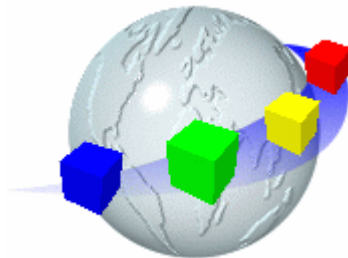


WebCab ScrollArea v2.0

WebCab
Components



Preface

The aim of this guide is to provide clear and concise guidance of all aspects of WebCab's ScrollArea v2.0 JavaBeans component that are likely to be encountered on a day-to-day basis by developers, administrators and end users of the component. ScrollArea v2.0 is a Graphical User Interface Text and Image area that supports extensive customization of colors, font characteristics, World Wide Web access, pictures and backgrounds. This component is developed using the fully browser compatible Java Development Kit v1.1 packages and hence provides excellent portability between operating systems.

This documentation contains a detailed description of the component packages content, functionality and applicability. An overview of the JavaBeans technology and its installation and use within IDEs is illustrated. Section three forms the Programmers guide which is the main body of this documentation, giving the developer a road map for taking advantage of every feature and functionality of the component. Then some examples are given which illustrate how the features which are detailed within the programmers guide can be applied in practice. In the next chapter we provide a UML diagram of the ScrollArea Component. Finally, we introduce WebCab Components, its philosophy and approach to serving the Java development community with robust and powerful J2EE applications.

We hope that this documentation helps us to help you further with your development needs. If you have any comments on how we can improve this documentation or our component then any feedback is always gratefully received.

Good luck with your project and thank you for your interest in our component.

The WebCab Components team

Contents

Preface	i
1 Introduction	1
1.1 Product Description	1
1.1.1 Details	1
1.2 Prerequisites and Compatibility	2
1.2.1 System Requirements	2
1.2.2 Compatibility	3
2 JavaBeansGuide	4
2.1 Overview	4
2.2 Installing the JavaBeans	4
2.3 Using JavaBeans inside an IDE	5
2.3.1 Importing the <i>JAR</i> file into Sun's One IDE	5
2.3.2 Importing the <i>JAR</i> file into Borland's JBuilder IDE	5
2.4 Developer Support	6
2.4.1 Documentation	6
2.4.2 Compilation and Custom Modification of Source Code	6
2.4.3 Online Support	7
3 Programmer's Guide	8
3.1 Getting Started	8
3.2 Introducing WebCab ScrollArea v2.0	8
3.3 WebCab ScrollArea v2.0 Structure	9
3.3.1 The Developer	9
3.3.2 The Administrator	9
3.3.3 The User	9
3.4 Developer's Guide to WebCab ScrollArea v2.0	10
3.4.1 The <i>ScrollArea</i> JavaBeans TM component	10
3.4.2 The <i>TextScroll</i> JavaBeans TM component	18
3.4.3 The Color Picker Java bean	20
3.4.4 ScrollArea Listeners	21
3.5 Administrator's Guide	21
3.5.1 Background Pictures	21
3.5.2 Colors	22
3.5.3 Fonts	24
3.5.4 Images	26
3.5.5 Miscellaneous	27
3.6 User's Guide	28

4	Examples	29
5	ScrollArea UML Diagram	33
5.1	Diagram	34
6	Guide to WebCab Components	35
6.1	The Company	35
6.2	Presentation of Products	35
6.3	Supported Clients, IDEs, Containers and DBMSs	35
6.4	Transparent Functionality	36
6.5	Code Conventions	36
6.6	Company Culture and Activity	36
6.7	Product Life cycle	36
6.8	Support, Warranty and Upgrades	36

Chapter 1

Introduction

1.1 Product Description

WebCab ScrollArea v2.0 is a Graphical User Interface Text & Image area, that supports extensive customization of colors, font characteristics, World Wide Web access, pictures and backgrounds, all programmed on a solid and fully browser-compatible 1.1 Java Development Kit. Without requiring to be run on a Java2 Virtual Machine, with a JDK1.3 Runtime Environment, WebCab ScrollArea v2.0 provides excellent portability between any operating systems running JavaTM.

The main features of WebCab ScrollArea v2.0 refer to text properties, such as colors, font sizes, styles and attributes, a new easy-to-use color picker, excellent smooth scrolling speed and reliability. Also, the very precise and detailed configuration options, both on the user and programmer side, provide portability between different platforms according to client-side specific needs.

1.1.1 Details

All of the WebCab ScrollArea v2.0 component features have been designed to work on a JDK1.1 platform, being compatible with virtually any browser running a Java Virtual Machine. The WebCab ScrollArea v2.0 characteristics are:

- Text Options
 - Colors - Every word or character may bear a different color, defined either by color picking, by choosing from a predefined list of colors or by specific RGB (red, green and blue) 24-bit color definition.
 - Font Faces - The standard Java AWT supports a limited range of Font Families, such as Courier, Dialog, Helvetica, Serif and Monospaced. These and many other fonts can be chosen by the user and by the developer, inside the ScrollArea Configuration Dialog, e.g “Lucida Sans”, “Bitstream Charter”, “Courier 10 Pitch”, “Utopia”, “Arial”, “Arial Narrow”, “Arial Black”, “Bookman Old Style”, “Impact”, “Times New Roman”.
 - Font Styles - WebCab ScrollArea not only supports the standard bold and italic styles as provided by the AWT package, but also is capable of rendering the underline style for sequences of characters.

- **Font Sizes** - The developer or client may choose the font sizes from the ScrollArea Configuration Dialog and inside any sequence of characters, as well.
- **Image Options**
 - **Background Pictures** - WebCab ScrollArea v2.0 supports any JPEG or GIF file image to be displayed on the background of the rendering surface. The user may choose between three types of arrangement, centered, tiled and stretched from the Configuration Dialog.
 - **Images** - Both images and text can be displayed inside the rendering surface. The images are selected by the developer or designer and displayed by the end-user at runtime.
- **Configuration Options**
 - **Configuration Dialog** - By clicking with the right or middle mouse button on the desktop the user can enter the ScrollArea Configuration Dialog. From there, using only the mouse, the user may choose any font, color and image setting.
 - **Init Configuration** - This configuration file is administrator and developer specific, allowing detailed customization of WebCab ScrollArea v2.0 behavior.
- **Miscellaneous**
 - **Text Word Wrapping** - The user may choose to wrap the words so as to fit the visible rendering surface. This way, scrolling may be avoided when attempting to view the entire text.
 - **Smooth Scrolling** - WebCab ScrollArea v2.0 allows a smooth pixel-by-pixel scrolling when dragging the scrollbar “bubble”, ensuring comfortable text and image viewing.
 - **List-like Behavior** - As an alternative, ScrollArea may behave like a selection list with the text gliding from top to bottom. Also in this mode each line may be selected allowing proper events to be cast.
 - **Text Flowing** - While scrolling through the text, in any direction, horizontally or vertically, the text and images are moved across the desktop, so that the background images remain static.

1.2 Prerequisites and Compatibility

1.2.1 System Requirements

- An Operating System running Java
- Pentium II 500 MHz
- 64MB RAM

1.2.2 Compatibility

Operating System for Deployment

- Windows XP, 2000, 2003, NT, 9x
- Sun Solaris
- Linux
- IBM AIX
- HP-UX

Component Type

- JavaBeans

Built Using

- Java2 SDK Standard Edition 1.3.1

Compatible IDE's

- Borland JBuilder
- Sun's ONE IDE (formerly Forte for Java)
- Eclipse
- BEA Studio
- Oracle JDeveloper

Chapter 2

JavaBeansGuide

2.1 Overview

The JavaBeans technology offers a new perspective on how to divide the labor between software developers and graphic designers. It provides mechanisms that both minimize the design and implementation effort for modest system upgrades while fully supporting the design, implementation, and assembly of enterprise wide software solutions.

The distinguishing feature of JavaBeans components as reusable components is that they operate across all platforms supported by Java. This means that any JavaBeans component will work inside any operating system running a Java Virtual Machine, or even inside a browser supporting Java applets.

2.2 Installing the JavaBeans

This JavaBeans suite contains the following files:

- Documentation
 - Product Details
 - * Product Description
 - * System Requirements
 - * License Agreement
 - JavaDocs Files
 - JavaBeans Guide
 - Guide to WebCab Components
- JavaBeans Java Archive
- Examples and Demos

The documentation provided includes the description for the product, the *java generated documentation* with precise explanations for each *class*, *method* and *field*. The beans come wrapped into an archive with a *JAR* extension. This file contains the whole directory structure of the beans, additional classes and serialized components and a *manifest file*. The manifest file is required by IDEs to tell the Java beans from the other classes in

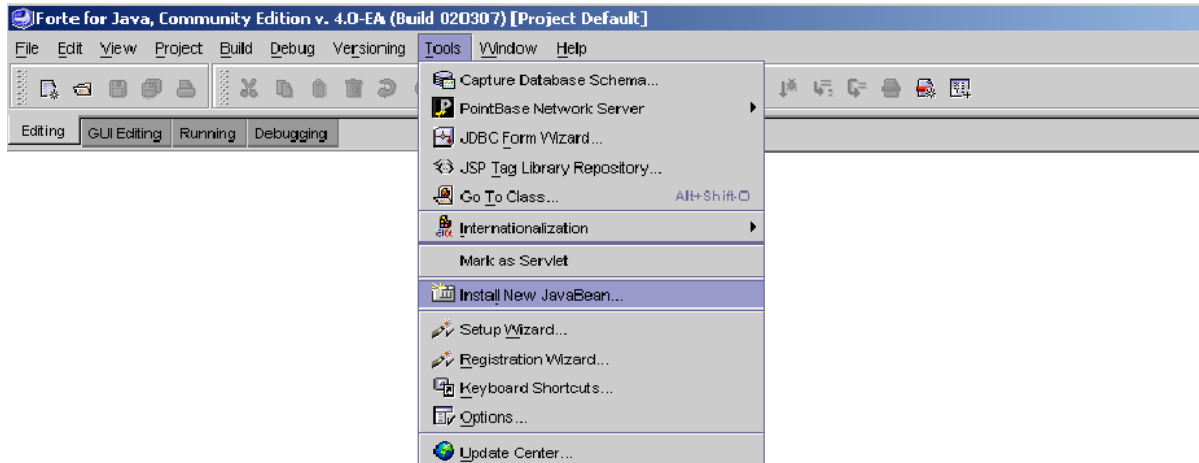
the archive. It is located in *META-INF/MANIFEST.MF*. After identifying each of these components, you may take the *JAR* file and import it into any compatible IDE.

2.3 Using JavaBeans inside an IDE

By importing the *JAR* file into a JavaBeans independent development environment (IDE), you can start adding the beans into your working area. As soon as you do this, you will be given the opportunity to edit their settings in a properties window. This is where attributes such as *font styles*, *colors*, *on/off switches*, *titles* can be fully customized, allowing the designer to control the design aspect of the bean without worrying about its implementation.

2.3.1 Importing the *JAR* file into Sun's One IDE

To make the *JAR* file available to the Sun's One IDE (formerly Forte For Java), first start up the IDE and select 'Tools > Instant New JavaBean'. Now select the *JAR* file which contains the JavaBeans by browsing the local file directory within the pop-up window and click OK. Another pop-up window will display the names of the JavaBeans contained within the *JAR* file. Select the beans you wish to install and click OK. In the next window please select the 'Palette Category' you wish to add these JavaBeans to and click OK. This completes the installation of the JavaBean component to the Palette from which the beans can be used within future application development.

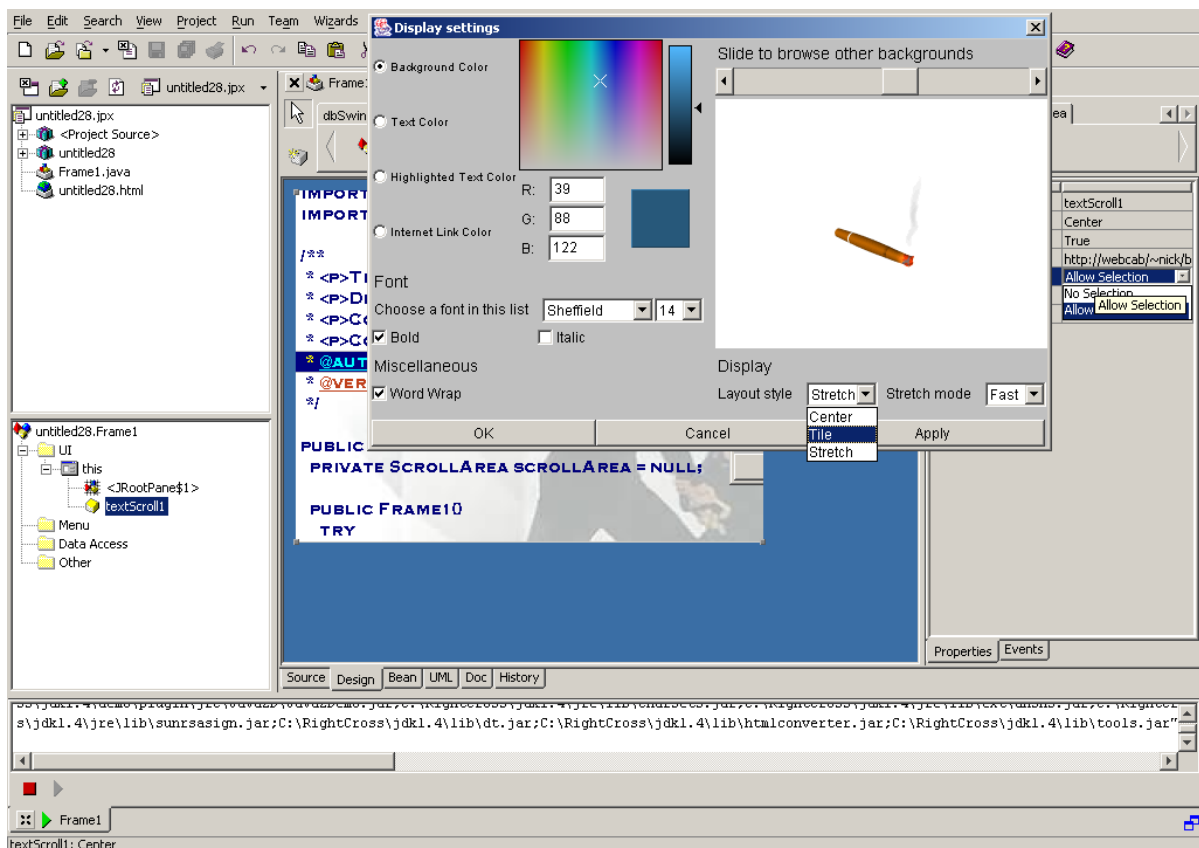


2.3.2 Importing the *JAR* file into Borland's JBuilder IDE

To import the *JAR* file containing the JavaBean components into a JBuilder project select from the menu 'Tools > Configure Palette'. Now a pop-up window will appear which has two tags, 'Pages' and 'Add Component'. You can either choose to create a page for the imported JavaBeans (i.e. *JAR*) or they may be added to an existing page. To create a new page click on 'Add' in 'Pages', chosen a name and click OK. To add the *JAR* file first select the 'Add Components' tab and then click on 'Select Library'. Another window will pop-up entitled 'Select a different library', if the *JAR* file is not displayed in the 'user home' folder then click the 'New' button. Now within the pop-up entitled 'New Library Wizard' enter the name you wish to give the imported library of JavaBeans contained

within the *JAR* and click ‘Add’. Select the *JAR* file by browsing your local file directory in the pop-up window and then click OK. You should have now been returned to the ‘New Library Wizard’ with the *JAR* file location inserted within the ‘Library Paths’, if this is the case click OK, otherwise click ‘Add’ and repeat the previous step. Select the *JAR*’s library name within the ‘Select a different library’ window and click OK. To finish click the ‘Add from Selected Library’ button within the ‘Palette Properties’ window. The results of the import will be displayed within a ‘Results’ window which completes the imported onto the JBuilder Palette.

Below is a screen shot of working with ScrollArea inside the JBuilder software.



2.4 Developer Support

2.4.1 Documentation

In order to use the JavaBeans component in any application, you should first read the information provided in the *Product Description* and *System Requirements*. As soon as you are aware of any incompatibilities that might turn up, you may start browsing the *Java Generated Documentation* (JavaDocs) to understand the structure of each bean.

2.4.2 Compilation and Custom Modification of Source Code

All WebCab Components are compiled using Sun’s JDK1.3, and should you prefer compilation of the source code to be done using another compiler then please contact us. Some functions (for example standard mathematical operations) are embedded within the source

code, thus, if you license classes which you believe offer a similar level of functionality from a third party and wish to include these classes within our components then please contact us. Both of the above services are offered free of charge at WebCab's discretion.

2.4.3 Online Support

Should you have any questions or queries, please feel free to email us at:

support@WebCabComponents.com

For updates and new releases, visit:

www.WebCabComponents.com

Chapter 3

Programmer's Guide

This section will explain every detail and aspect a programmer needs to know in order to fully exploit the features and capabilities of the WebCab ScrollArea v2.0 JavaBeansTM component. Here, you will be able to understand the principles by which WebCab ScrollArea v2.0 works, how to best accommodate them and how to make every function count.

3.1 Getting Started

WebCab ScrollArea v2.0 is a JavaBeans component and because of this, it is meant to be used as a part of a Java-based application. There are at least two ways of embedding it inside an application; one is by deploying the JavaBean inside a Development Environment, such as Borland's JBuilder or Eclipse, the other way is by direct implementation. Deployment inside a Development Environment is easy-to-use, comprehensive and does not require intricate knowledge about Java as a programming language. Yet, in order to entirely benefit the advantages a Java bean can offer, the developer will need to acquire a deeper understanding of the ways it works.

This information can be obtained from this PDF documentation, the "README" files located inside the package structure, and this products JavaDocs. After getting acquainted with the basics of WebCab ScrollArea v2.0, you will be - almost exclusively - using the JavaDocs, where details concerning all classes, their fields and methods is described.

3.2 Introducing WebCab ScrollArea v2.0

WebCab ScrollArea v2.0 JavaBeansTM component is a Graphical User Interface (GUI) area, used for displaying text and image information. The information inside the rendering area consists of lines of text, where every word or character may be customized to look and behave independent of each other. A simple concept standing behind the making of WebCab ScrollArea v2.0 is the standard line-by-line text area, that can display more than one line at a time and scroll the text, both horizontally and vertically.

The needs for more complex rendering of text and even images, without satisfaction from Java's AWT, determined the creation of another component, specifically designed for this purpose. Even though similar components can be found in Sun's Java2 Swing package, they are quite slow and cannot run inside a browser without a proper plugin. Since In-

ternet Chat programs are great consumers of such components, it is vital that portability be kept between different browsers and platforms. This is the main reason why WebCab ScrollArea v2.0 is fully JDK1.1 compliant, allowing easy integration inside an applet that complies with the same JDK1.1 specifications.

3.3 WebCab ScrollArea v2.0 Structure

WebCab ScrollArea v2.0 is a *three-leveled* JavaBeans component. The first level is represented by the developer, who implements the Java bean inside a Java application or applet. The developer configures the Java bean so as to run with standard settings, as specified by the administrator, who represents the second level. The administrator configures the settings for the end-user, the last party, also able to personalize certain features.

3.3.1 The Developer

Implementing the JavaBeansTM component inside a Java-based application will be conceived and accomplished by a developer or group of developers. Before starting the actual development, it is vital that the final purpose of the product be defined and that the exact relationship between the final product and the Java component be established. In order to start using the component, the Java bean will first be instantiated and subsequent calls to its methods will be performed. For detailed description about each method, you may help yourself by browsing the JavaDocs Generated Documentation files. For example, the central JavaBeansTM component is `com.webcab.chat.gui.ScrollArea`. Developer specific information will be found in the “Developer’s Guide” section.

3.3.2 The Administrator

While embedding WebCab ScrollArea v2.0 inside a Java-based application, the developer may choose to allow external configuration of the JavaBeans component’s behavior. The great advantage is that such an external configuration does not require any kind of knowledge regarding product development or technical details. The party that is responsible with the non-technical customization of WebCab ScrollArea v2.0 is the administrator. The administrator will take care of the personalization of the bean and will be the direct connection between the component and the end-user. The administrator specific informations are found in the “Administrator’s Guide” section.

3.3.3 The User

The user is the most important of the three parties, since it directly interferes with all the features that the product offers to him. This means that the user should be given the opportunity to personalize certain functions, in a similar manner to the way a developer or an administrator are allowed to. In order to provide a compact solution, WebCab ScrollArea v2.0 supplies such means of personalization, by allowing the end-user to graphically configure the component’s behavior at run-time. All details on how these configurations are performed are fully described in the “User’s Guide” section.

3.4 Developer's Guide to WebCab ScrollArea v2.0

The WebCab ScrollArea v2.0 JavaBeansTM component comprises several public Java class files, each of them being described in the following subsections.

3.4.1 The *ScrollArea* JavaBeansTM component

The “ScrollArea” class is where every programmable function can be configured by the developer, who may avail himself, or herself, of the methods and fields appointed to each individual feature. Every field and method described in these sections are public and can be used inside any Java-based application without discrimination.

Before starting to implement WebCab ScrollArea v2.0 in an application, you will instantiate the JavaBeans component with the following code:

```
import java.beans.*;
import com.webcab.chat.gui.*;

...

try {
    ScrollArea scrollArea = (ScrollArea) Beans.instantiate (
        getClass ().getClassLoader (), "com.webcab.chat.gui.ScrollArea");
}
catch (java.io.IOException e) {
}
catch (java.lang.ClassNotFoundException e) {
}
```

After the instantiation, all functions become available for later development.

In order to dispose all related resources, after having finished using the bean, invoke the following method from the *ScrollArea* class:

- void destroy ()

§ Adding and Removing Text

Inside the rendering surface (working area), the text is displayed line-by-line, and can be subsequently altered and removed. Every line of text is a **java.lang.String** type object and may contain attributes control sequences, as defined by the administrator.

In order to add a line of text to the rendering surface, a developer may invoke any of the methods:

- void add (String item)
- void addItem (String item)
- void addString (String item),

where `item` represents the text line. Also, you may need to use the following command in order to later modify any existing lines:

```
void setItemAt (String line, int index)
```

In order to retrieve information located at a certain index, you may need to know the total number of lines, already within the rendering surface by using:

```
int getItemCount (),
```

and eventually retrieving the text at any line position:

```
int getItem (int index)
```

Similar commands may be used in order to remove specific lines of text. The following command:

```
void delItem (int index)
```

is used in order to remove the text at the line position specified by `index`, and:

```
void delAll ()
```

represents the fastest alternative that you may use to clear the entire working area of text and images.

§ Background Pictures Features

The text and images inside the working area can be rendered on a background image, which will be lying on the desktop in different positions. A desktop image will be loaded from a URL and can be shown in three different modes, *centered*, *tiled* and *stretched*. When centered, the picture is lying in the middle of the desktop, in its original size, when tiled, the picture is repeatedly drawn in order to fill the entire desktop and when stretched, the picture is resized to fit the size of the desktop.

Only the administrator may choose the background pictures available to the end-user, since modifying them at run-time might endanger the reliability of the entire Java application. In return, WebCab ScrollArea v2.0 offers the developer extensive customization capabilities regarding the existing desktop images. A complete list of these methods and related fields follows:

- `int getNoBackgrounds ()`

This method returns the number of existing background images, as configured by the administrator.

- `int getBackgroundNumber ()`

The index of the currently visible background picture, as specified by the administrator and selected by the user. In case this method returns a negative number, most likely -1, the desktop is currently not displaying any pictures.

- `Image getBackgroundImage ()`

Returns an object of type `java.awt.Image` which represents the current background picture lying on the desktop.

- `Image getBackgroundImageAt (int index)`

Returns an object of the same type as the previous method, which represents the background picture that corresponds to the specified index.

- `int getDisplayFlag ()`

This method tells the developer which of the three displaying modes is currently used for the background picture, centered, tiled or stretched. The returned integer can have any of the following values: `TextScroll.IMAGE_CENTER`, `TextScroll.IMAGE_TILE` and `TextScroll.IMAGE_STRETCH`, with the corresponding meanings.

- `int getStretchMode ()`

Provided that the administrator or the user choose to stretch the background image on the desktop, WebCab ScrollArea v2.0 offers two ways of displaying it, either by stretching in a *fast* mode, when the resizing will be done fast with less stress on quality, or by stretching in a *smooth* mode, when the resizing will be accomplished with great care for image quality, determining a longer rendering time. The value of the returned integer may be either `Image.SCALE_FAST` or `Image.SCALE_SMOOTH` as defined in the Java AWT package.

- `void setDisplayFlag (int flag)`

This method actually changes the value of the displaying flag, allowing a different positioning of the background picture on the desktop, as previously described. You may choose any of the `TextScroll.IMAGE_CENTER`, `TextScroll.IMAGE_TILE` and `TextScroll.IMAGE_STRETCH`.

- `void setStretchMode (int mode)`

This is how a developer will be able to change the stretching mode of a background picture. Specify a fast or smooth resize by choosing any of the `Image.SCALE_FAST` or `Image.SCALE_SMOOTH` values, as defined in the Java AWT package.

- **void changeImage (int index)**

A developer will change the current background picture with this method. Specify an index from the valid range from 0 to `getNoBackgrounds () - 1` to select a picture or choose -1 in order to remove any picture from the desktop. If you choose that no background picture be displayed, the desktop will use the background color as specified by the administrator and adjusted by end-user.

- **void changeImage (int index, int displayFlag, int stretchMode)**

Similar to the previously described ones, this method selects a background picture for the desktop, specifying at the same time a different displaying flag and stretching mode. Although it is a convenience for the previous three methods, this method is the fastest way to change the background picture, the display and stretching modes when the Java bean is already showing on screen. The reason is that after each of the previous methods is invoked, WebCab ScrollArea v2.0 will automatically redraw the entire rendering surface. Save time with a single call to this method.

§ Images Features

Besides text, WebCab ScrollArea v2.0 offers the user image displaying capabilities. Like background pictures, the text-embedded images are selected by the administrator - in a configuration file - and can only be accessed by the end-user. The developer is not able to interfere directly with the collection of images, but can only access them in a user specific manner, by adding, changing and removing lines of text.

§ Colors Features

WebCab ScrollArea v2.0 allows color customization for the following areas:

- Background
- Normal Text
- Highlighted Text
- Internet Links

The *background* color is the color displayed on the desktop, when there is either no background image or the background picture is centered. In case that the background picture is tiled or stretched, there is no background color to display.

WebCab ScrollArea v2.0 supports attributes control sequences, which are contained directly within lines of text. When no color-related attribute is set, the text is displayed using the *normal text* color, whereas when highlight-related attributes are specified, the text is drawn using the *highlighted text* color. The highlighted text is supposed to be used for emphasizing certain phrases, words or characters, proving useful inside any visual content.

Apart from normal and highlighted text, there is another category of objects inside WebCab ScrollArea v2.0, the *Internet Links*. These are World Wide Web specific Universal Resource Locators (URL) which can also be internally customized to display any Red-Green-Blue 24bit color. A complete list of color-control methods follows:

- **Color getBackgroundColor ()**

This method retrieves the current desktop color, as either chosen by the administrator or the end-user.

- **Color getTextColor ()**

Returns the current normal text color. The normal text is the text that has not been affected by any attributes control sequences.

- **Color getHighlightColor ()**

Returns the current color of the highlighted text. The highlighted text color may also be chosen by the end-user and administrator.

- **Color getLinkColor ()**

This is the color of Internet links, as specified by the end-user.

- **void setBackgroundColor (Color color)**

The developer may choose to change the color of the desktop by invoking this method. The `color` argument represents the desired color.

- **void setTextColor (Color color)**

Similar to the previous method, this one is responsible with selecting the normal text color.

- **void setHighlightColor (Color color)**

This method changes the current highlighted text color to the color specified by the `color` argument.

- **void setLinkColor (Color color)**

This is how the color of Internet links can be changed by the developer.

In addition, offered as a development convenience, the following methods may prove faster and more reliable when several colors need to be retrieved or changed at a stroke:

- `Color[] getColorTable ()`

This method retrieves the colors for each object, in the following order: background color, normal text color, highlighted text color and Internet links color. It is equivalent to a slower quadruple call to `getBackgroundColor ()`, `getTextColor ()`, `getHighlightColor ()` and `getLinkColor ()`.

- `void setColorTable (Color[])`

When you need to change more than one of the four colors, this method is a fast alternative to the one-by-one color selection. Using this method is essential when the Java bean is showing on screen. The order of the colors in the `Color` array is: background color, normal text color, highlighted text color and Internet links color.

§ Fonts Features

WebCab ScrollArea v2.0 offers a detailed customization of font attributes, such as font faces, sizes and style, both for the user and the administrator. Low level personalization abilities are given to the developer, who can adjust settings such as default text font and sizes preferences.

The end-user is initially given a default font to start with, which will be used for the default font-free attribute control sequences text. The two methods that change the default text font are:

- `Font getTextFont ()`
- `void setTextFont (Font font)`

Yet, since the user may be able to choose different font sizes, a lower limit and an upper limit for the font size is required. Even though the Java bean itself has such predefined limits, the administrator or the developer are given the opportunity to change these settings according to their needs.

First, the JavaBeansTM component has two variable lower and upper limits for the font size, called `minFontSize` and `maxFontSize`. These variables can be changed by the administrator, at initialization time, and by the developer, at run-time. The user will not be able to go beyond any of these limits when configuring the look of the Java bean from the Configuration Dialog. The developer will be using the following methods for font size limits control:

- `int getMinFontSize ()`

This method is used to learn the current minimum admissible font size.

- `int getMaxFontSize ()`

Similarly, this method retrieves the current maximum allowed font size.

- `void setMinFontSize (int minFontSize)`

In order to define a new minimum font size, the developer will invoke this method with the desired `minFontSize` argument.

- `void setMaxFontSize (int maxFontSize)`

This is how the developer will be changing the maximum font size, by specifying a new `maxFontSize` parameter.

On second hand, these variable lower and upper limits cannot go beyond other two predefined limits. Their values can be known by accessing:

- `TextScroll.DEFAULT_MIN_FONT_SIZE`, which represents the absolute minimum size that any font may have, and
- `TextScroll.DEFAULT_MAX_FONT_SIZE`, the absolute maximum font size.

In conclusion, the font size that is chosen by any of the three parties must conform to the administrator and developer imposed limits, which themselves must comply with the built-in absolute font size limits, represented by the two above integer fields.

§ Miscellaneous Features

WebCab ScrollArea v2.0 offers additional development customizing options, that will enhance the displaying capabilities, behavior functionalities and constrain over user interference. Here is a list of all miscellaneous capabilities:

- **Word Wrapping**

This feature enables the end-user to visualize the entire text within the rendering area without an additional horizontal bar. Yet, the developer and end-user should note that the horizontal bar might be visible in case that the word is too long and cannot fit the rendering surface. The developer can toggle the “word wrapping” feature with the following methods:

- `boolean getWordWrap ()`

If word wrapping is enabled, this method will return a `TRUE` value or `FALSE`, if disabled.

- `void setWordWrap (boolean wordWrap)`

Invoke this method with a `TRUE` argument in order to enable word wrapping, or with a `FALSE` argument so as to disable it.

- **Vertical Alignment**

WebCab ScrollArea v2.0 allows the administrator and the developer to customize the vertical alignment of the text inside the working area. There are two types of such alignments: *text area* alignment and *list-like* alignment. The text area vertical alignment is the standard way of displaying lines of text, that start from bottom and go up, so that the last line is always on the bottom of the area. This way is a very common visualization type in Internet chat programs, when the latest information is the first to be seen.

The second vertical alignment type, the list-like alignment, is common to text editors and choice lists and glides from up to bottom, leaving the last line always on the screen. This feature will prove very useful in many Java-based applications, and it can accommodate most text area alignments.

In order to change or retrieve the vertical alignment type of the Java bean, you may use the following methods:

- `int getValignMode ()`
This is the method that informs the developer of the current vertical alignment mode. The returned integer may take any of the two available values, which are: `TextScroll.VALIGN_AREA`, representing the text area alignment, and `TextScroll.VALIGN_LIST`, representing the list-like alignment.
- `void setValignMode (int valignMode)`
This method changes the current vertical alignment mode, according to the value of the `mode` integer. In order to configure a text area alignment, assign `TextScroll.VALIGN_AREA`, otherwise specify `TextScroll.VALIGN_LIST`.

- **Items Selection**

This feature proves vital when dealing with list-like vertical alignment types of WebCab ScrollArea v2.0 Java beans instances. It allows the user to select every line of text independently, and to listen for any related action events. In order to allow the user to take advantage of this functionality, you will use these two methods:

- `int getSelectFlag ()`
This method allows the user to study the current behavior of the JavaBeans component, regarding item selection. If the bean is currently not activated for item selections, the method will return a `TextScroll.SELECT_NONE` value, whereas if the bean has been initiated for this purpose, it will result in a `TextScroll.SELECT_ALLOW` integer value.
- `void setSelectFlag (int value)`
In order to activate the item selection feature, the developer can make a call to this method with a `TextScroll.SELECT_ALLOW` value argument. To disable such selection, assign the `TextScroll.SELECT_NONE` to the `value` argument.

In addition, you can manipulate the selections and retrieve selected items with the use of the following methods:

- `int getSelectedIndex ()`
The number of the currently selected line of text is returned by a call to this method. A negative number corresponds to a situation when no line is selected.
- `String getSelectedItem ()`
This method returns the line of text that is currently selected by the end-user.
- `void setSelectedIndex (int index)`
This method determines a new item selection to be made, corresponding to the value of the `index` parameter.

• End-user Configurability

This functionality is developer specific and can be neither accessed by an administrator, nor can it be altered by the end-user. Normally, the end-user is allowed to change many settings accessible to the administrator and the developer, by simply right-clicking on the desktop and using the “Configuration Dialog”. The Configuration Dialog can adjust many visual settings, regarding font characteristics, background pictures, word wrapping and color settings.

Yet, for implementing purposes, the developer may choose to disable this embedded feature of WebCab ScrollArea v2.0, so that the right-click on the rendering surface desktop be ignored. For example, when trying to provide another functionality for the desktop mouse clicking or for security reasons, this feature proves imperative.

To adjust the configurability of the component, you may use the following methods. A `TRUE` value indicates that the user is given access to the Configuration Dialog, whereas a `FALSE` boolean value denies it:

- `boolean getConfigurable (),` and
- `void setConfigurable (boolean configurable)`

• Applet Interactivity

In order to fully support the Internet links capability, you must deploy WebCab ScrollArea v2.0 inside an applet. After doing so, all you need to do, to activate this feature, register the component with the following methods:

- `Applet getApplet ()`
- `void setApplet (Applet applet),`

where `applet` is the currently running Java-based applet, from which the bean instantiation has been made.

3.4.2 The *TextScroll* JavaBeansTM component

The *TextScroll* class is the programming core of the WebCab ScrollArea v2.0 component. Although *TextScroll* is also a JavaBeans component, it is used only by means of the previously described bean, *ScrollArea*. The *TextScroll* bean represents the rendering surface

where the text and the images are displayed for the end-user, whereas *ScrollArea* is responsible with the horizontal and vertical scrolling of the text and with the developer specific interface. *ScrollArea* acts like a portal between the developer and the actual JavaBeans component, since any process between developer and WebCab ScrollArea v2.0 takes place only inside the *TextScroll* bean instance.

All methods that are provided by the *ScrollArea* Java bean have a direct equivalent inside the *TextScroll* component, and all related documentation can be found inside the JavaDocs Generated Documentation. Even though instances of this class should not be directly accessed, it might prove necessary that the *TextScroll* methods be accessed from a *ScrollArea* instance. Such a need could arise, for example, when the developer needs to implement different events behavior directly to the JavaBeans component. The following method provides direct access to this class:

- `TextScroll getTextScroll ()`,

and is located inside the *ScrollArea* Java bean. As an example, consider that a developer needs to add a `MouseMotionListener` to the component. For this purpose, the corresponding code would be:

```
ScrollArea scrollArea = (ScrollArea) Beans.instantiate (...);
scrollArea.getTextScroll ().addMouseMotionListener (mouseMotionListener);
```

Yet, WebCab ScrollArea v2.0 provides accessible listener specific methods, directly through the *ScrollArea* Java bean, so that any such call be applied to the inner *TextScroll* instance. Here is a complete list of these methods:

- `void addActionListener (ActionListener l)`
- `void addFocusListener (FocusListener l)`
- `void addKeyListener (KeyListener l)`
- `void addMouseListener (MouseListener l)`
- `void removeActionListener (ActionListener l)`
- `void removeFocusListener (FocusListener l)`
- `void removeKeyListener (KeyListener l)`
- `void removeMouseListener (MouseListener l)`

For example, in order to add or remove a `KeyListener`, the following code should be used instead:

```
ScrollArea scrollArea = (ScrollArea) Beans.instantiate (...);
scrollArea.addKeyListener (keyListener);
...
scrollArea.removeKeyListener (keyListener);
```

3.4.3 The Color Picker Java bean

The “Color Picker” is the third Java bean of WebCab ScrollArea v2.0. It is represented by several non-public classes, such as *Gradient*, *ColorPlate* and *PlateCreationThread*, and a central public JavaBeansTM component class, named *ColorPicker*. This bean represents an important part of the communication bridge between administrator and the end-user, because it allows the user to personalize the background color and text color, with a simple drag of mouse. Since it is only used inside the “Configuration Dialog”, the color picker has little development importance, so that methods provided require minimum configuration effort and time.

In order to start using the WebCab Color Picker inside your Java-based application, you will need to create an instantiation of the bean, with the following code, similar to the one presented in the *ScrollArea* subsection:

```
import java.beans.*;
import com.webcab.chat.gui.*;

...

try {
    ColorPicker cPicker = (ColorPicker) Beans.instantiate (
        getClass ().getClassLoader (), "com.webcab.chat.gui.ColorPicker");
}
catch (IOException e) {
}
catch (ClassCastException e) {
}
```

After instantiating the *ColorPicker* bean, you may use the following two methods to access the picker's color:

- **Color getColor ()**
This method tells the developer what color is currently displaying inside the color picker.
- **void setColor (Color newColor)**
In order to initialize the color picker with a new color, invoke this method with a corresponding **newColor** argument.

3.4.4 ScrollArea Listeners

There are two types of listeners provided by WebCab ScrollArea v2.0, *ActionListener* and *ScrollAreaListener*. The *ActionListener* is invoked every time the end-user double-clicks with the left mouse button inside the rendering surface, for example when running in a list-like mode. The other listener, *ScrollAreaListener* provides a similar method, which acknowledges the developer if the currently selected item has changed.

The two methods corresponding to each event are:

- `void actionPerformed (ActionEvent e)`,
which is located inside the `java.awt.event` package, and
- `void selectedItemChanged (ScrollAreaEvent e)`,
from the `ScrollAreaListener` interface.

Use these methods to supervise any mouse events inside the working area.

This is the end of the development related documentation. Further reading will take you through the administrator specific documentation, followed by the end-user guide. You may want to skip the following two sections and start browsing the “Examples” chapter.

3.5 Administrator's Guide

As previously explained, the administrator is the party responsible with the non-technical end-user WebCab ScrollArea v2.0 configuration. Even though this basically means that as an administrator, you will never need to know any Java programming details, this section covers certain areas, where direct support from the developer will be required.

The administrator will be working all the time with only one file, called the “Configuration File”. This file will be specified as a Universal Resource Locator (URL) to the developer, who will include it inside the source code. For example, a valid configuration file URL is `http://www.webcabcomponents.com/webcabscrollarea/my.settings`, where “my.settings” is the actual configuration file. The file contains an option for every feature, each option being written on a separate line.

To activate the configuration file, the developer will invoke the `setConfiguration (String url)`, by specifying the Configuration File's URL as a parameter. In order to learn about each administrator configurable feature, feel free to read the following subsections.

3.5.1 Background Pictures

WebCab ScrollArea v2.0 allows the user to display GIF and JPEG background pictures on the desktop of the rendering surface, allowing the text and images to glide over. The background pictures may be aligned on three different postures: *centered*, when the picture is lying in the center of the working area, in its original size, *tiled*, when the picture will be repeatedly displayed in order to fill the entire surface and *stretched*, when the background

picture fits the entire area.

This feature can be configured, by specifying the following lines inside the configuration file:

Configuration Option	How it Works
<code>NEW_BACKGROUND = "url"</code>	Registers the picture at the "url" address into the user's collection of background pictures. You may add an indefinite number of such pictures, without limitations of any kind.
<code>DISPLAY_FLAG = flag</code>	This option is used to change the position of the background pictures on the desktop, viz. centered, tiled or stretched. In order to select a centered type of positioning, use "CENTER" as <code>flag</code> , for tiled, use "TILE" and "STRETCH" for stretched.
<code>STRETCH_MODE = mode</code>	When stretched, the background picture can display in two different modes, <i>fast</i> and <i>smooth</i> . When stretched fast, the resizing process concentrates on speed and less on quality, as opposed to when the the background picture is stretched smoothly, and the quality of the resized image is very good.
<code>STARTUP_IMAGE = number</code>	When initializing, WebCab ScrollArea v2.0 will display a start-up background picture, according to the value of the <code>number</code> specified in this option. The number of the start-up image ranges from 0 to the total number of background pictures, minus 1. The order is the same with that in the configuration file, so that if, for example, you have added 4 background pictures, you would assign a value of 2 to the <code>number</code> parameter, in order to select the 3rd one.

3.5.2 Colors

You can set colors attributes for four objects inside WebCab ScrollArea v2.0, *background*, *normal text*, *highlighted text* and *Internet links*. Moreover, WebCab ScrollArea v2.0 JavaBeansTM component offers additional color managing features, available only to the administrator and user parties. This means that, when adding a line of text to the rendering surface, you may choose to use certain "Attributes Control Sequences", that allow dynamic Look & Feel text personalization. These sequences are configured by the administrator and used by the user, without discrimination.

By inserting "Color Control Sequences" inside a line of text, you will change the color of the succeeding text, until another similar sequence is used. All color related options are described below:

Color Option	How it Works
<code>backgroundColor = color</code>	Defines a new color for the desktop. This color is visible when there is either no background image activated, or the image does not entirely cover rendering surface. The <code>color</code> parameter can be defined by either specifying a color name, such as "white", or by providing the 24bit Red-Green-Blue comma separated indices. Here is the list of all predefined colors available in the configuration file: • White • Black • Blue • Red • Cyan • Pink • Yellow • Orange • Gray • Lightgray • Darkgray • Green • Magenta To specify a RGB color, type a "Red value, Green value, Blue value" sequence in place of the <code>color</code> parameter, such as "255, 127, 255", or "120, 0, 221".
<code>textColor = color</code>	Specifies the color of the normal text, the text which is not affected by any color control sequences.
<code>highlightColor = color</code>	Changes the color of how the highlighted text is displayed. The highlighted text can be displayed by color control sequences.
<code>linkColor = color</code>	This option selects a color for how the Internet links are displayed inside the working area. All links are automatically detected by WebCab ScrollArea v2.0.

Color Option	How it Works
DEFINE_COLOR "Definition" color	This is the option that defines the "Color Control Sequences". When typing this definition in any line of text, the text will change to the specified color. The color argument may be configured by name or by RGB, similar to the previously described options. In addition, two other color definitions are available: "default", used to specify the "normal text" color, and "highlight", for the "highlighted text" color, both as defined by the end-user. Please refer to the examples section for a "hands-on" explanation.

3.5.3 Fonts

The font of the text displayed inside the working area of a WebCab ScrollArea v2.0 Java bean instance may be personalized by name, style and size.

Configuration Option	How it Works
MIN_FONTSIZE = size	Defines the lower limit of the size value that a font may take. This setting limits the Configuration Dialog font size settings as well, imposing the same restriction for the end-user.
MAX_FONTSIZE = size	Similar to "MIN_FONTSIZE", this option defines the upper limit of the size value of the text font. It also limits the user's options, so that no size may go beyond.
FONT = "Name", style, size	Choose the default text font with this option. The name is a valid Java font face; you may find all the supported fonts inside the Configuration Dialog. The valid styles that can be used in the configuration file are: • PLAIN - simple style • BOLD - bold font • ITALIC - italic font, slanted to the right • BOLDITALIC - both bold and italic at the same time The size is a positive number, that should not exceed the limits defined in the "MIN_FONTSIZE" and "MAX_FONTSIZE" configuration options.

Fonts can also be used inside attributes control sequences, allowing extensive personalization of the text that the user wishes to see. Every font characteristic can be customized

with a corresponding control sequence, as described in the following sections.

§ Font Names

A font name control sequence is defined with the "DEFINE_FONT" configuration option, followed by a "Definition" and a font-name. For example, in order to assign an "[s]" definition to the "Serif" font name, you would type the following line in the configuration file:

Font Name Option	How it Works
DEFINE_FONT "[s]" "Serif" <i>or</i> DEFINE_FONT "[s]" Serif	Defines the "[s]" font name control sequence. By typing this sequence in a line of text, the succeeding font name will change to "Serif", until a similar font control sequence is used.

In order to select the normal text font, as defined by the administrator with the "FONT" option and configured by the user at run-time, use "default" as font name.

Font Name Option	How it Works
DEFINE_FONT "[n]" default	Defines a normal font name control sequence, so that when the user types "[n]" inside any line of text, the font face defaults to the predefined font name, as specified by the administrator or the end-user.

§ Font Styles

Besides the standard run-time available font styles, "plain", "bold" and "italic", a control sequence also supports the "underline" font style. A font style control sequence starts with "DEFINE_STYLE", followed by the "Definition" and the style name. A list of complete style options follows.

Font Style Option	How it Works
PLAIN	Defines the plain font style option, by discarding the bold, italic and underline style settings.
BOLD	This is the option for a bold font style.
ITALIC	Specifies an italic font style, slanted to the right.
UNDERLINE	Use this style name to define the underline font style.
NO_BOLD	In order to deselect a previous bold style selection, use the "NO_UNDERLINE" style name.
NO_ITALIC	Similar to the previous option, the "NO_ITALIC" style name will discard a preceding italic font setting.
NO_UNDERLINE	Removes the underline font style setting at the current line position.
restore	This option discards the last style name defined in the same line of text, whether bold, italic or underline.

Font Style Option	How it Works
default	Restores the default font style settings, as shown in the configuration dialog.

§ Font Sizes

The size of a font is selected relatively to the current size of the text font. It may be increased or decreased accordingly with the use of font size control sequences. There are two types of such sequences, *fixed* and *virtual*.

A fixed font size control sequence alters the size of the text font by a predefined amount of points, whereas a virtual font size control sequence allows the user to specify a variable amount of points by which to modify the size of the text. The following table explains each of them:

Font Size Option	How it Works
DEFINE_SIZE "Definition" "+number"	This is a standard positive fixed font size control sequence. The "Definition" is the text which typed in a line of text will apply an increase of font size by the number of points specified by the number parameter.
DEFINE_SIZE "Definition" "-number"	Like the previous option, this "Definition" will decrease the font size by the number of points.
DEFINE_SIZE "...%..." "+%"	This is a virtual font size control sequence, which defines a variable amount of points by which to increase the font size. When the user types a number in place of the "%" sign, the size of the text font, which follows the sequence, will increase by that number of points. For example, if the option is "DEFINE_SIZE "[plus%]" "+%", typing "[plus20]" in a line of text will increase the font by 20 points.
DEFINE_SIZE "...%..." "-%"	This option bears a similar functionality to the previous one, by decreasing the font size with the number of points, specified in place of the "%" sign.

3.5.4 Images

Like font and color attributes control sequences, the user can also display images inside lines of text. These images are defined by the administrator, in a similar manner to the background pictures definition, and declared with image specific sequences. The following options allow you to customize images behavior.

Configuration Option	How it Works
NEW_PICTURE "url"	Defines a picture located at the specified "url". The number of pictures is not limited, and they are indexed in the same order as shown in the configuration file.
DEFINE_IMAGE number "Definition"	When the "Definition" is typed in a line of text, the image corresponding to the <code>number</code> parameter will be displayed in the rendering surface. The numbering starts from 0 to the total number of images, minus 1, so that, for example, number 3 points to the fourth image, as ordered in the configuration file.
DEFINE_IMAGE "...%..." "%"	This option allows the users to input the exact number of the image they want to include in the line of text. By typing an image number in place of the "%" sign, the bean will display the corresponding image, as specified in the configuration file. For example, if you type "DEFINE_IMAGE "image:%" "%" in the configuration file and the user writes "image:5" inside any line of text, the sixth image will be drawn on the screen.

3.5.5 Miscellaneous

WebCab ScrollArea v2.0 offers additional features that enhance graphical display and functionality, such as *word wrapping*, *vertical alignment* and *item selection*. The configuration options follow:

Miscellaneous Option	How it Works
WORD_WRAP = TRUE/FALSE	Select "TRUE" to enable word wrapping inside the rendering surface and "FALSE" in order to disable it. Word wrapping allows the user to visualize all lines of text without requiring to scroll the horizontal bar.
VALIGN = Area/List	This option selects the type of vertical alignment of the JavaBeans TM component. The "Area" type of alignment selects a chat-like displaying mode, whereas "List" asks for a list-like behavior.
SELECT_FLAG = SELECT_NONE or SELECT_FLAG=SELECT_ALLOW	If you choose the "SELECT_ALLOW" option, the user will be able to select every line in the text independently and double-click on each item. To disable the item selection, choose "SELECT_NONE".

Remember that inside a definition you may use spaces and other keyboard special characters. In case you wish to use the equals sign in one of your definitions, you will require to type a slash sign in front of it, in order to avoid any misunderstandings. For example, the following option, `DEFINE_IMAGE <img src\= %> %` would register `<img src=number>` as an image control sequence.

This ends the documentation on WebCab ScrollArea v2.0 administration. The following section describes the end-user features, so that you may want to skip to the next chapter and start browsing for examples.

3.6 User's Guide

This section covers brief information regarding the use of the graphical interface and personalization capabilities of WebCab ScrollArea v2.0. Since all configuration options are easy to use and intuitive, the end-user related documentation is limited to a minimum, saving time and space with unnecessary explanations.

The end-user has many customization possibilities, allowing him to change settings regarding fonts, colors and background pictures. All the graphical settings can be accessed inside the “Configuration Dialog” and this will be the tool that the user will use for configuration issues.

The configuration dialog is activated by pressing the right (or middle) mouse button on the desktop of the working area. In the upper left side of the configuration window, the color settings are located, where you can change the background color, the normal text color, the highlighted text color and the Internet links color. Below, in the lower left side, the user may modify the current font settings, such as name, size and styles, choosing between plain, bold and italic.

Underneath the font settings, you may toggle the word wrapping feature, that allows you to visualize the entire text, without needing to scroll the horizontal bar. In the right side of the configuration window, you can select the background picture settings. Slide through the scrolling bar in order to change the desktop image and choose a displaying mode, *tiled*, *centered* or *stretched*. For stretched background pictures, you may select a *fast* or *smooth* rendering.

After finishing configuring the settings, press the OK button for the changes to take place. The **Apply** button allows you to update the settings on-the-fly without closing the configuration window. If you decide to preserve the current settings of WebCab ScrollArea v2.0, press the **Cancel** button. Any other information regarding the use of the component should be provided by the administrator of the Java-based product the user will be running.

This is where the Programmer's Guide chapter ends. Explore the next chapter for a list of explanatory examples on how to configure WebCab ScrollArea v2.0.

Chapter 4

Examples

This chapter provides practical examples on how a developer and an administrator can best personalize the features offered by WebCab ScrollArea v2.0. All examples comply with the “[Developer’s Guide](#)”, “[Administrator’s Guide](#)” and the JavaDocs Generated Documentation WebCab ScrollArea v2.0 specifications. Examples of configuration files and Java source codes, with included explanations, follow.

Hardware Presentation Example

This is an administrator related configuration file. You may add personal comments and remarks about the options used, without marking them with any special characters. This is a possible example about how WebCab ScrollArea v2.0 may be used for a hardware products presentation.

#image 0# This image should be the first to be displayed, i.e. `STARTUP_IMAGE = 0`, as it welcomes the user to the hardware presentation.

`NEW_BACKGROUND = "http://www.webcabcomponents.com/intro/welcome.jpg"`

#image 1# The Enterprise JavaBeansTM Dual Processor Application Server.

`NEW_BACKGROUND = "http://www.webcabcomponents.com/intro/dualsystem.gif"`

#image 2# Monthly sales for Pentium4 processors.

`NEW_BACKGROUND = "http://hardware.webcab.co.uk/intel/pentium4/library/sales.jpg"`

#image 3# Thunderbird Athlon processor sales.

`NEW_BACKGROUND = "http://hardware.webcab.co.uk/amd/thunderbird/sales.gif"`

#image 4# The exit screen.

`NEW_BACKGROUND = "http://www.webcabcomponents.com/intro/bye.gif"`

`STARTUP_IMAGE = 0`

`DISPLAY_FLAG = STRETCH`

Hardware Presentation Example (*continued*)

```
> The background pictures should be clear and stretched.  Avoid tiling!
STRETCH.MODE = SMOOTH
```

```
>> All text should be visible on the screen.  We don't want people to drag
the horizontal bar all the time.  One vertical bar is enough.
WORD_WRAP = TRUE
```

```
SELECT_FLAG = SELECT_NONE
```

```
FONT = "Impact", PLAIN, 20
```

```
MIN_FONTSIZE = 15
MAX_FONTSIZE = 100
```

```
backgroundColor = white
textColor = black
highlightColor = blue
linkColor = 127, 127, 255
```

```
The end of the configuration file for WebCab ScrollArea v2.0
```

Example 4.1**List-like Behavior Example**

```
Configuration file for a list-like WebCab ScrollArea v2.0 bean
NEW_BACKGROUND = "http://www.webcab.co.uk/chat/library/back1.jpg"
```

```
# No background image for startup
STARTUP_IMAGE = -1
```

```
WORD_WRAP = FALSE
```

```
DISPLAY_FLAG = CENTER
```

```
# Allow item selections
SELECT_FLAG = SELECT_ALLOW
```

```
# List-like alignment
SELECT_FLAG = SELECT_NONE
```

```
FONT = "Times New Roman", PLAIN, 10
```

```
backgroundColor = lightgray
textColor = 0, 0, 0
highlightColor = blue
linkColor = "red" # END
```

Example 4.2

Attribute Control Sequences Example

This configuration file exploits almost every attribute control sequence definition.

```
NEW_BACKGROUND = "http://webcab.co.uk/chat/library/back1.jpg"
NEW_BACKGROUND = "http://webcab.co.uk/chat/library/back2.jpg"
NEW_BACKGROUND = "http://webcab.co.uk/chat/library/open.jpg"
NEW_BACKGROUND = "http://webcab.co.uk/beans/gifs/cigar_med_wte.gif"
NEW_BACKGROUND = "http://webcab.co.uk/chat/library/back5.jpg"
NEW_BACKGROUND = "http://webcab.co.uk/chat/library/back6.jpg"
```

```
STARTUP_IMAGE = 5
```

```
WORD_WRAP = TRUE
```

```
DISPLAY_FLAG = TILE
STRETCH_MODE = FAST
```

```
SELECT_FLAG = SELECT_ALLOW
```

```
FONT = "SansSerif", PLAIN, 14
```

```
MIN_FONT_SIZE = 8
MAX_FONT_SIZE = 48
```

```
backgroundColor = 255, 255, 255
textColor = 0, 0, 0
highlightColor = 0, 0, 128
linkColor = 128, 128, 255
```

```
NEW_PICTURE = "http://webcab.co.uk/cigar_med_wte.gif" [image 0]
NEW_PICTURE = "http://webcab.co.uk/cmtcigars_logo.jpg" [image 1]
NEW_PICTURE = "http://webcab.co.uk/animcigar.gif" [image 2]
NEW_PICTURE = "http://webcab.co.uk/happy1.gif" [image 3]
NEW_PICTURE = "http://webcab.co.uk/happy2.gif" [image 4]
NEW_PICTURE = "http://webcab.co.uk/happy3.gif" [image 5]
```

```
-- Font definitions --
```

```
DEFINE_FONT "[s]" "Serif"
DEFINE_FONT "[ss]" "SansSerif"
DEFINE_FONT "[di]" "Dialog"
DEFINE_FONT "[mo]" "Monospaced"
DEFINE_FONT "[n]" "default"
```

Attribute Control Sequences Example (*continued*)

```

-= Color definitions -=
DEFINE_COLOR "[b]" blue
DEFINE_COLOR "[r]" red
DEFINE_COLOR "[y]" yellow
DEFINE_COLOR "[g]" green
DEFINE_COLOR "[m]" magenta
DEFINE_COLOR "[k]" black
DEFINE_COLOR "[w]" white
DEFINE_COLOR "[o]" orange
DEFINE_COLOR "[c]" cyan
DEFINE_COLOR "[p]" pink
DEFINE_COLOR "[a]" gray
DEFINE_COLOR "[l]" lightGray
DEFINE_COLOR "[d]" darkGray
DEFINE_COLOR "[w]" 125, 0, 123
DEFINE_COLOR "[n]" default
DEFINE_COLOR "[z]" highlight

-= Font Style definitions -=
DEFINE_STYLE "[x]" PLAIN
DEFINE_STYLE "[h]" BOLD
DEFINE_STYLE "[i]" ITALIC
DEFINE_STYLE "[u]" UNDERLINE
DEFINE_STYLE "[h]" NO_BOLD
DEFINE_STYLE "[i]" NO_ITALIC
DEFINE_STYLE "[u]" NO_UNDERLINE
DEFINE_STYLE "[]" restore
DEFINE_STYLE "[n]" default

-= Font Size definitions -=
DEFINE_SIZE "[-%]" "-%"
DEFINE_SIZE "[+%]" "+%"
DEFINE_SIZE "[h1]" "+10"
DEFINE_SIZE "[n]" default

-= Images definitions -=
DEFINE_IMAGE "[i%]" "%"
DEFINE_IMAGE "<img src\=%>" "%"
DEFINE_IMAGE "(:)" "1"
DEFINE_IMAGE "(:)" "1"
DEFINE_IMAGE "*cigar*" "3"
DEFINE_IMAGE "*leo*" "4"

```

Example 4.3

This is the end of all examples offered by WebCab ScrollArea v2.0. For questions, suggestions and comments, please contact us at support@WebCabComponents.com.

Chapter 5

ScrollArea UML Diagram

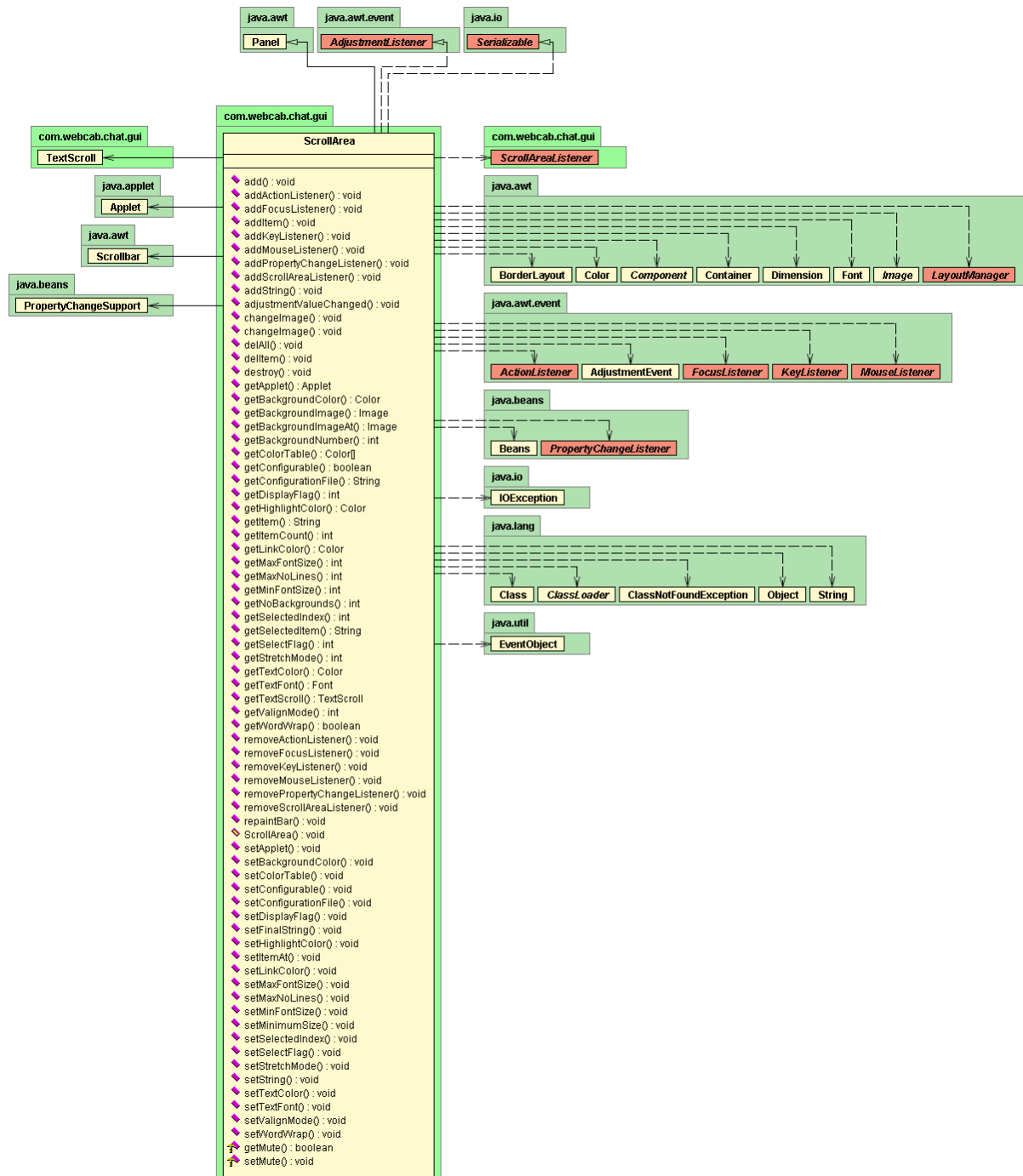
Within this chapter we provide UML diagram(s) for the interfaces of the ScrollArea component. The diagrams use standard UML notation detailing the inheritance, dependency, association, interface, methods, fields and properties of this component and associated packages. We present the UML diagrams with a large amount of white space so that the developer has space in which to add additional remarks.

UML Diagram Key

- **Extended Classes (interfaces)** - A solid line with a large triangle that points from the subclass (resp subinterface) to the superclass (resp inherited interface)
- **Classes** - Displayed in a rectangular box with a default yellow background with the name at the top and field, methods, and properties listed below it
- **Abstract Class and Methods** - Displayed in italic font
- **Implementing Classes** - A dashed line with a large triangle which points from the implementing class to the inherited interface
- **Interfaces** - A rectangle with orange background and the interface name in italic font
- **Implemented Interfaces** - A dashed line with a large triangle which points from the implementing class to the implemented interface
- **Dependencies/Reverse Dependencies** - A dashed line with arrowhead
- **Associations/Reverse Associations** - A solid line with an arrowhead
- **Packages** - A green rectangle with a tab and the package name in the tab or below it
- **Methods** - Listed below members and fields, including the return type
- **Members/fields** - Listed below the class name, including the return type
- **Properties** - Properties are displayed separately in the bottom of the class diagram
- **Static** - Static members, fields, variables, and methods are underlined in the diagram

5.1 Diagram

Zoom to at least 360% in order to be able to read the names of the fields, methods, component names and packages.



Chapter 6

Guide to WebCab Components

6.1 The Company

WebCab is a privately owned British company that has built business solutions since its inception in 1999. We continue to refine and develop our Mathematical and Financial Framework which we have implemented on the J2SE, J2EE and .NET platforms.

6.2 Presentation of Products

We aim to provide good quality, useful information to help you decide which J2EE application best suits your development needs. WebCab is committed to honesty and realism when presenting our products. In order to achieve this a detailed, clear and factual style for presentation is adopted within our documentation and marketing material.

6.3 Supported Clients, IDEs, Containers and DBMSs

We support all major development, server and client side technologies. Each product contains detailed examples and advice concerning the integration of our enterprise applications within existing infrastructure. Our documentation provides the information that the developer needs to get their applications up and running as quickly and as easily as possible.

We detail exactly how the developer can use our J2EE Applications with the following technologies:

- Client containers - Internet Explorer, Netscape Navigator, Opera
- IDEs - JBuilder, Eclipse, BEA Studio, Sun ONE (formerly Forte For Java), IBM VisualAge, Oracle JDeveloper
- Client Side Technologies - Application Clients, Servlet, JSP, Applets
- EJB containers - IBM WebSphere, BEA WebLogic, Oracle 9iAS, Sun ONE AppServer, Borland AppServer, Sybase EAServer, Ironflare Orion, JBoss
- DBMS - Oracle, IBM DB2, SQL Server, Sybase, MySQL
- Platforms - Windows XP/2000/NT, Linux, Solaris, IBM AIX

For these technologies we include all the deployment descriptors, installation scripts and template examples which will help the developer quickly and easily assemble their multi-tier enterprise application.

6.4 Transparent Functionality

All technical and business intelligence incorporated within our products is described within the associated documentation. This allows the developer to see the nature of the methodology implemented within our proprietary algorithms.

6.5 Code Conventions

WebCab adheres to the ‘Code Conventions for the Java Programming Language’ as specified on April 20, 1999 by Sun Microsystems Inc. These conventions cover filenames, file organization, indentation, comments, declarations, statements, white space, naming conventions and programming practices. Code conventions improve the readability of the software, allowing engineers to understand new code more quickly and thoroughly.

6.6 Company Culture and Activity

WebCab Components is focused solely on the production of high quality software modules within our Financial and Mathematical Framework. The WebCab team contains a wide range of expertise and experience from the academic, investment banking and software development worlds.

Within the company there exists significant internal competitive pressures with constant peer review and evaluation, resulting in higher quality products, which adhere to high professional standards and offer extended functionality.

6.7 Product Life cycle

We continuously add value to our products by evolving them according to new J2EE specifications, customer feedback and market demands. We give particular emphasis to incorporating our clients suggested modifications and additions into our product development cycle.

6.8 Support, Warranty and Upgrades

WebCab warrants that each component will perform substantially in accordance with the accompanying written material. We provide with all our products without charge sixty (60) days product support services including fixing bugs, compatibility issues and other technical support issues.

All maintenance updates (including service packs) will be distributed free of additional license cost. New version releases of this product will be offered to present licensee holders

at a greatly reduced rate.

Dr Ben Fairfax

Founder and CEO

WebCab Components - From Developer, To Developer

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. .NET is a trademark of Microsoft Corporation in the U.S. and other countries