, data items in that series will be connected by a line. There will be one legend item for each series.

SPREADSHEET

When organizing a Plot data set in a spreadsheet format, each series of data will consist of three columns. For example, the first series consists of the data values in the first through third columns, the second series consists of the data in the fourth through sixth columns, and so on.

The first row of the spreadsheet is reserved for series names. The name for each series will appear in the first column for that series, while any remaining cells in the first row should be left blank. All other rows represent individual data items. In the first column for the series,

you should place the x or date value for the data item. In the second column for the series, you should place the y value for the data item. You should leave the third column empty unless you are plotting bubble values.

Example 12.5 illustrates a typical Plot data set in a spreadsheet.

Example 12.5

Plot Data Set (Time)

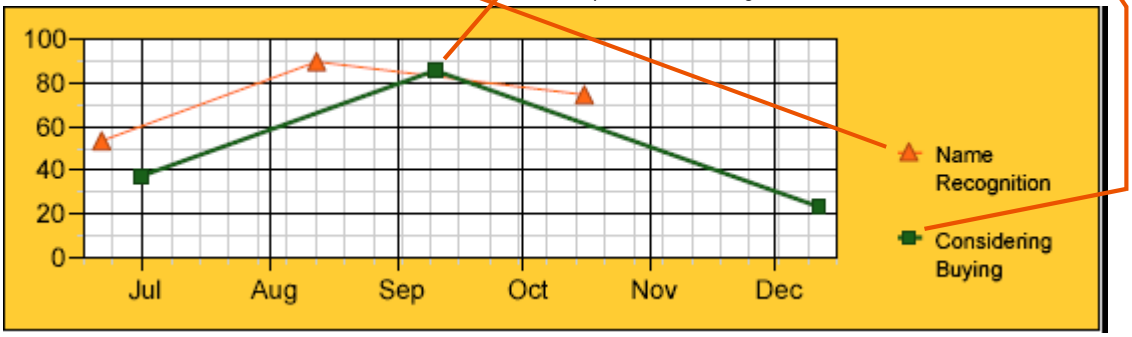
If you are using Plot data without a bubble value, you must still have a bubble column, but leave that column empty.

The data in the second set of columns is used for the Considering Buying series.

Name Recognition			Considering Buying		
6/21/2001	54		7/1/2001	37	
8/12/2001	90		12/12/2001	23	
10/16/2001	75		9/10/2001	86	

The data in the first two columns is used to form the data points in the Name Recognition series.

The data values in these two cells form a single data item, which is plotted in a green square at 86 on the y-axis, and September 10th along the time-axis.



Note: *For information about how to format dates in Time plot graphs, you should refer to "Date Input Format (Time Plot)" on page A-26 of the PopChart Builder User Guide.*

Any graph that uses a Plot data set can become a bubble graph by inserting values into the third column of for the series (the first series consists of columns one to three, the second series consists of columns four through six, etc.). This third column will contain bubble values for each data item.

Example 12.6 illustrates a Plot data set with bubble values.

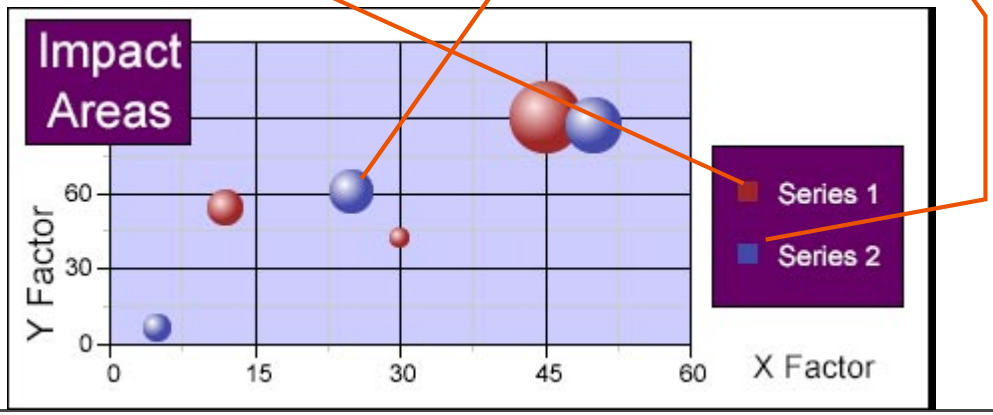
Example 12.6 **Plot Data Set (X-Y Bubble)**

Series 1			Series 2		
12	54	5	5	6	4
45	90	10	25	60	6
30	42	3	50	87	8

The data in the second set of columns is used for the Series 2 series.

The data in the first three columns is used to form the data points in the Series 1 series.

The data values in these three cells form a single data item (bubble), which is plotted at 25 on the x-axis, 60 along the y-axis, and has a 6 pixel diameter.



POPCHART XML

For the Plot data class, PopChart XML `Data` elements require either the `X` and `Y` attributes (for X-Y graphs) or the `Date` and `Value` attributes (for Time graphs).

Example 12.7 illustrates how a data set for a Time graph would look in PopChart XML. Note that the data set is the same one used in Example 12.5.

Example 12.7 **Time Plot Data Set in PopChart XML**

```
<GraphData Name='graph2' SortData='Yes'>
    <Series Name='Name Recognition'>
        <Data Date='6/21/2001' Value='54'/>
        <Data Date='8/12/2001' Value='90'/>
        <Data Date='10/16/2001' Value='75'/>
    </Series>
```

```
    <Series Name='Considering Buying'>
        <Data Date='7/1/2001' Value='37'/>
        <Data Date='12/12/2001' Value='23'/>
        <Data Date='9/10/2001' Value='86'/>
    </Series>
</GraphData>
```

Note: *The `SortData` attribute in a graph makes it so the data points are connected in order of their date (or x value) instead of the order in which they are listed.*

Example 12.8 illustrates how a data set for a X-Y graph would look in PopChart XML. Note that the data set is the same one used in Example 12.6, minus the bubble values.

Example 12.8 X-Y Plot Data Set in PopChart XML

```
<GraphData Name='graph'>
    <Series Name='Series 1'>
        <Data X='12' Y='54'/>
        <Data X='45' Y='90'/>
        <Data X='30' Y='42'/>
    </Series>
    <Series Name='Series 2'>
        <Data X='5' Y='6'/>
        <Data X='25' Y='60'/>
        <Data X='50' Y='87'/>
    </Series>
</GraphData>
```

Any graph using a Plot data set can become a bubble graph by adding the `Bubble` attribute. Example 12.9 illustrates how a data set for a Bubble graph would look in PopChart XML. Note that the data set is the same one used in Example 12.6.

Example 12.9 Bubble Plot Data Set in PopChart XML

```
<GraphData Name='graph'>
    <Series Name='Series 1'>
        <Data X='12' Y='54' Bubble='5'/>
        <Data X='45' Y='90' Bubble='10'/>
        <Data X='30' Y='42' Bubble='3'/>
    </Series>
    <Series Name='Series 2'>
        <Data X='5' Y='6' Bubble='4'/>
        <Data X='25' Y='60' Bubble='6'/>
```

```
        <Data X='50' Y='87' Bubble='8'/>
    </Series>
</GraphData>
```

For an explanation of the PopChart XML data format, refer to *"Sending Data with PopChart XML"* on page 6-5 of the *PopChart Server User Guide*.

PCSCRIPT

To send a Plot data set via PCScript, simply send each series to the graph via separate `Graph.SetSeries()` commands. The first parameter in this command should be the series name. Then you should list each data item in the series, separating data values with a comma and data items with a semi-colon.

Note: *The Plot data class does not use the `Graph.SetCategories()` command.*

A Plot data item will consist of two or three data values, each separated by a comma. For X-Y Plot data, these values should be listed in the following order: `x`-value, `y`-value, `bubble` value. For Time Plot data, the order will be: `date`, `value`, `bubble` value. In both cases, the `bubble` value is optional.

Assuming that your graph object is named *timegraph*, you could send the Time Plot data set of Example 12.5 to a graph with the PCScript Example 12.10:

Example 12.10 **Time Plot Data Set in PCScript**

```
timegraph.SetSeries(Name Recognition; 6/21/2001,54; 8/12/2001,90;
                    10/16/2001,75)
timegraph.SetSeries(Considering Buying; 7/1/2001,37;
                12/12/2001,23; 9/10/2001,86)
```

Note: *You cannot have enable data sorting in PCScript. You will need to either pre-sort your values (which is recommended anyway), or enable data sorting in the appearance file by opening it in PopChart Builder and checking the Sort Data box in the Graph Properties > General dialog pane.*

Assuming that your graph object is named *xygraph*, you could send the X-Y Bubble data set of Example 12.6 to a graph with the PCScript in Example 12.11:

Example 12.11 **X-Y Bubble Data Set in PCScript**

```
xygraph.SetSeries(Series 1; 12,54,5; 45,90,10; 30,42,3)
xygraph.SetSeries(Series 2; 5,6,4; 25,60,6; 50,87,8)
```

To learn more about sending data via PCScript, refer to *"Sending Data with PopChart Script"* on page 6-10 of the *PopChart Server User Guide*.

STOCK DATA CLASS

This section discusses the Stock data class. It first of all offers a Conceptual Overview of the data class. Then, it discusses how to organize your data in the following formats:

- Spreadsheet
- PopChart XML
- PCScript

CONCEPTUAL OVERVIEW

The *Stock* data class includes all Stock Graphs, including:

- Candlestick
- High-Low
- High-Low / Open-Close

Because these graphs are all in the same data class, any data set that can be used for one of these graph types can be used for all of these graph types.

The Stock data class is used by graphs that analyze a stocks performance. Stock data items consist of up to four data values. Two are required—`High` and `Low`, while `Open` and `Close` are optional. For more information about what these data values stand for, see "Stock Graphs" on page 13-17.

Graphs in the Stock data class organize data items by *category.* There is only one series of data, and all data items are the same color.

SPREADSHEET

When organizing a Stock data set in a spreadsheet format, the cell in second column of the first row of the spreadsheet will be used for the series name. Any other cells in the first row will be ignored. All remaining rows will be interpreted as categories (not series, as you would expect in other data classes). Since there is only one series of data, you could also say that each row represents an individual data item.

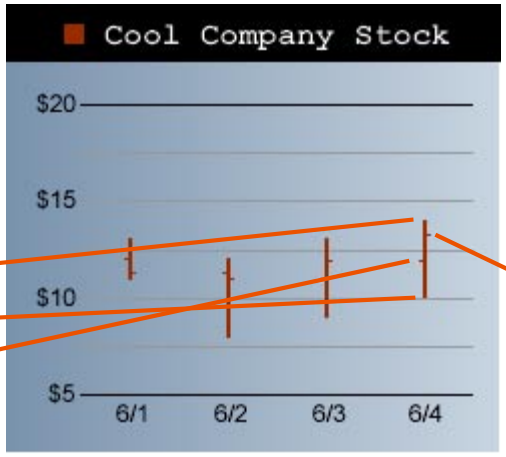
The first column of the spreadsheet contains category names. The second column represents the high value for each data item. The third column represents the low value, the fourth column represents the open value, and the fifth column represents the close value. These last two columns are optional, and will be invisible if you are just displaying a High-Low graph.

The spreadsheet in Example 12.12 displays a typical data set for a graph that uses the Stock data class.

Example 12.12 **Stock Data Set**

	Cool Company Stock			
6/1	13	11	12	11.3
6/2	12	8	11.3	11
6/3	13	9	11	11.9
6/4	14	10	11.9	13.2

The name of the stock appears in legend.



Stock's High Price

Stock's Low Price

The values in this row are used to create the data item in the **6/1** *category*. The data item translates into the stock bar for 6/1.

The opening price of the stock for the day, indicated by a left tick on the stock

The closing price of the stock for the day, indicated by a right tick on the stock bar.

POPCHART XML

For the Stock data class, PopChart XML `Data` elements require either the `High` and `Low` attributes. They can also include `Open` and `Close` attributes. These latter values will only be used for Candlestick and High-Low/Open-Close graphs.

Example 12.13 illustrates how a data set for a graph in the Stock data class would look in PopChart XML. Note that the data set is the same one used in Example 12.12.

Example 12.13 **Stock Data Set in PopChart XML**

```
<GraphData Name='graph'>
    <Categories>
        <Category Name='6/1'/>
        <Category Name='6/2'/>
        <Category Name='6/3'/>
        <Category Name='6/4'/>
    </Categories>
    <Series Name='Cool Company Stock'>
        <Data High='13' Low='11' Open='12' Close='11.3'/>
        <Data High='12' Low='8' Open='11.3' Close='11'/>
```

```
        <Data High='13' Low='9' Open='11' Close='11.9'/>
        <Data High='14' Low='10' Open='11.9' Close='13.2'/>
    </Series>
</GraphData>
```

Note: *With stock graphs, you can also set your categories directly in the* `<Data>` *tag. To do this, simply use the* `Name` *attribute and set it to your category name. This allows you to remove everything in the* `Categories` *element.*

For an explanation of the PopChart XML data format, refer to *"Sending Data with PopChart XML"* on page 6-5 of the *PopChart Server User Guide*.

PCSCRIPT

To send a Stock data set via PCScript, first specify your category names by calling the `Graph.SetCategories()` method. Each category name should be separated by a semi-colon. Then, send the data series to the graph with the `Graph.SetSeries()` commands. The first parameter in this command should be the series name. Then you should list each data item in the series, separating data values with a comma and data items with a semi-colon.

A Stock data item will consist of two to four data values, each separated by a comma. These values should be listed in the following order: `high` value, `low` value, `open` value, `close` value. The last two values are optional.

Assuming that your graph object is named *graph*, you could send the Stock data set of Example 12.12 to a graph with the PCScript in Example 12.14:

Example 12.14 **Stock Data Set in PCScript**

```
graph.SetCategories(6/1; 6/2; 6/3; 6/4)
graph.SetSeries(Cool Company Stock; 13,11,12,11.3; 12,8,11.3,11;
             13,9,11,11.9; 14,10,11.9,13.2)
```

Note: *With stock graphs, you can also specify your category names directly in the* `Graph.SetSeries()` *method. To do this, simply list your category name as the first value in the data item (e.g. the first data item in Example 12.14 would be* `6/1,11,12,11.3`*).This allows you to remove the* `Graph.SetCategories()` *statement.*

If you need to change the order of the data values (e.g. `open`, `high`, `low`, `close` instead of `high`, `low`, `open`, `close`) using the `Graph.SetStockHLOCOrder()` Command. For instance, Example 12.15 shows how you would send the data from Example 12.12 if you changed the order of your data values to `open`, `high`, `close`, `low`.

Example 12.15 **Changing Stock Data Value Order in PCScript**

```
graph.SetStockHLOCOrder( 4, 2, 5, 3)
graph.SetCategories(6/1; 6/2; 6/3; 6/4)
graph.SetSeries(Cool Company Stock; 12,13,11,11.3; 11.3,12,8,11;
              11,13,9,11.9; 11.9,14,10,13.2)
```

Note: *When importing data from a table, column 1 is assumed to be the category name.*

To learn more about sending data via PCScript, refer to *"Sending Data with PopChart Script"* on page 6-10 of the *PopChart Server User Guide*.

GAUGE DATA CLASS

This section discusses the Stock data class. It first of all offers a Conceptual Overview of the data class. Then, it discusses how to organize your data in the following formats:

- Spreadsheet
- PopChart XML
- PCScript

CONCEPTUAL OVERVIEW

The *Gauge* data class includes the following graph types:

- Gauges

Gauges bear little resemblance to other graph types. They "graph" only one data value, which determines the color of the gauge.

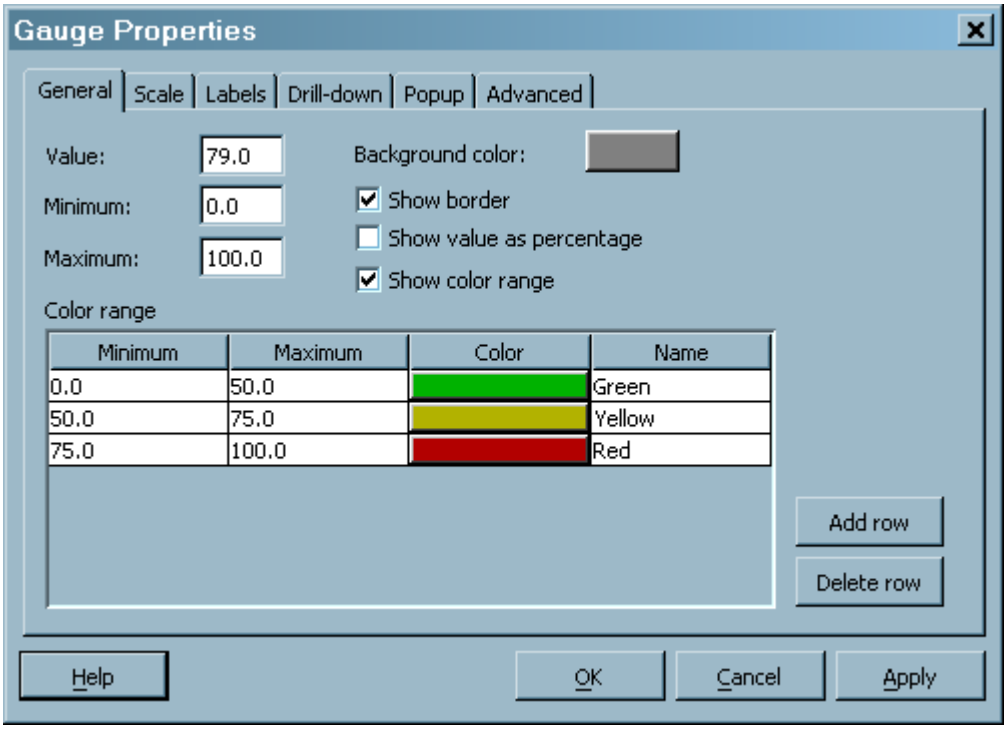
Besides the value of a gauge, you can also specify a label for the gauge, and color ranges. Color ranges, which consists of a range name (e.g. *Critical*), a color value (e.g. *#FF0000*), and a maximum and minimum value, are used to determine the color of the gauge. If the data value falls between the maximum and minimum value of a range, the gauge will appear in that range's color. On some gauges you can also set a minimum and maximum value for the gauge itself.

Note: *When you have overlapping ranges, the data value will be in the most recently created range. For example, if you have a green range from 80 to 130, and you have a red range of 93 to 110, a data value of 94 would be considered to be in the red range.*

Gauges do not have legends.

SPREADSHEET

You cannot import data to a gauge in spreadsheet format. In PopChart Builder you will be given the following interface for manipulating gauges.



Because you cannot import data to a gauge in spreadsheet form, do not use the `loadData()` or `setData()` PopChart Embedder methods or the `LoadFile()` PCScript method with gauges. Instead you should use PopChart XML or the `SetGaugeRange()` and `SetGaugeValue()` PCScript methods.

POPCHART XML

Gauges do not use PopChart XML's `GraphData` element. Instead, you set the value and ranges of the gauge in its `Graph` element.

To set a gauge's data value, label, and minimum/maximum values, you should use the `Values` subelement in `Graph`. The `Values` subelement contains these attributes: `Value`, `Text`, `Minimum`, and `Maximum`. For each range in the graph, your `Graph` element should have a separate `ColorRange` subelement. The ColorRange subelement has the following attributes: `Name`, `Color`, `Minimum`, and `Maximum`.

Example 12.16 illustrates how a `Graph` element for a gauge would look in PopChart XML.

Example 12.16 Setting Gauge Values and Ranges in PopChart XML

This line resets the color ranges.

The green color range is between 0 and 50.

The yellow color range is between 50 and 75.

```
<Graph Name='graph' Type='Gauge' SubType='Filled Bar'>
    <Values Text='Server Load' Value='79.0'/>
    <ColorRange NoColorRanged='True' />
    <ColorRange Name='Normal' Color='#00b200'
        Minimum='0.0' Maximum='50.0' />
    <ColorRange Name='Warning' Color='#b2b200'
        Minimum='50.0' Maximum='75.0' />
    <ColorRange Name='Critical' Color='#b20000'
        Minimum='75.0' Maximum='100.0' />
    <Scale Minimum='0' Maximum='100' />
</Graph>
```

The red color range is between 75 and 100. The data value, 79, is in this range, so the range is "filled" until 79, and empty afterwords.

Sets the maximum and minimum of the scale.

PCSCRIPT

You can set a gauge's value and ranges via the `Graph.SetGaugeValue()` and `Graph.SetGaugeRange()` PCScript methods.

The `Graph.SetGaugeValue()` method accepts multiple sets of one to four parameters in the following order: the gauge value, the gauge label, the gauge minimum value, and the gauge maximum value.

To set the color ranges in a gauge, you should use the `Graph.SetGaugeRange()` command. This method accepts multiple sets of four parameters: the name of the range, the six-digit hexadecimal color code, the minimum value in the range, and the maximum value in the range. Note that the color code should *not* be preceded by a pound # sign. Each parameter set should be separated by a semi-colon.

Assuming that your graph object is named *gauge*, you could set the values for the gauge from Example 12.16 with the PCScript in Example 12.17:

Example 12.17 **Setting Gauge Values and Ranges in PCScript**

```
Gauge.SetGaugeValue(79,Server Load,0,100)
Gauge.SetGaugeRange(Normal,00b200,0,50; Warning,b2b200,50,75;
                Critical,b20000,75,100)
```

13

Graph Types

This chapter describes all of the available graph types in PopChart Server. These descriptions will help you decide which graph types you should use in your PopChart images. It will also explain how you can create and manipulate the graph in PopChart XML.

The following graph types are discussed in this chapter. You can jump to the description of the graph type by clicking on a topic.

Note: *Most graph types have several subtypes. For example, the Bar graph has Vertical and Horizontal subtypes. These subtypes will be discussed in the same section as the main graph type. 3D graphs are considered variations, not subtypes.*

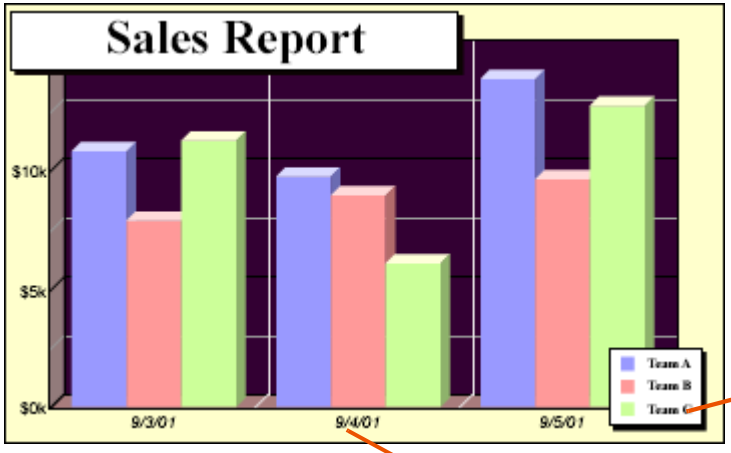
BAR GRAPHS

Bar graphs can be used for:

- Comparing independent data sets
- Money distribution by time
- Production against time

Bar graphs are the most common graph type, and are useful for a wide variety of data. Bar graphs group data together according to categories, displaying one bar for each data item in that category. Each bar is color-coded according to the data series that it represents. Example 13.1 shows how a Bar graph works.

Example 13.1 **Vertical Bar Graph**



	9/3/01	9/4/01	9/5/01
Team A	10789	9745	13841
Team B	7861	8916	9612
Team C	11263	6047	12743

Each value on this row is a data item in the **Team C** *series* of data. The data items will appear as green bars.

Each value in this column is considered to be a data item in the **9/4/01** *category*. They are grouped together in the graph.

Bar graphs are in the Standard Data Class (see Chapter 12 for details). To create a Bar graph with PopChart XML, create a `Graph` element and set its `Type` attribute to *Bar*, as shown in the example below.

```
<Graph Name='graph' Type='Bar'>
```

SUBTYPES

Listed below are the Bar graph subtypes.

VERTICAL BAR

The graph depicted in Example 13.1 is a Vertical bar graph (i.e. the bars are vertical).

If you don't specify a subtype for a Bar graph in PopChart XML, PopChart Server will assume that it is of the Vertical subtype. Or, to specify the Vertical subtype explicitly, set the `SubType` attribute of your `Graph` element to *Vertical*, as in the example below.

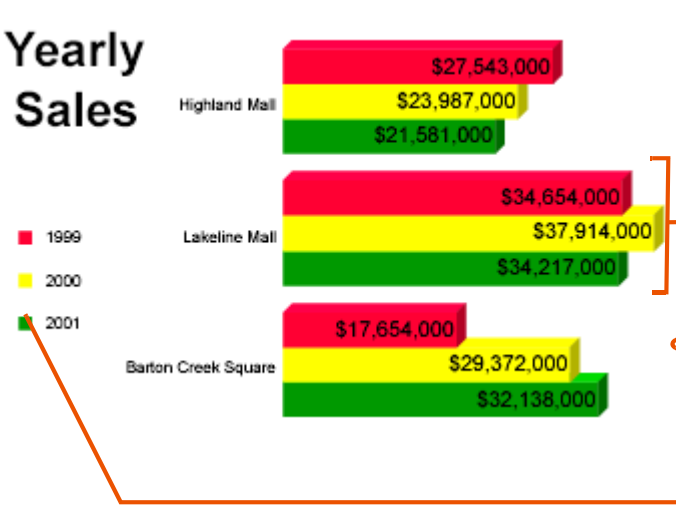
```
<Graph Name='graph' Type='Bar' SubType='Vertical'>
```

HORIZONTAL BAR

Horizontal Bar graphs are similar to Vertical Bar graphs, the only difference being that they are rotated 90 degrees. If you have long category or data labels and you don't want to rotate the labels, a Horizontal Bar graph may be more convenient than a Vertical Bar graph.

Example 13.2 gives an example of a Horizontal Bar graph.

Example 13.2 **Horizontal Bar Graph**



Each value in this column is considered to be a data item in the **Lakeline Mall** *category*. They are grouped together in the graph.

	Highland Mall	Lakeline Mall	Barton Creek Square
1999	27,543,000	34,654,000	17,654,000
2000	23,987,000	37,914,000	29,372,000
2001	21,581,000	34,217,000	32,138,000

Each value on this row is a data item in the **2001** *series* of data. The data items will appear as green bars.

To create the Horizontal Bar graph in PopChart XML, set the `SubType` attribute of your `Graph` element to *Horizontal*, as in the example below.

```
<Graph Name='graph' Type='Bar' SubType='Horizontal'>
```

VERTICAL STACKED BAR

The Vertical Stacked bar subtype is discussed in the "Stacked Bar Graphs" section.

HORIZONTAL STACKED BAR

The Vertical Stacked bar subtype is discussed in the "Stacked Bar Graphs" section.

FORMATTING OPTIONS

Listed below are several ways you can customize your Bar graphs

SINGLE SERIES MULTICOLOR MODE

When you only have one series of data, you may want to represent each category in that series with a different color. To do this you can enable Single Series Multicolor Mode by right-clicking on the graph, choosing Graph Properties, and, in the dialog box that will appear, clicking the check box marked Single Series Multicolor Mode.

STACKED BAR GRAPHS

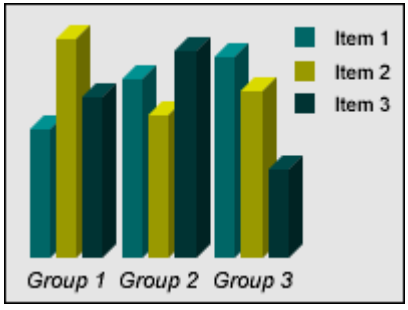
Stacked Bar graphs can be used for:

- Comparing independent data sets
- Money distribution by time
- Production against time

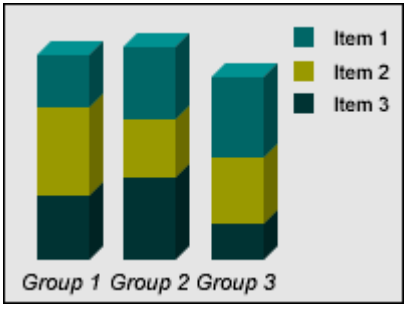
Technically, the Stacked Bar graph is a graph subtype, not a separate type of graph. But since it is a subtype for several types of graphs, it is more convenient to discuss the concept of a Stacked Bar graph separately.

Stacked bar graphs work well when you need to show the cumulative effect of a category of data. In a Stacked Bar graph, all of the bars in a category are stacked on top of each other. The resulting stacked bar represents the sum of all of the data items in the respective category. Each data item is represented as a segment in the stacked bar.

The graph also gives you an idea of how data from a particular series contributes to the category sum. Each segment of the bar is color-coded according to the series of data that the segment represents. The difference between a Stacked Bar graph and a typical Bar graph is illustrated below.



Vertical Bar Graph



Stacked Vertical Bar Graph

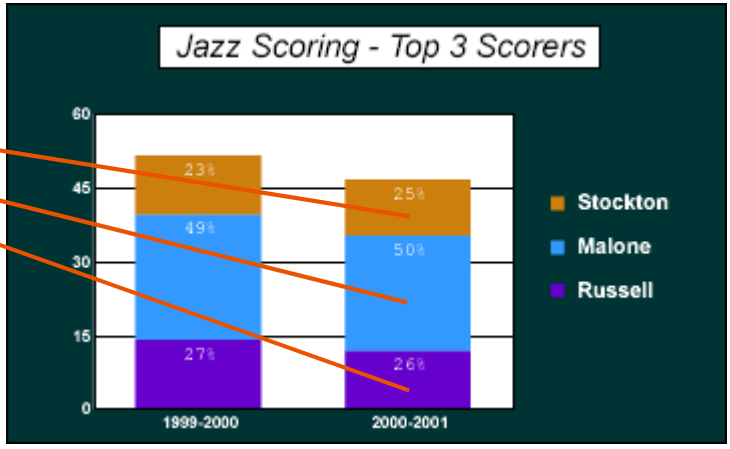
Stacked Bar graphs are in the Standard Data Class (see Chapter 12 for details). Example 13.3 shows exactly how your data is used to form a Stacked Bar graph.

Example 13.3 **Stacked Bar Graph**

	1999-2000	2000-2001
Stockton	12.1	11.5
Malone	25.5	23.2
Russell	14.1	12.0

Each value in this column is considered to be a data item in the **2000-2001** *category*. They are summed together to create a stacked bar for that category.

Each value on this row is a data item in the **Russell** *series* of data. The data items will appear as purple segments.



SUBTYPES

Listed below are the different types of graphs that use Stacked Bars.

VERTICAL STACKED BAR

The graph depicted in Example 13.1 is a Vertical bar graph (i.e. the bars are vertical).

To create a Vertical Stacked bar graph in PopChart XML, you should set the `Type` attribute of `Graph` to *Bar*, and the `SubType` attribute to *Vertical Stacked*. The tag below shows how this is done.

```
<Graph Name='graph' Type='Bar' SubType='Vertical
  Stacked'>
```

HORIZONTAL STACKED BAR

Horizontal Stacked Bar graphs are similar to Vertical Stacked Bar graphs, the only difference being that they are rotated 90 degrees. If you have long category or data labels and you don't want to rotate the labels, a Horizontal Stacked Bar graph may be more convenient than a Vertical Stacked Bar graph.

To create a Vertical Stacked bar graph in PopChart XML, you should set the `Type` attribute of `Graph` to *Bar*, and the `SubType` attribute to *Vertical Stacked*. The tag below shows how this is done.

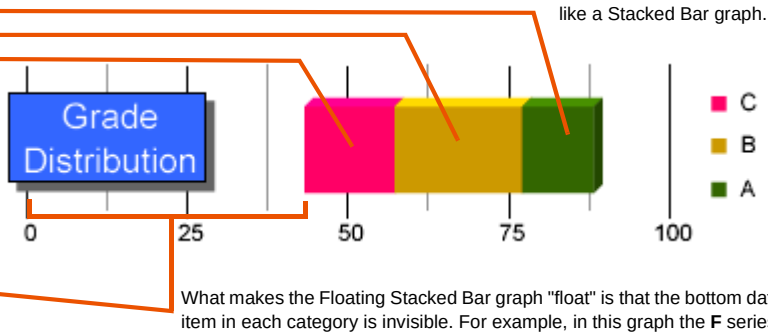
```
<Graph Name='graph' Type='Bar' SubType='Horizontal
  Stacked'>
```

FLOATING STACKED BAR

Both Vertical and Horizontal Stacked Bar graphs can be turned into Floating Stacked Bar graphs. In a Floating Stacked Bar graph, the bottom segment (last data item) in each stacked bar will be invisible. This has the effect of making the bar seem to float. Example 13.4 shows how to use a Floating Stacked Bar graph.

Example 13.4 **Floating Stacked Bar Graph**

	Grade
A	11
B	20
C	14
F	45



For the most part, the Floating Stacked Bar graph works just like a Stacked Bar graph.

What makes the Floating Stacked Bar graph "float" is that the bottom data item in each category is invisible. For example, in this graph the **F** series does not appear as a segment in the bar, nor does it appear as a legend entry.

To create a Floating Stacked Bar graph in PopChart XML, simply add a `Properties` subelement to your `Graph` element (if it does not already exist). In the `Properties` subelement, add a `FloatingBars` attribute and set its value to *Yes*. The PopChart XML in Example 13.5 below shows how you would turn a Horizontal Stacked Bar graph into a Floating Horizontal Stacked Bar graph.

Example 13.5 **Floating Stacked Bar in PopChart XML**

```
<Graph Name='graph' Type='Bar' SubType='Horizontal Stacked'>
    <Properties FloatingBars='Yes'/>
</Graph>
```

PIE GRAPHS

Pie graphs can be used for:

- Comparing data elements against the sum of the elements
- Budgets
- Money distribution

The Pie graph is another commonly used graph type. Pie graphs are most often used to show data as a percentage of the whole. Pie graphs represent an entire data category as a pie (all other data is thrown out). Each data item in that category is shown as a pie wedge. The wedge size will be proportional to that data item's percentage of the sum of all data items in the category.

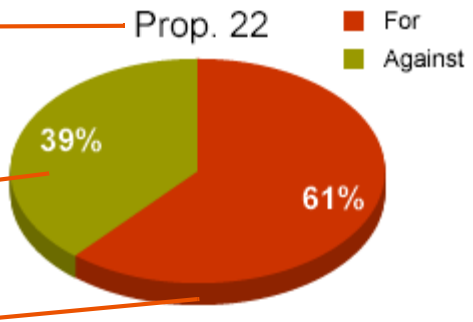
Though this may sound complex, Pie graphs are actually quite intuitive. Example 13.6 shows exactly how they work.

Example 13.6 **Pie Graph**

Each pie graph has exactly one category.

	Prop. 22
For	4,158,823
Against	2,617,004

The category label becomes the graph label.



The most obvious way of looking at this graph is that the number of ***For*** votes was 61% of the total votes, so its wedge is 61% of the pie.

In technical terms, the **Against** data item was 39% of the sum of all data items in the category, so its wedge is 39% of the graph.

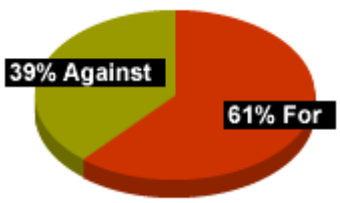
Pie graphs are in the Standard Data Class (see Chapter 12 for details). To create a Pie graph with PopChart XML, create a `Graph` element and set its `Type` attribute to *Pie*, as shown in the example below.

```
<Graph Name='graph' Type='Pie'>
```

FORMATTING OPTIONS

DATA LABELS

Pie graphs do not have scales, ticks, or gridlines. They do have data labels, but you will notice that there are many options for displaying them that are not available for other graph types. For example, in the Graph Properties > Data Labels pane, in addition to the Show Value As option (which lets you display the data label as either the actual value or as a percentage of the pie), you will see the Show Series Text option. This lets you display a data label that consists of both the series label (*For* or *Against* in the previous example) and the regular data label. The Position option also allows you to make the data labels callouts outside of the pie using a Leader.



Show Series Text as "Name and Value"

Position as "Out W/ Leader"

GAPS

Also, this subtype gives you the option of putting a gap between each wedge. To do this, go to the Graph Properties dialog and in the General pane, select the option 2D With Gaps.

Regular Pie

Pie With Gaps

LINE GRAPHS

Line graphs can be used for:

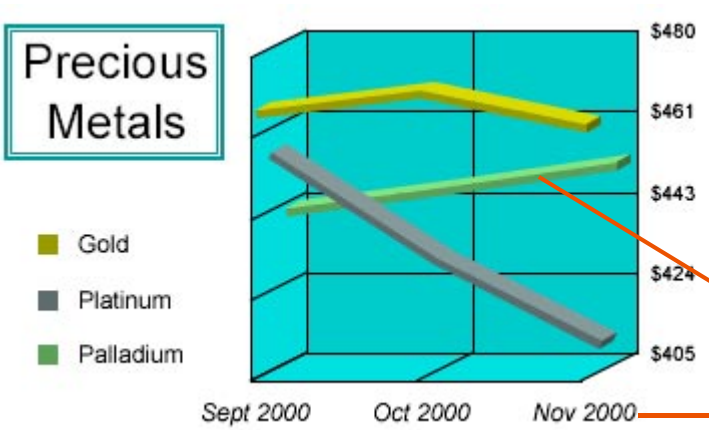
- Determining trends or cyclical variations
- Money distribution over time
- Production over time
- Price variation over time
- Environmental changes over time
- Material characteristics by temperature
- Clinical response by concentration

The Line graphs is another commonly used graph type. Line graphs are useful when you want to show data trends. In a Line graph, data items are represented by points or symbols. Data items in the same series are connected to each other by a line so that you can see a trend in your data over a number of data categories. The categories of data become the scale indexes along the x-axis, while each series of data is represented by an individual line.

Note: *The X-Y Line subtype of the X-Y Plot graph type looks very similar to a Line graph. However, it uses data differently. If your data is in X-Y pair data points you will want to use X-Y Line subtype of the X-Y Plot graph. Otherwise, you will probably want to use the standard Line graph described here.*

Example 13.7 illustrates the use a Line graph.

Example 13.7 **Line Graph**



	Sept 2000	Oct 2000	Nov 2000
Gold	466.47	470.92	463.12
Platinum	455.00	430.23	411.12
Palladium	440.19	445.00	450.68

The values on this row are data items in the **Palladium** *series* of data. Each value represents the value of the **Palladium** (green) line for the *category* named at the bottom of its column.

The values in this column are data items in the Nov 2000 *category* of data. They will be plotted along the same vertical gridline in the graph.

Line graphs are in the Standard Data Class (see Chapter 12 for details). To create a Line graph with PopChart XML, create a `Graph` element and set its `Type` attribute to *Line*, as shown in the example below.

```
<Graph Name='graph' Type='Line'>
```

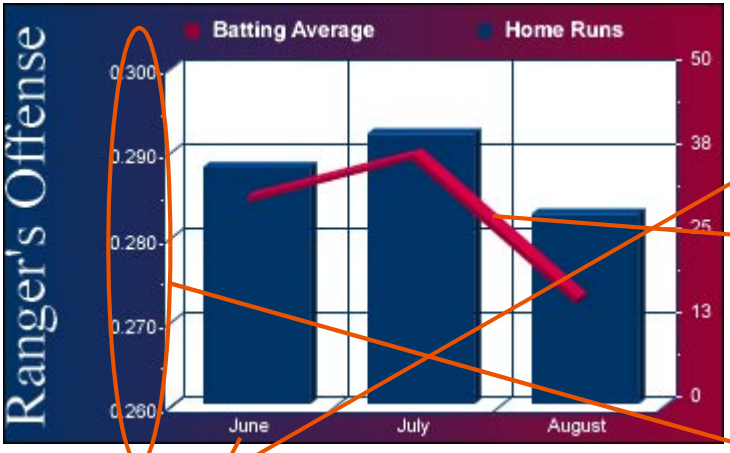
LINE BAR COMBO GRAPHS

Line Bar Combo graphs allow you to have bars and lines in the same graph. They are very similar to Line Graphs and Most graph types have several subtypes. For example, the Bar graph has Vertical and Horizontal subtypes. These subtypes will be discussed in the same section as the main graph type. 3D graphs are considered variations, not subtypes.

A Line Bar Combo graph represents its data according to the number of data series in it. By default, those data series that constitute the first half of the total number of series are represented as lines. The remaining data series are represented as bars. If there is an odd number of data series there will be one more bar data series than line data series.

Example 13.8 shows how Line Bar Combo graphs work.

Example 13.8 **Line Bar Combo Graph**



	June	July	August
Batting Average	.285	.290	.273
Home Runs	35	40	28

Because the **Batting Average** *series* is in the first half of all the series (rows) of data in the graph, it is represented as a line.

Because the **Home Run** *series* is in the last half of all the series (rows) of data in the graph, it is represented with bars.

All data values in this column are placed/plotted in the **June** *category*.

The value scale on the left of the graph is for all the data series in the graph that are displayed as line. In this case, that means just the **Batting Average series**.

Line Bar Combo graphs are in the Standard Data Class (see Chapter 12 for details). To create a Line Bar Combo graph with PopChart XML, create a `Graph` element and set its `Type` attribute to *Line Bar*, as shown in the example below.

```
<Graph Name='graph' Type='Line Bar'>
```

SUBTYPES

Listed below are the Line Bar Combo graph subtypes.

Note: *Line Bar Combo graphs only use Vertical Bars.*

BASIC

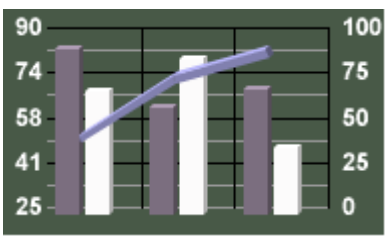
The graph depicted in Example 13.8 is a basic Line Bar Combo graph.

If you don't specify a subtype for a Line Bar Combo graph in PopChart XML, PopChart Server will assume that it is of the Basic subtype. Or, to specify the Basic subtype explicitly, set the `SubType` attribute of your `Graph` element to *Basic*, as in the example below.

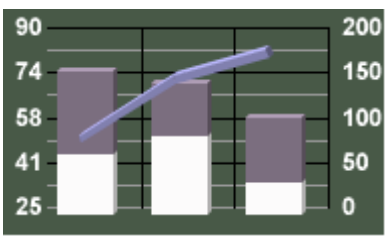
```
<Graph Name='graph' Type='Line Bar' SubType='Basic'>
```

LINE-STACKED BAR COMBO

The Line-Stacked Bar Combo subtype for Line Bar Combo graphs works much like a standard Line Bar Combo graph, except it uses stacked bars instead of bars. The two graphs below illustrate the difference:



Line Bar Combo



Line-Stacked Bar Combo

By default, in a Line-Stacked Bar Combo graph only the first data series is displayed as a line. All other data series are displayed as segments in the stacked bars.

To create a Line-Stacked Bar Combo graph in PopChart XML, you should set the `SubType` attribute of your `Graph` element to *Stacked*. The tag below shows how this is done.

```
<Graph Name='graph' Type='Line Bar' SubType='Stacked'>
```

For more information about stacked bars, refer to "Stacked Bar Graphs" on page 13-5.

FORMATTING OPTIONS

DUAL Y SCALES

One benefit of Line Bar Combo graphs is that they permit the bars and lines to have different value scales—one on the right side of the graph and one on the left. This is especially useful when you want to compare two data sets with different ranges. The category labels along the bottom of the graph will be the same for both the lines and the bars. However, the left and right scales can have the same or different ranges.

NUMBER OF LINES / BARS

You can also manually set the number of series that will be used for lines or bars. To do this, go to the General pane in the Graph Properties, check on Manually Set Number of Lines, and type the number of data series you wish to be displayed as lines in the Number of Lines box. The graph will then display the first *n* data series as lines, where *n* is the number you typed. All other data series will appear as bars.

AREA GRAPHS

Area Graphs can be used to:

- Displaying changes in cumulative value or percentage over time
- Comparing groups on outcome measurements
- Display group trends

Area Graphs combine the features of Line graphs (refer to "Line Graphs" on page 13-10) and Stacked Bar graphs (refer to "Stacked Bar Graphs" on page 13-5). They look like someone took a Line graph and colored in the areas underneath each line. But each of these area segments are stacked on top of each other, like in a Stacked Bar graph. Thus, the total size of the area at each category reflects the cumulative value of all data items in that category.

Each data series in the graph corresponds to a colored segment of the total area. The first series will always be the top area segment.

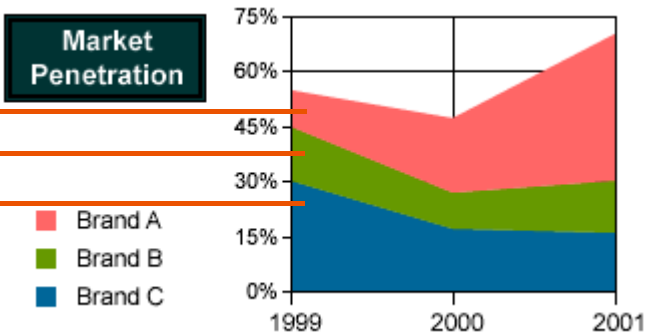
Example 13.9 below shows how Area graphs work.

Example 13.9 **Area Graph**

	1999	2000	2001
Brand A	10	20	40
Brand B	15	10	14
Brand C	30	17	16

Each value in this column is considered to be a data item in the **1999** *category*. They are summed together to compute the size of the area at that

Each value on this row is a data item in the **Brand C** *series* of data. The data items are represented by the blue area on the graph.



Area Graphs are useful when you want to show group trends in addition to comparing trends between individual series of data. For instance, Example 13.9 shows year-by-year trends in overall market penetration for a certain type of product. At the same time, it breaks down market penetration by brand, with each area segment revealing year-by-year market penetration trends for a different brand.

Area graphs are in the Standard Data Class (see Chapter 12 for details). To create an Area graph with PopChart XML, create a `Graph` element and set its `Type` attribute to *Area*, as shown in the example below.

```
<Graph Name='graph' Type='Area'>
```

STOCK GRAPHS

Stock graphs can be used for:

- Show daily stock prices over time
- Show statistical data with a confidence range over time

As the name of this graph type implies, the Stock graph type is useful for displaying information about stocks. Each data item is displayed as a stock bar, which can represent the following information: the high and low prices for the day and the opening and closing prices for the day. A stock graph is able to graph information about the high, low, opening and closing values of stocks. The look of the stock bar depends on which Stock graph subtype you are using.

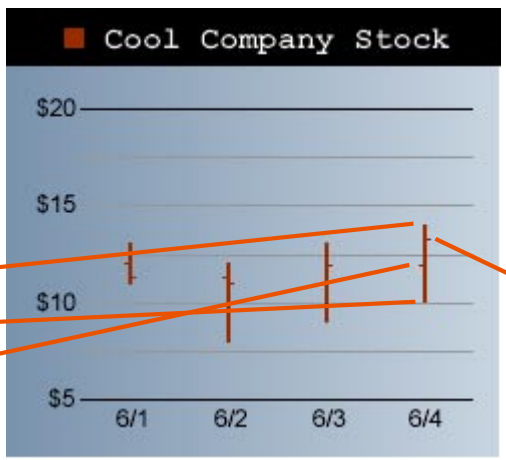
While there can be an infinite number of categories (usually dates) in a Stock graph, there is only one data series (usually representing a single stock), meaning that there is only one data item (stock bar) per category. Of: course, your graph could also use each category to represent the prices for a separate stock, which would allow you to compare prices for separate stocks on a single day.

Data items in stock bars must have at least two values: the high value and the low value. The open value and the close value may be optional, depending on the graph subtype. Example 13.10 below shows how a typical Stock graph works.

Example 13.10 **Stock Graph**

	Cool Company Stock			
6/1	13	11	12	11.3
6/2	12	8	11.3	11
6/3	13	9	11	11.9
6/4	14	10	11.9	13.2

The name of the stock appears in legend.



Stock's High Price

Stock's Low Price

The values in this row are used to create the data item in the **6/1** *category*. The data item translates into the stock bar for 6/1.

The opening price of the stock for the day, indicated by a left tick on the stock

The closing price of the stock for the day, indicated by a right tick on the stock bar.

Stock graphs are in the Stock Data Class (see Chapter 12 for details). To create a Stock graph with PopChart XML, create a `Graph` element and set its `Type` attribute to *Stock*, as shown in the example below.

```
<Graph Name='graph' Type='Stock'>
```

SUBTYPES

Listed below are the Stock graph subtypes.

HIGH-LOW/OPEN-CLOSE

The graph depicted in Example 13.10 is a High-Low/Open-Close graph. The bottom of the stock bars in a High-Low/Open-Close graph indicates the stock's low value. The top of the stock bar indicates the stock's high value. The tick on the left side of the bar indicates the stock's open value, and the tick on the right side of the bar indicates the stock's close value.

To create a High-Low/Open-Close graph in PopChart XML, you should set the `SubType` attribute of `Graph` to *HighLowOpenClose*. The tag below shows how this is done.

```
<Graph Name='graph' Type='Stock'
  SubType='HighLowOpenClose'>
```

HIGH-LOW

The High-Low subtype of Stock graphs is just like the High-Low/Open-Close subtype, only the open and close values are ignored. The stock bar will not have any ticks on it for the open and close values.

To create a High-Low/Open-Close graph in PopChart XML, you should set the `SubType` attribute of `Graph` to *HighLow*. The tag below shows how this is done.

```
<Graph Name='graph' Type='Stock'
  SubType='HighLowOpenClose'>
```

HIGH-LOW/CLOSE

The High-Low subtype of Stock graphs is just like the High-Low/Open-Close subtype, only the open value is missing. The stock bar will only show a close tick on the left.

To create a High-Low/Open-Close graph in PopChart XML, you should set the `SubType` attribute of `Graph` to *HighLowOpenClose*. The tag below shows how this is done.

```
<Graph Name='graph' Type='Stock'
  SubType='HighLowOpenClose'>
```

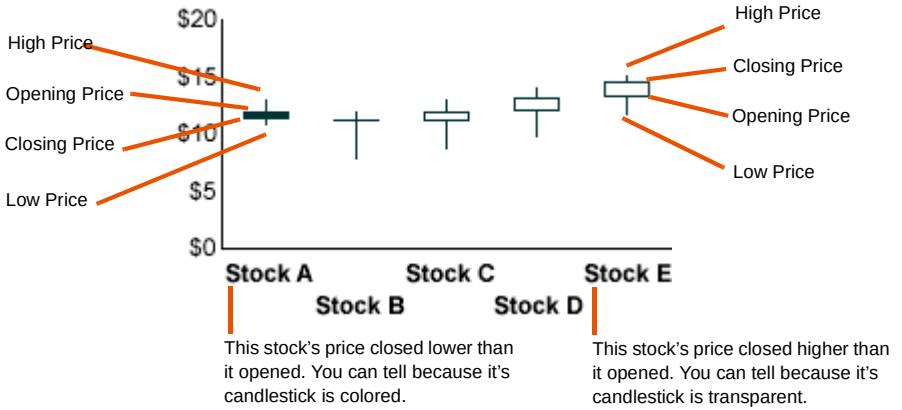
Then, you should make sure that you only send three data values for each data item. The third data value will be considered the close data value (instead of the open value, as it would normally be). So for example, in the following PCScript, all of the third data values are close data values.

```
graph.SetSeries(My Stock; 12,23,17; 20,25,21)
```

CANDLE STICK

The Candle Stick subtype of the Stock graph works as explained at the beginning of this section. However, it looks somewhat different. It uses candlesticks instead of stock bars. Candlesticks look like stock bars, except they don't use tick marks to indicate open/close information. Instead, they use a wide bar to show the difference between the opening and closing price. If the closing price is higher than the opening price (meaning the stock gained for the day), the bar will be transparent. If the closing price is lower than the opening price (meaning the stock lost for the day), the bar will be colored in.

The image below illustrates how a Candle Stick graph works.



To create a Candle Stick graph in PopChart XML, you should set the `SubType` attribute of `Graph` to *CandleStick*. The tag below shows how this is done.

```
<Graph Name='graph' Type='Stock' SubType='CandleStick'>
```

X-Y PLOT GRAPHS

X-Y Plot graphs can be used for:

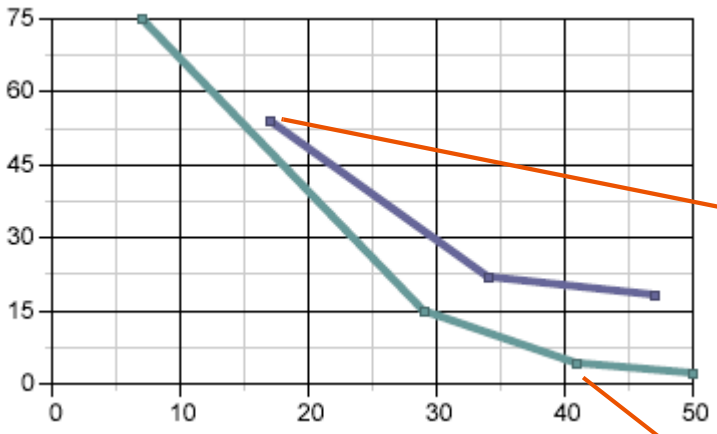
- Identifying relationships between large data sets
- Identifying trends in large data sets

If your data consists of X-Y data pairs, then you will want to use an X-Y Plot graph. X-Y Plot data items consist of two values: the x-value and the y-value, which represent the coordinates of that data item. The data item is displayed as a symbol. You can also add a line, bubble, or fill area for the data item. Because of the nature of the coordinate system, X-Y graphs do not have categories.

Example 13.11 below shows how an X-Y graph works.

Example 13.11

X-Y Plot Graph



Two data values are used for each data item: x (plotted along the x-axis) and y (plotted along the y-axis).

The data items in these columns are considered to be in the **Series 2** *series*. They will appear as teal dots and will be connected by a teal line.

Series 1			Series 2	
17	54		7	75
34	22		29	15
47	18		41	4
			50	2

This top line shows how the data item indicated by the upper left purple dot was resolved. The first value was used as the x coordinate. The second was used as the y coordinate.

The data item indicated by the teal dot that this bottom line points to was resolved similarly. The data point plotted was (*41, 4*).

NOTE: If you are using Plot data without a bubble value, you must still have a bubble column, but leave that column empty.

X-Y graphs are in the Plot Data Class (see Chapter 12 for details). To create an X-Y graph with PopChart XML, create a `Graph` element and set its `Type` attribute to *XY*, as shown in the example below.

```
<Graph Name='graph' Type='XY'>
```

SUBTYPES

Listed below are the X-Y graph subtypes.

X-Y LINE

The graph depicted in Example 13.10 is an X-Y Line graph. In an X-Y Line graph, all of the data items in a series are connected by the same line.

To create an X-Y Line graph in PopChart XML, you should set the `SubType` attribute of `Graph` to *Line*. The tag below shows how this is done.

```
<Graph Name='graph' Type='XY' SubType='Line'>
```

You will probably want to sort your data items when you use an X-Y Line graph, so that the line does not weave back and forth between data items. PopChart Server can do this automatically for you (refer to "Sorted Data" on page 13-23).

X-Y SCATTER

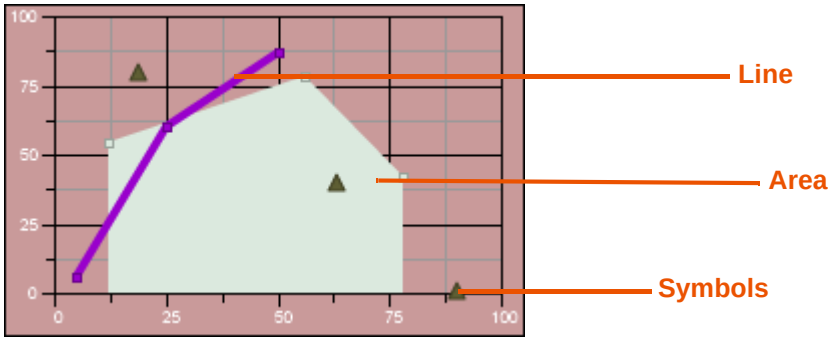
X-Y Scatter graphs are like X-Y Line graphs, except they have no lines. Only symbols will be plotted for each data item.

To create an X-Y Scatter graph in PopChart XML, you should set the `SubType` attribute of `Graph` to *Line*. The tag below shows how this is done.

```
<Graph Name='graph' Type='XY' SubType='Scatter'>
```

X-Y COMBO

X-Y Combo graphs allow you to display any data series with one or more of the following effects: symbols, lines, and fill area. The image below gives an example of an X-Y Combo graph.



Technically, you do not need to declare a graph as X-Y Combo—once you are outside of PopChart Builder you can add area fills, lines, or symbols to any X-Y graph. However, for

those wanting to be technically correct, you designate an X-Y graph as being X-Y Combo by setting the `SubType` attribute of `Graph` to *Combo*, as in the tag below.

```
<Graph Name='graph' Type='XY' SubType='Combo'>
```

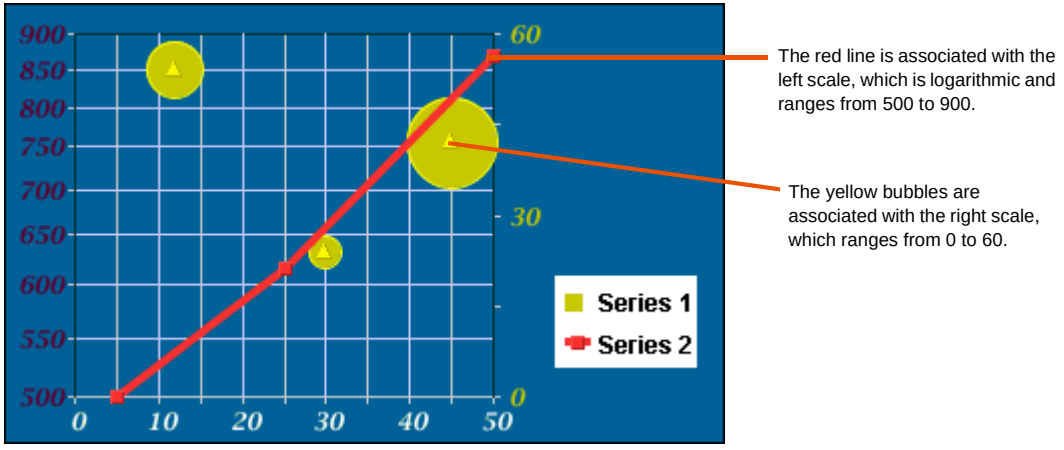
To add or change the effects for a data series, follow the instructions given in the "Formatting Options" section below.

X-Y COMBO (DUAL Y SCALES)

X-Y Combo graphs can also take advantage of dual y scales. When you use dual y scales, there will be scales on both sides of the graph. Each data series is associated with either the y scale to the left of the graph or the y scale to the right of the graph. The two y scales operate independently of each other.

This means, for example, that if all of your y values in the data series associated with the left scale are less than 60, while all of your y values in data series associated with the right scale are in the hundreds, the left scale will range from 0 to 60, while the right scale will be much larger. It's almost as if you were showing two separate graphs.

The image below shows how a dual y scale graph works.



To create a Dual Y Scale graph in PopChart XML, you should set the `SubType` attribute of `Graph` to *Dual Y*. The tag below shows how this is done.

```
<Graph Name='graph' Type='XY' SubType='Dual Y'>
```

To associate a series with the left or right y scale, you will need to override its `SeriesDefinition` element. You should set the `Scale` attribute for the SeriesDefinition to either *left* or *right*.

For example, suppose we want to associate the third data series in our graph with the right scale. Inside of our graph element, we use the following `SeriesDefinition` subelement.

```
<SeriesDefinition Number='3' Scale='right' />
```

For more information about overriding the PopChart XML elements, see "PCXML Transformations" on page 10-11 of the *PopChart Server User Guide*.

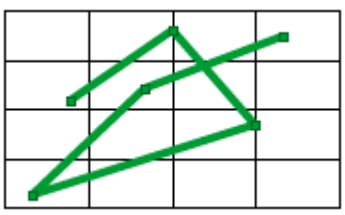
X-Y BUBBLE

The X-Y Bubble subtype is discussed in the "Bubble Graphs" section.

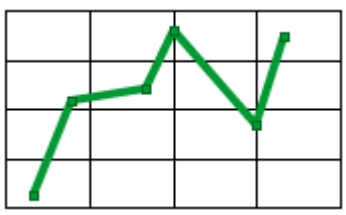
FORMATTING OPTIONS

SORTED DATA

The data points will usually be connected in the order that they are listed in the Data Editor. However, you can also connect them in the order of their x-coordinates. To do this, open up the Graph Properties and, in the General pane, check the Sort Data box.



Unsorted Data



Sorted Data

Warning: Do not use PopUp text when you've selected Sort Data. PopUp text will not work correctly.

TIME PLOT GRAPHS

Time Plot graphs can be used for:

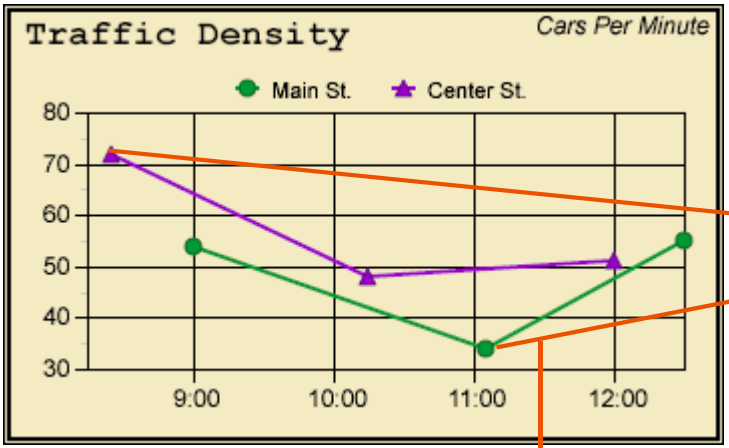
- Determining trends or cyclical variations
- Money distribution over time
- Production over time
- Price variation over time
- Environmental changes over time

If your data consists of Time-value data pairs, then you will want to use a Time Plot graph. Time Plot graphs work almost exactly like X-Y graphs. Time Plot data items consist of two data values: the time and the value, which translate to an x and y coordinate, respectively. The data item is displayed as a symbol. You can also add a line, bubble, or fill area for the data item. Because of the nature of the coordinate system, Time Plot graphs do not have categories.

Time graphs work well when you are graphing the value of something at indiscrete intervals—in other words, you are sampling the data at random times. If you are sampling your data at regular intervals, or if you sample all of your data series at the exact same times, you will probably find it easier to use a graph in the Standard Data Class, using each category of data to represent a discrete point in time.

Example 13.12 below shows how a Time Plot graph works.

Example 13.12 **Time Plot Graph**



Two data values are used for each data item: time (plotted along the x-axis) and value (plotted along the y-axis).

The data items in these columns are considered to be in the **Center St.** *series*. They will appear as purple triangles and will be connected by a purple line.

Main St.			Center St.	
9:00	54		8:25	73
11:05	34		10:15	48
12:30	55		12:00	51

This top line shows how the data item indicated by the upper left purple triangle was resolved. The first value was used as the time coordinate. The second was used as the value coordinate.

**The date input format for this graph was %H:%M (e.g. 8:00).*

The data item indicated by the green dot that this bottom line points to was resolved similarly. The data point plotted was (*10:15, 48*).

NOTE: If you are using Plot data without a bubble value, you must still have a bubble column, but leave that column empty.

Time Plot graphs are in the Plot Data Class (see Chapter 12 for details). The `time` data value, by default, should be in the following format: *%m/%d/%Y*, where *%m* represents the month number, *%d* represents the day number, and *%Y* represents the four-digit year. You can change the date input format to anything you like. For instance, Example 13.12 uses a *%H:%M* format (meaning *hour:minute*, where all times occur on the same day). Refer to "Date Input Format (Time Plot)" on page A-26 of the *PopChart Builder User Guide* for information about changing this format.

To create a Time Plot graph with PopChart XML, create a `Graph` element and set its `Type` attribute to *Time*, as shown in the example below.

```
<Graph Name='graph' Type='Time'>
```

SUBTYPES

Listed below are the Time graph subtypes.

TIME LINE

The graph depicted in Example 13.10 is a Time Line graph. In a Time Line graph, all of the data items in a series are connected by the same line.

To create a Time Line graph in PopChart XML, you should set the `SubType` attribute of `Graph` to *Line*. The tag below shows how this is done.

```
<Graph Name='graph' Type='Time' SubType='Line'>
```

You will probably want to sort your data items when you use an Time Line graph, so that the line does not weave back and forth between data items. PopChart Server can do this automatically for you (refer to "Sorted Data" on page 13-23).

TIME SCATTER

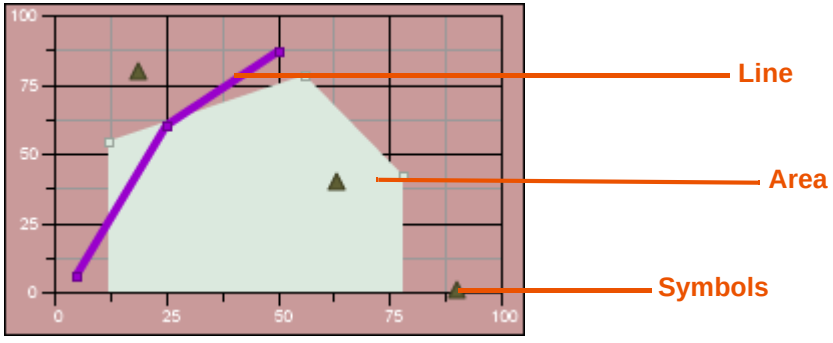
Time Scatter graphs are like Time Line graphs, except they have no lines. Only symbols will be plotted for each data item.

To create an Time Scatter graph in PopChart XML, you should set the `SubType` attribute of `Graph` to *Line*. The tag below shows how this is done.

```
<Graph Name='graph' Type='Time' SubType='Scatter'>
```

TIME COMBO

Time Combo graphs allow you to display any data series with one or more of the following effects: symbols, lines, and fill area. The image below gives an example of an X-Y Combo graph, which works almost exactly like the Time Combo graph.



Technically, you do not need to declare a graph as a Time Combo graph—once you are outside of PopChart Builder you can add area fills, lines, or symbols to any XY graph. However, for those wanting to be technically correct, you designate an Time graph as being Time Combo by setting the `SubType` attribute of `Graph` to *Combo*, as in the tag below.

```
<Graph Name='graph' Type='Time' SubType='Combo'>
```

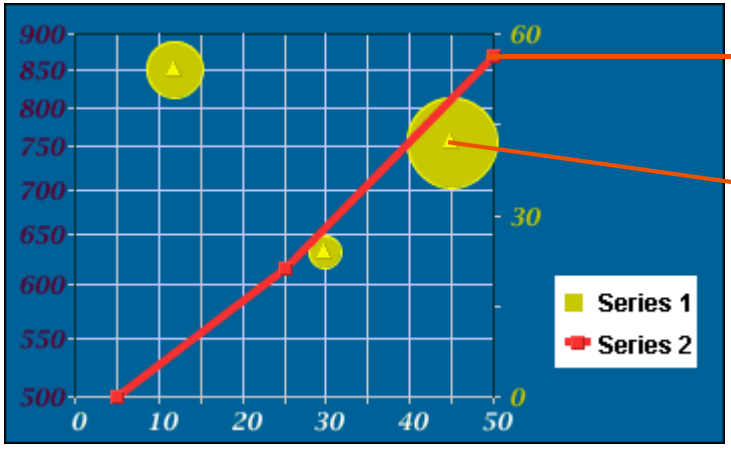
To add or change the effects for a data series, follow the instructions given in the "Formatting Options" section below.

TIME COMBO (DUAL Y SCALES)

Time Combo graphs can also take advantage of dual y scales. When you use dual y scales, there will be scales on both sides of the graph. Each data series is associated with either the y scale to the left of the graph or the y scale to the right of the graph. The two y scales operate independently of each other.

This means, for example, that if all of your y values in the data series associated with the left scale are less than 60, while all of your y values in data series associated with the right scale are in the hundreds, the left scale will range from 0 to 60, while the range of the right scale will be much larger. It's almost as if you were showing two separate graphs.

The image below shows how a dual y scale graph works. It is of an X-Y Combo graph, but the principle is still the same.



To create a Dual Y Scale graph in PopChart XML, you should set the `SubType` attribute of `Graph` to *Dual Y*. The tag below shows how this is done.

```
<Graph Name='graph' Type='XY' SubType='Dual Y'>
```

To associate a series with the left or right y scale, you will need to override its `SeriesDefinition` element. You should set the `Scale` attribute for the SeriesDefinition to either *Left* or *Right*.

For example, suppose we want to associate the third data series in our graph with the right scale. Inside of our graph element, we use the following `SeriesDefinition` subelement.

```
<SeriesDefinition Number='3' Scale='Right' />
```

TIME BUBBLE

The Time Bubble subtype is discussed in the "Bubble Graphs" section.

BUBBLE GRAPHS

Bubble graphs can be used to:

- Display data with three variables
- Display financial data

The Bubble graph is not actually a separate graph type. It is actually a subtype of all Plot graphs. In Bubble graphs, a third data value, called the `bubble` data value, is added to each data item. The `bubble` value is used to determine the size of a bubble that will appear at the data item's plot point.

A bubble graph can be used for three-dimensional data—that is, x-y data items that have a quantitative element to them. Example 13.13 below shows an X-Y graph works.

Example 13.13

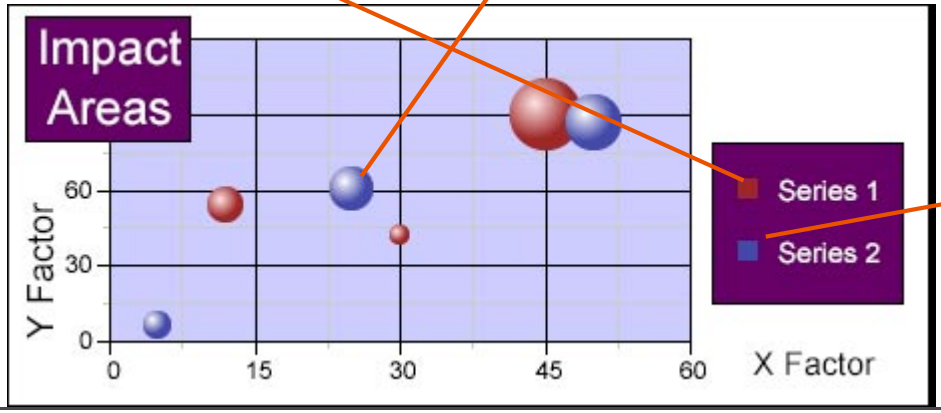
Bubble Graph

Series 1			Series 2		
12	54	5	5	6	4
45	90	10	25	60	6
30	42	3	50	87	8

The data in the second set of columns is used for the Series 2 series.

The data in the first three columns is used to form the data points in the Series 1 series.

The data values in these three cells form a single data item (bubble), which is plotted at 25 on the x-axis, 60 along the y-axis, and has a 6 pixel diameter.



SUBTYPE

Listed below are the different types of graphs that have the bubble subtype.

X-Y BUBBLE

The graph depicted in Example 13.13 is an X-Y Bubble graph (i.e. it uses X-Y Plot points).

To create an X-Y Bubble graph in PopChart XML, you should set the `Type` attribute of `Graph` to *X-Y*, and the `SubType` attribute to *Bubble*. The tag below shows how this is done.

```
<Graph Name='graph' Type='XY' SubType='Bubble'>
```

Be sure that each of your data items have a bubble value, as described in "Plot Data Class" on page 12-7.

TIME PLOT

A Time Plot Bubble graph uses Time Plot points instead of X-Y points.

To create a Time Plot Bubble graph in PopChart XML, you should set the `Type` attribute of `Graph` to *Time*, and the `SubType` attribute to *Bubble*. The tag below shows how this is done.

```
<Graph Name='graph' Type='Time' SubType='Bubble'>
```

Be sure that each of your data items have a bubble value, as described in "Plot Data Class" on page 12-7.

RADAR GRAPHS

Radar graphs can be used for:

- Mathematical and statistical applications

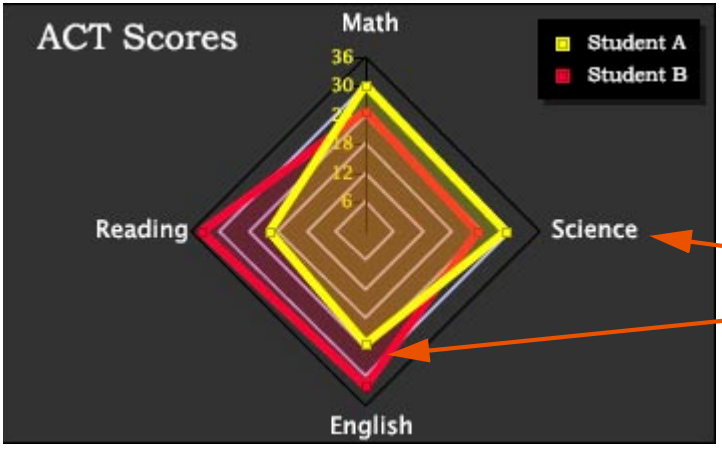
Radar graphs, for the most part, act like Line graphs. However, they are unique in that they use a radial grid to display data items. In a radial grid, the scale value grid lines circle around a central point, which represents zero. The further away a data item is from this center point, the higher its data value. The graph type is called Radar because it looks a lot like a radar screen.

The grid is not entirely circular, rather it is an equilateral polygon, with each category being plotted at a point of the polygon. Thus, if there are three categories, the grid is triangular. If there are eight categories, it will be octagonal. There must be at least three categories for a Radar graph to make sense.

Like in a line graph, all data items in a data series are connected by a line. However, since the grid is circular, the line segments come together to form a polygon. The area inside of the polygon for each series will be colored translucently. This first of all gives you an idea of the total "size" of a data series. Secondly, since it is translucent, you can still see the data series underneath the data series you are looking at.

Example 13.14 below shows a Radar graph works.

Example 13.14 **Radar Graph**



	Math	Science	English	Reading
Student A	30	29	23	20
Student B	25	23	32	34

The values on this row are data items in the **Student B** *series* of data. Each value represents the radius of the **Student B** (red) area for its particular *category.*

The values in this column are data items in the Science *category* of data. They will be plotted along the same compass point in the radar graph.

Radar graphs are in the Standard Data Class (see Chapter 12 for details). To create a Radar graph with PopChart XML, create a `Graph` element and set its `Type` attribute to *Radar*, as shown in the example below.

```
<Graph Name='graph' Type='Radar'>
```

PARETO GRAPHS

Pareto graphs can be used for:

- Prioritizing significant items among a large amount of data
- Determine production failure causes
- Determine best or worst performance
- Identify most of least important cost elements

Pareto graphs (pronounced *pa-ray-toe*) are single-series bar graphs, where the categories are ordered according to the size of their data item (largest to smallest). A line above the bars shows the cumulative value of all of the data items so far. A pareto graphs has two scales—one on the left which shows the actual data values, and one on the right which measures percentages of the total value of the data series.

Pareto graphs are useful for discerning the impact of certain categories of data on a data series. They allow you to quickly see which categories contributed the most (or least) to the entire series.

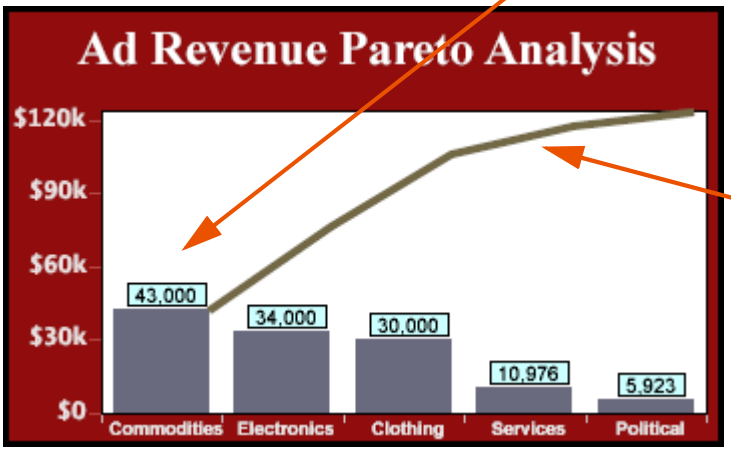
Example 13.15 below shows a Pareto graph works.

Example 13.15

Pareto Graph

	Electronics	Clothing	Commodities	Services	Political
Ad Revenue	34,000	30,000	43,000	10,976	5,923

The value in this column is a data item in the Commodities *category* of data.



The bars are sorted according to the largest data value.

This line graphs the cumulative value of all the data items so far.

Pareto graphs are in the Standard Data Class (see Chapter 12 for details), but they only graph one data series. To create a Pareto graph with PopChart XML, create a `Graph` element and set its `Type` attribute to *Pareto*, as shown in the example below.

```
<Graph Name='graph' Type='Pareto'>
```

FORMATTING OPTIONS

DATA LABELS

Data labels on a pareto graph work just like data labels fro any other graph (...). However, since the bars typically indicate a different type of value (e.g. dollars) than the line (percentage), you need to "trick" your appearance file into displaying a different symbol for the data labels along the line.

For example, suppose you set your data label format to *$%_VALUE*. A data label along the bar would look much like you would expect it to (e.g. *$43,000*). However, a data label along the line would unfortunately also include the dollar sign. So, wehereas you would want the data label along the line to be labelled as a percentage (e.g. *43%*), it would instead

use the same data label format as the bar (e.g. *$43*). This, of course, is probably not what you want to happen.

To work around this problem, first set your global data label string (select Graph Properties > Data Labels in PopChart Builder) to the format that you want data labels along the line to use (e.g. *%_VALUE%*). Next, use the series data label override (select Series Properties > Data Leabels in PopChart Builder) to change the data label format of the bars (e.g. *$%_VALUE*). This should give you the correct data label format both for the pareto line and bars.

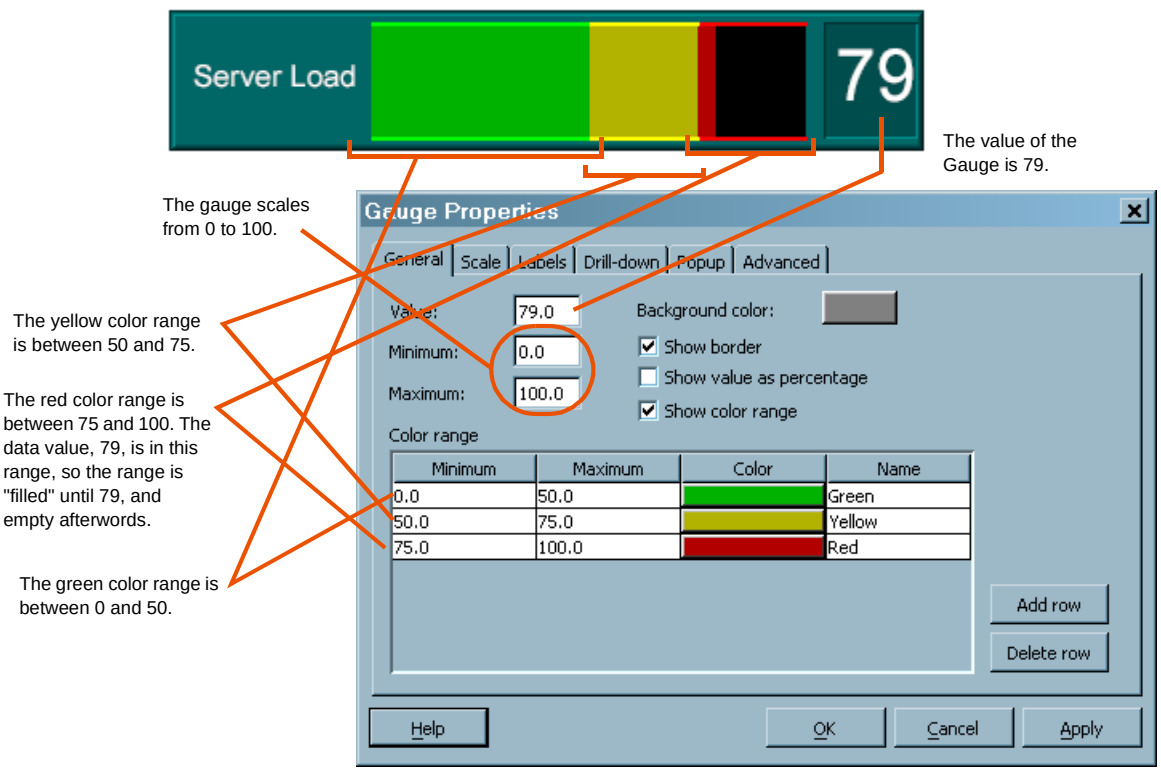
GAUGES

Gauges are used to indicate where a single data value falls within a predefined set of ranges. For example, suppose you had a gauge for a student's grade on an exam. The gauge might have the following set of ranges: A (90-100), B (80-90), C (70-80), D (60-70), and F (0-60). If the students score were 85, the gauge would indicate that the score was a B. Likewise, if the student's score were 54, the gauge would indicate that the score was in the F range.

Because gauges only graph a single data item, they are very different from other graphs (in fact, most people would say that they aren't graphs). You cannot, for example, import gauge data from a spreadsheet. The concepts of series and categories are also irrelevant.

PopChart XML and PCScript have special functions for setting gauge values and ranges (refer to "Gauge Data Class" on page 12-17). In PopChart Builder, you will be able to set these values in the Graph Properties dialog. Example 13.15 shows how the different elements of this dialog are used to form a gauge.

Example 13.16 Gauge



Note: *When you have overlapping ranges, the data value will be in the range whose minimum value is closest to the data value. For example, if you have a green range from 80 to 130, and you have a red range of 93 to 110, a data value of 94 would be considered to be in the red range. If the data value falls in a gap between ranges, the gauge will indicate that it was in no range.*

To create a Gauge with PopChart XML, create a `Graph` element and set its `Type` attribute to *Gauge*, as shown in the example below.

```
<Graph Name='graph' Type='Gauge'>
```

SUBTYPES

Listed below are the Gauge subtypes.

FILLED BAR

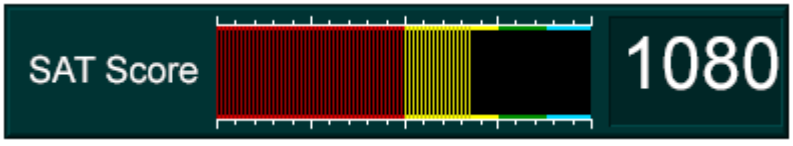
The gauge depicted in Example 13.15 is a Filled Bar Gauge. It uses a colored bar to indicate size of the data value. The bar is divided into different colored segments representing each color range. The data value is in the range indicated by the color of the last colored segment of the bar.

To create a Filled Bar Gauge in PopChart XML, you should set the `SubType` attribute of `Graph` to *Filled Bar*. The tag below shows how this is done.

```
<Graph Name='graph' Type='Gauge' SubType='Filled Bar'>
```

LED BAR

LED Bar Gauges look exactly like a Filled Bar Gauge, except the colored part of the bar is ridged or perforated so that it looks like an LED display. The image below shows an LED Bar Gauge.

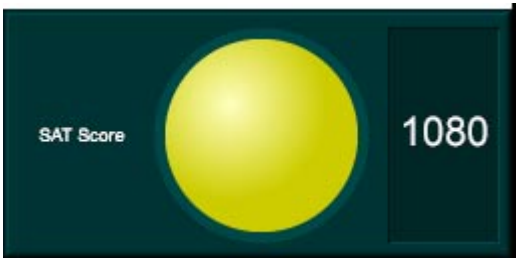


To Create an LED Bar Gauge in PopChart XML, you should set the `SubType` attribute of `Graph` to *LED Bar*. The tag below shows how this is done.

```
<Graph Name='graph' Type='Gauge' SubType='LED Bar'>
```

3D BULB

A 3D Bulb Gauge consists of a single three-dimensional bulb that is shaded in the color of of the range that the data value is in. It can also display a data value and a gauge label. The image below shows a 3D Bulb gauge.



To create a 3D Bulb Gauge in PopChart XML, you should set the `SubType` attribute of `Graph` to *3D Bulb*. The tag below shows how this is done.

```
<Graph Name='graph' Type='Gauge' SubType='3D Bulb'>
```

BULB

A regular Bulb Gauge is exactly like a 3D Bulb Gauge, except it is two dimensional. The image below shows a regular Bulb Gauge.

To create a Bulb Gauge in PopChart XML, you should set the `SubType` attribute of `Graph` to *Bulb*. The tag below shows how this is done.

```
<Graph Name='graph' Type='Gauge' SubType='Bulb'>
```

14

IMAGE FORMATS

With PopChart Server, you can generate PopChart images in the following formats: Macromedia® Flash™, SVG™, PNG, GIF, PDF, EPS, and WBMP. This section describes these formats in detail and tells you how to generate a PopChart image in each particular format. The last section of this chapter, "Comparison of Image Format Features," has a convenient table comparing all of the features of the different formats.

Note: *Some image formats will only be available in the PopChart Server Pro or PopChart Server Enterprise.*

Macromedia® Flash™

Available only in PopChart Server Pro and PopChart Server Enterprise.

Flash is a vector graphic image type developed by Macromedia and is widely used in Web sites. Flash images are small, load quickly and have interactive capabilities such as drill down, rollover and PopUp text. Flash images print at a high resolution. Another advantage is that a user can put a Flash image directly into a Microsoft PowerPoint® presentation.

The Flash viewer (Flash Player 3.0 or later) is required to view a Flash image. It is estimated that more than 96% of Internet users already have the proper Flash viewer.

More information about the Flash format is available from Macromedia's website at http://www.macromedia.com/software/flash/.

Note: *With* Best Image Fallback *you can have the user's web browser dynamically decide what image format to display. In this case, it will probably not matter if you choose Flash or SVG as the format of your PopChart Image. The user will see the PopChart image in GIF format if they don't have the Flash or SVG plug-in. For more information refer to "Best Image Fallback" on page 4-13 of the PopChart Server User Guide.*

EMBEDDING A FLASH IMAGE USING PopChart Embedder

You can generate a Flash PopChart image by setting the PopChart Embedder `imageType` attribute to "FLASH". For example, assuming your PopChart Embedder object is named *myPopChart*, you should use the following command to set the image format to Flash:

```
myPopChart.imageType = "FLASH";
```

Note: *FLASH images in Internet Explorer are embedded with an `<object>` tag. The `<object>` tag allows you to extend the functionality of the FLASH plug-in by including `<param>` tags. This provides support for effects such as transparency. To have the PopChart Embedder include a `<param>` tag within a FLASH image that it embeds, use the `addObjectParamTag(String, String)` method.*

EMBEDDING A FLASH IMAGE USING HTML

You can generate a FLASH PopChart image using the `@_FLASH` server command. For example, the following HTTP request will generate a FLASH PopChart image:

```
http://localhost:2001/?@_FILEexamples/apfiles/bar.pcxml@_FLASH
```

Flash images can be embedded into web pages using an <object> tag for Microsoft Internet Explorer browsers, and an <embed> tag for Netscape® browsers. You can use the @_FLASH server command to instruct PopChart Server to generate a Flash image.

The code below demonstrates how you can embed a Flash image. Note that by embedding the <embed> tag within the <object> tag, the code makes sure that the image will display correctly on both Microsoft Internet Explorer and Netscape.

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-
  444553540000"
  codebase="http://download.macromedia.com/pub/shockwa
  ve/cabs/flash/swflash.cab#version=4,0,0,0"
  width="600" height="400">
  <param name="MOVIE"
  value="http://localhost:2001/?@_FILEexamples/apfiles
  /bar.pcxml@_HEIGHT400@_WIDTH600@_FLASH">
  <embed type="application/x-shockwave-flash"
  pluginspage="http://www.macromedia.com/shockwave/dow
  nload/index.cgi?P1_Prod_Version=ShockwaveFlash"
  width="600" height="400"
  src="http://localhost:2001/?@_FILEexamples/apfiles/b
  ar.pcxml@_HEIGHT400@_WIDTH600@_FLASH">
  </embed></object>
```

You could also implement *Best Image Fallback* (i.e., an alternate image if the browser is incapable of downloading the Flash plug-in) to a GIF or PNG image by embedding an <img> tag within the <embed> tag. Refer to "URL Encoding" on page 11-8 of the *PopChart Server User Guide* for more information on embedding an image in the <img> tag.

Note: *For reasons that should be apparent, it is much more convenient to embed a FLASH image with the PopChart Embedder.*

SVG™

Available only in PopChart Server Pro and PopChart Server Enterprise.

SVG (Scalable Vector Graphics) is a new image format created and adopted by the W3C (World Wide Web Consortium) as the standard for vector graphic images. SVG is based on the powerful new XML (eXtensible Markup Language) created by W3C. SVG images can be zoomed in and out without losing any details and have smaller file size than do JPEG, GIF, Flash, or PNG images. SVG makes possible high-resolution printing, animation, drill down, rollover, and pop up text along with other special effects. SVG is an open standard.

Adobe is heavily promoting SVG and bundles the SVG viewer with Acrobat® 5.0. However, many users do not yet have the plug-in and will have to download it before they can view PopChart Images in the SVG format.

Note: *For the most part, plug-ins are very small and easy to download. Most web browsers will automatically start the download process when a user arrives at a page that requires the plug-in. Often times, the only thing a user will have to do is click a button.*

More information about the SVG format is available from Adobe's website at http://www.adobe.com/svg/.

Note: *With Best Image Fallback you can have the user's web browser dynamically decide what image format to display. In this case, it will probably not matter if you choose Flash or SVG as the format of your PopChart Image. The user will see the PopChart image in GIF format if they don't have the Flash or SVG plug-in. For more information refer to "Best Image Fallback" on page 4-13 of the PopChart Server User Guide.*

EMBEDDING AN SVG IMAGE USING PopChart Embedder

You can generate an SVG PopChart image by setting the PopChart Embedder `imageType` attribute to "SVG". For example, assuming your PopChart Embedder object is named *myPopChart*, you should use the following command to set the image format to SVG:

```
myPopChart.imageType = "SVG";
```

SVG TEMPLATES

You can have PopChart Server automatically embed your PopChart image within an SVG template and return the resulting SVG image. This allows you to create simple effects like animation, or to embed your dynamic images in more elaborate SVG documents. To do this, simply use the `svgTemplate` attribute of the PopChart Embedder, as illustrated below:

```
myPopChart.svgTemplate = "svg_template/Grow.svg";
```

EMBEDDING AN SVG IMAGE USING HTML

You can generate an SVG PopChart image using the @_SVG server command. For example, the following HTTP request will generate a SVG PopChart image:

```
http://localhost:2001/?@_FILEexamples/apfiles/bar.pcxm
  l@_SVG
```

SVG images can be embedded into web pages using an <embed> tag and the @_SVG server command. You must also be sure to specify the width and the height of the image in the <embed> tag.

```
<embed type="image/svg-xml"
  pluginspage="http://www.adobe.com/svg/viewer/install
  /main.html" width="600" height="400"
  src="http://localhost:2001/?@_FILEexamples/apfiles/b
  ar.pcxml@_HEIGHT400@_WIDTH600@_SVG">
  </embed>
```

SVG TEMPLATES

You can have PopChart Server automatically embed your PopChart image within an SVG template and return the resulting SVG image. This allows you to create simple effects like animation, or to embed your dynamic images in more elaborate SVG documents. To do this, simply use the @_USESVGTEMPLATE server command, as illustrated below:

```
http://localhost:2001/?@_FILEexamples/apfiles/bar.pcxm
  l@_HEIGHT400@_WIDTH600@_SVG@_USESVGTEMPLATEsvg_templ
  ates/Grow.svg
```

PNG

The PNG (Portable Network Graphics) format is a lossless image format similar to GIF, but higher in quality (they can contain more than 256 colors) and much smaller in file size. Their file size is slightly larger than FLASH and significantly larger than SVG, however their quality is significantly less than that of both formats. PNG images print only at a low resolution of 72 DPI.

In order to provide interactivity such as drill down and pop up text with PNG images, an image map must be generated. Thus, user interactivity is very limited. The availability of PopUp text will depend on how the browser displays alt text in an image map. Some browsers show it, but wait a few seconds before doing so. Others are not capable of showing PopUp text with PNG images at all.

The PNG format is supported in approximately 90% percent of all web browsers and does not require a plug-in. If your end users run Netscape 4.0 or higher, or Microsoft Internet Explorer 4.01 or higher, their browsers will support PNG images.

In PopChart Server Pro and PopChart Server Enterprise, PNG images are anti-aliased.

AUTOMATIC PNG DETECTION

One significant feature of PopChart Server is *Automatic PNG Detection*. PopChart Server can detect from a client's request whether or not a web browser will support PNG images. If a client request a GIF image, but supports PNG images, PopChart Server will will return a PNG image when this feature is on.

Unlike *Best Image Fallback* for Flash or SVG (see "Best Image Fallback" on page 4-13 of the *PopChart Server User Guide*), this process requires no additional HTML or PopChart Embedder code. As long as Automatic PNG Detection (see "Automatic PNG Detection" on page 3-27 of the *PopChart Server User Guide*) is enabled in the Administration Console, PopChart Server will automatically return the most appropriate image format.

EMBEDDING A PNG IMAGE USING PopChart Embedder

If *Automatic PNG Detection* is enabled, you will usually embed a PNG image by requesting a GIF image instead. However, you can explicitly request a PNG PopChart image by setting the PopChart Embedder `imageType` attribute to "PNG". For example, assuming your PopChart Embedder object is named *myPopChart*, you should use the following command to set the image format to PNG:

```
myPopChart.imageType = "PNG";
```

EMBEDDING A PNG IMAGE IN A WEB PAGE

If *Automatic PNG Detection* is enabled, you will usually embed a PNG image by requesting a GIF image instead. However, you can explicitly generate a PNG PopChart image using the `@_PNG` server command. For example, the following HTTP request will generate a PNG PopChart image:

```
http://localhost:2001/?@_FILEexamples/apfiles/bar.pcxml@_PNG
```

If you have enabled Automatic PNG Detection (see "Automatic PNG Detection" on page 3-27 of the *PopChart Server User Guide*), the following request would also create a PNG image, as long as the browser supports PNG:

```
http://localhost:2001/?@_FILEexamples/apfiles/bar.pcxml@_GIF
```

You can embed a PNG image in a web page simply by making it the source of an image tag. For example, the following image tag embeds the PNG image we created with the HTTP request above. It also provides an alternate text description of *My PopChart* in case the image loads slowly, or for the visually impaired.

```
<img src="http://localhost:2001/?@_FILEexamples/apfiles/bar.pcxml@_PNG" alt="My PopChart" />
```

GIF

GIF (Graphics Interchange Format) is a lossless image format that is almost universally supported. GIF images are larger in file size than SVG, Flash, or PNG, and thus take longer to send over the Internet. GIF images print only at a low resolution of 72 DPI.

In order to provide interactivity such as drill down and pop up text with GIF images, an image map must be generated. Thus, user interactivity is very limited. The availability of PopUp text will depend on how the browser displays alt text in an image map. Some browsers show it, but wait a few seconds before doing so. Others are not capable of showing PopUp text with GIF images at all.

PopChart uses the RLE algorithm for creating GIF images, avoiding any licensing issues regarding the LZW algoritm.

In PopChart Server Pro and PopChart Server Enterprise, GIF images are anti-aliased.

EMBEDDING A GIF IMAGE USING THE PopChart Embedder

Note: *If you have enabled Automatic PNG Detection (see "Automatic PNG Detection" on page 3-27 of the PopChart Server User Guide), this request will generate a PNG image on browsers that support PNG. You cannot override this functionality.*

You can generate a GIF PopChart image by setting the PopChart Embedder `imageType` attribute to "GIF". For example, assuming your PopChart Embedder object is named *myPopChart*, you should use the following command to set the image format to GIF:

```
myPopChart.imageType = "GIF";
```

EMBEDDING A GIF IMAGE IN A WEB PAGE

Note: *If you have enabled Automatic PNG Detection (see "Automatic PNG Detection" on page 3-27 of the PopChart Server User Guide), this request will generate a PNG image on browsers that support PNG. You cannot override this functionality.*

You can generate a GIF PopChart image using the `@_GIF` server command. For example, the following HTTP request will generate a GIF PopChart image:

```
http://localhost:2001/?@_FILEexamples/apfiles/bar.pcxm
  l@_GIF
```

You can embed a GIF image in a web page simply by making it the source of an image tag. For example, the following image tag embeds the GIF image we created with the HTTP

request above. It also provides an alternate text description of *My PopChart* in case the image loads slowly, or for the visually impaired.

```
<img src="http://localhost:2001/?@_FILEexamples/apfile
  s/bar.pcxml@_GIF" alt="My PopChart" />
```

PDF

Available only in PopChart Server Enterprise.

As its name implies, PDF (Portable Document Format) is more of a document format than an image format. Because of its high quality, relatively small file size (its size is comparable to Flash), and widespread use, it is an attractive format for displaying PopChart images.

PopChart images in the PDF format are somewhat limited. They offer no drill-down or PopUp capabilities, and can't display transparency. Another drawback is that PDF images are difficult to embed in web pages. Some browsers display the image in an embedded Acrobat® Reader® interface that may be distracting or confusing to some users. Other browsers can't embed PDF documents at all. Most of the time, you will want to link to the PDF image rather than embed it in a web page.

PDF documents may be viewed in a wide variety of viewers, including Adobe's free Acrobat Reader, included on the installation CD, or available from Adobe at http://www.adobe.com. PDF documents may be viewed in web browsers by using a plug-in (such as the one that is automatically included with Acrobat Reader). Such a plug-in is already installed on the large majority of web browsers.

EMBEDDING A PDF DOCUMENT USING THE PopChart Embedder

You can generate a PDF PopChart image by setting the PopChart Embedder `imageType` attribute to "PDF". For example, assuming your PopChart Embedder object is named *myPopChart*, you should use the following command to set the image format to PDF:

```
myPopChart.imageType = "PDF";
```

For the afore-mentioned reasons, most people don't embed PDF documents within web pages. Instead, they create PDF PopChart images that can be served by themselves or can be embedded into other PDF documents. If this is your intent, you will probably be interested in the `getImageData()` method, which allows you to access a byte array of the PDF image and even return a PDF image directly from a Java Servlet.

Another alternative is to save a PDF image for future viewing using either the `saveImageToAppServer(String, String)` or `saveImageToPopChartServer(String)` PopChart Embedder methods. For example, you could run a batch process to generate a PDF image at regular intervals, and then provide a link to it for users to download. If you save the PDF image to PopChart Server, you could load the image in a different session with the `loadServerSideImage(String)` method.

Many users use PopChart Server as a library to create PDF images for non-web applications. To learn how to create a PopChart image using PopChart Server as a library, contact the Corda Technologies support team (support@corda.com).

EMBEDDING A PDF DOCUMENT IN HTML

You can request that PopChart Server generate an image in the PDF format by using the `@_PDF` server command. For example, assuming PopChart Server is running on the local system with its default settings, the following HTTP request will generate an PDF PopChart document:

```
http://localhost:2001/?@_FILEexamples/apfiles/bar.pcxml@_PDF
```

If you attempt to browse to a PDF document—for example, entering the location above in the location bar of your browser—a plug-in, such as Acrobat® Reader®, will most likely start-up. Depending on your browser and the version of the plug-in, the plug-in may start as a separate program, or it may start within the browser window.

If you do not have the proper plug-in, you will be asked to specify a location to download the file to. In some instances, despite the fact that you don't have the plug-in, you may still have software capable of opening the downloaded document.

Most people don't embed PDF documents within a web page. Instead, they provide a link to the PDF document, which can be viewed separately. Example 14.1 shows how you might link to a PopChart PDF document.

Example 14.1 Linking to a PDF Document

```
<a href="http://localhost:2001/?@_FILEexamples/apfiles/bar.pcxml@_PDF">
Click here to view the graph in PDF format
</a>
```

When a viewer clicks on this link, the browser will either jump to the PDF document or, if the proper plug-in is not installed, download the document. If you are using the Acrobat Reader plug-in, you will see the Acrobat Reader menu above the PopChart image and just below the browser's menu and location bars. You will be able to navigate and save the document using this interface just as you would in Acrobat Reader.

If you want to embed the PopChart PDF document within your web page, you can try one of the following techniques. The first technique is to embed it using the `<object>` tag. To do this, simply specify your PopChart PDF document as the source to of this object (using the `src` attribute).

This technique will work in both Microsoft Internet Explorer 5.0 and up, as well as Netscape Navigator 4.08 and up. However, the results are slightly different. Assuming that you're using the Acrobat Reader plug-in, Internet Explorer will use the Acrobat Reader interface to show the PDF document, meaning that you will see Acrobat Reader menu and control bars as part of your "image." Netscape browsers, meanwhile, will show the entire PDF document as an image. There will be no menu or control bar; however, the user will be unable to resize or save the image.

Example 14.2 illustrates this technique.

Example 14.2 Using an `<object>` Tag To Embed a PDF Document

```
<object height="300" width="400"
                src="http://localhost:2001/?@_FILEexamples/apfiles/
                bar.pcxml@_PDF">
</object>
```

If you know that your user is using either Microsoft Internet Explorer 5.0 or above, or Netscape 6.1 or above, you can also "embed" the PDF document using an `<iframe>` tag. The `<iframe>` tag allows you to create a frame within the current viewing window. You can specify your PopChart PDF document as the source to of this frame (using the `src` attribute). If you are using the Acrobat Reader plug-in, this technique will result in the Acrobat Reader menu and control bars being shown within the frame, regardless of what browser you use.

Example 14.3 illustrates this technique. It also provides a link to the PDF document in case the user's browser does not support the `<iframe>` tag.

Example 14.3 Using an `<iframe>` Tag To Embed a PDF Document

```
<p>The graph in the PDF document below plots last month's sales.</p>
<iframe height="300" width="400"
                src="http://localhost:2001/?@_FILEexamples/apfiles/
                bar.pcxml@_PDF">
[If you see this text, your browser cannot display this graph inside
                of this document. Click <a
                href="http://localhost:2001/?@_FILEexamples/apfiles
                /bar.pcxml@_PDF">here</a> to view it separately.]
</iframe>
```

EPS

Available only in PopChart Server Enterprise.

EPS (Encapsulated PostScript) is a standard file format for importing and exporting PostScript files. A PopChart image in this format is essentially a single page PostScript file that describes the PopChart.

The EPS format is very similar to PDF, and consequently PopChart images in EPS format are of the same size as images in PDF. They are of very high quality, but like PDF, have no drill-down or PopUp capability, and can't display transparency.

The main purpose of an EPS file is to be included in other pages. EPS images can not be viewed from a web browser, and must instead be viewed from a graphics manipulation program, such as Adobe PhotoShop®. They cannot be embedded into a web page.

PopChart Server's EPS capabilities are not intended for most users. In other words, if you have a reason to produce PopChart images in the EPS format, you will probably know it. Otherwise, you will probably not need to concern yourself with this format.

EMBEDDING AN EPS IMAGE USING THE PopChart Embedder

EPS images cannot be embedded into a web page—they can only be downloaded and viewed with an external viewer. One common practice is to run a batch process to generate an EPS image at regular intervals, and then provide a link to it for users to download.

You can generate an EPS PopChart image by setting the PopChart Embedder `imageType` attribute to "EPS". For example, assuming your PopChart Embedder object is named *myPopChart*, you should use the following command to set the image format to EPS:

```
myPopChart.imageType = "EPS";
```

For the afore-mentioned reasons, most people don't embed EPS documents within web pages. Instead, they create EPS PopChart images that can be downloaded by themselves or can be embedded into other EPS documents. If this is your intent, you will probably be interested in the `getImageData()` method, which allows you to access a byte array of the EPS image and even return a EPS image directly from a Java Servlet.

Another alternative is to save an EPS image for future viewing using either the `saveImageToAppServer(String, String)` or `saveImageToPopChartServer(String)` PopChart Embedder methods. For example, you could run a batch process to generate an EPS image at regular intervals, and then provide a link to it for users to download. If you save the EPS image to PopChart Server, you could load the image in a different session with the `loadServerSideImage(String)` method.

Many users use PopChart Server as a library to create EPS images for non-web applications. To learn how to create a PopChart image using PopChart Server as a library, contact the Corda Technologies support team (support@corda.com).

EMBEDDING AN EPS IMAGE IN HTML

You can request that PopChart Server generate an image in the EPS format by using the @_EPS server command. For example, the following HTTP request will generate an EPS PopChart image:

```
http://localhost:2001/?@_FILEexamples/apfiles/bar.pcxm
  l@_EPS
```

If you attempt to browse to an EPS image—for example, entering the location above in the location bar of your browser—you will most likely be prompted to enter a location on your computer to download the file. This is because you cannot view it in a browser. You will need to save it (be sure to give it a .eps extension) and then open it up in a suitable graphics program.

EPS images cannot be embedded into a web page. Instead, most people will provide a link to the EPS image, which can be viewed separately. Example 14.4 shows how you might link to an EPS image.

Example 14.4 Linking to an EPS Image

```
<a href="http://localhost:2001/?@_FILEexamples/apfiles/bar.pcxml@_E
                PS">
Click here to view the graph in EPS format
</a>
```

When a viewer clicks on this link, the browser will download an EPS file containing your PopChart image.

If you want to be able to view an EPS PopChart image in any context other than as a downloaded file, you will need to set up your own proprietary system. If this is your intention, you will probably want to generate your image using the PopChart Embedder library method, as explained above.

WBMP

Available only in PopChart Server Pro and PopChart Server Enterprise.

WBMP stands for Wireless BitMaP. It is the standard image format for wireless devices that use WAP (Wireless Application Protocol). WBMP's are uncompressed, black and white bitmaps that are intended for use on devices with small screens and limited bandwidth connections.

When you design PopChart images meant to be displayed in wireless browsers, it is important to remember several things. First of all, WBMP images will appear in only two colors. *Everything* that is colored will appear black. Secondly, you should design your appearance file to be small, as scaling larger appearance files down to a size suitable for wireless browsers will likely result in distorted fonts and images.Typically, you will want to limit the size of your appearance file (and PopChart image) to 96x48 pixels.

Because PopChart images in the WBMP format often appear very different than images in other formats, previewing your images in a wireless browser should be an integral part of your design process.

EMBEDDING A WBMP IMAGE USING THE PopChart Embedder

You can generate a WBMP PopChart image by setting the PopChart Embedder `imageType` attribute to "WBMP". For example, assuming your PopChart Embedder object is named *myPopChart*, you should use the following command to set the image format to WBMP:

```
myPopChart.imageType = "WBMP";
```

EMBEDDING A WBMP IMAGE IN WML

You can generate a WBMP PopChart image using the `@_WBMP` server command. For example, the following HTTP request will generate a WBMP PopChart image:

```
http://localhost:2001/?@_FILEexamples/apfiles/bar.pcxml@_WBMP
```

You can embed a WBMP image in WML simply by making it the source of an image tag. For example, the following image tag embeds the WBMP image we created with the HTTP request above. It also provides a description of the image (*My PopChart*) in case the browser cannot display images.

```
<img src="http://localhost:2001/?@_FILEexamples/apfiles/bar.pcxml@_WBMP" alt="My PopChart" />
```

COMPARISON OF IMAGE FORMAT FEATURES

This table offers you a side-by-side comparison of image format features:

	Macromedia® Flash™	SVG™	PNG	GIF	PDF	EPS	WBMP
Availability	Pro, Enterprise	Pro, Enterprise	All	All	Enterprise	Enterprise	Pro, Enterprise
Typical Image Size	10 KB	4 KB	10-15 KB	30-40 KB	5-20 KB	5-20 KB	.5-2 KB[a]
High Resolution Printing	Yes	Yes	No (72 DPI)	No (72 DPI)	Yes	Yes	No
Drill-down Text	Yes	Yes	Yes[b]	Yes[c]	No	No	No
PopUp Text	Yes	Yes	Limited	Limited	No	No	No
Rollover Text	Yes	Yes	No	No	No	No	No
Browser Support[d]	96%	Limited	90%	100%	Most Browsers	None	100%[e]

a. Since they are typically smaller in size than other image types and only support 2 colors, WBMP images will naturally have a significantly smaller size.

b. Drill-down for PNG is supported via image maps and will have limited feedback.

c. Drill-down for GIF is supported via image maps and will have limited feedback.

d. With Best Image Fallback (see "Best Image Fallback" on page 4-13 of the *PopChart Server User Guide*), browser support is irrelevant for Flash or SVG, as the browser will automatically display the PopChart image in the highest quality format available. Similarly, PopChart Server's Automatic PNG Detection (see "Automatic PNG Detection" on page 3-27 of the *PopChart Server User Guide*) feature makes browser support irrelevant for PNG images.

e. WBMP images are supported in 100% of WAP browsers that can actually display images. They are not supported in desktop browsers.

TABLE OF MIME TYPES

The table below lists the mime types for the image formats that PopChart Server supports.

Format	mime type
GIF	image/gif
PNG	image/png
SVG	image/svg
FLASH	application/x-shockwave-flash
PDF	application/pdf
EPS	application/postscript
WBMP	image/vnd.wap.wbmp