Java Development Library

wls-dl-java-20060727-03

Revised 8/31/06

Wavelink Corporation
6985 South Union Park Avenue, Suite 335
Midvale, Utah 84047
Telephone: (801) 316-9000
Fax: (801) 316-9099
Email: customerservice@wavelink.com
Website: http:\\www.wavelink.com

Email: sales@wavelink.com

Table of Contents

Introduction

The Studio 5.0 Java Development Library is a complete list of classes, methods, and properties for wireless application development in the Java programming language. This document provides usage information, code syntax, and example code.

This introduction presents the following information:

- About this Document
- About the Studio 5.0 Java Development Library

About this Document

This section describes the assumptions and conventions of this document.

Document Assumptions

This document makes the following assumptions:

- Studio is installed on your computer
- A Java Development Kit is installed on your computer
- You are proficient with the Java Development Kit
- You are familiar with programming conventions and practices

Document Conventions

This section describes how different information is formatted in this document.

Code Syntax

Code syntax is represented in Courier New font. Within the code blocks, methods and properties are formattted in bold type. Placeholders are shown in italic type.

Example:

```
public void StoreBarcode(String FileName, int DefaultFlag)
```

Constants

Constants (including error codes) are represented as indented, uppercase text. Their descriptions, when necessary, are listed at the right.

Example:

WLFUNCTIONFAILED	- Wavelink function call failed at device
WLINVALIDRETURN	- Invalid WaveLink function return packet

Sample Code

Sample code is displayed in Courier New font.

Example:

```
barcodeIface = new WaveLinkBarcode();

// add two barcodes with type, expand, decode, min, max settings
barcodeIface.AddBarcode(WaveLinkBarcode.CODE_39, false, true, _
                        1, 32);
barcodeIface.AddBarcode(WaveLinkBarcode.UPC_A, false, false, _
                        1, 32);
barcodeIface.StoreBarcode("BCDemo", _
                          WaveLinkBarcode.BCDISABLED);
```

About the Studio 5.0 Java Development Library

The Studio 5.0 Java Development Library is a set of classes that you reference from your Java application. The library provides sixteen classes and hundreds of methods designed specifically for wireless-application development. These methods and objects range from basic input and output functions over a wireless network to playing custom tone files on a mobile device.

NOTE Java version 1.4 or compatible is required to use the Java Development Library.

This section includes the following information:

- Referencing the Java Development Library
- Programming Considerations
- Additional Information

Referencing the Java Development Library

Before developing custom wireless applications, you must include the Java Development Library in your specific development environment. This provides access to the custom classes and methods necessary for wireless application development.

Java Development Library: wavelink.jar

To access the Java Development Library in Java, you simply reference it in your javac command line. The Wavelink Development Library is contained in `wavelink.jar`.

```
javac -classpath ..\..\..\lib\wavelink.jar
TestSignon.java
```

Java Development Library: The WaveLinkApplication base class

All Wavelink applications extend the WaveLinkApplication base class. The Java PortMonitor dynamically loads the Wavelink application and calls the method WaveLinkMain. WaveLinkMain is the entry point for the custom Wavelink application. This is not a static method. It is not necessary to create a new object of your application class.

Programming Considerations

For the most part, developing Wavelink wireless applications is no different than developing any other conventional application. However, certain programming considerations must be kept in mind to gain maximum performance from Wavelink applications.

Display Size for Mobile Devices

The first programming consideration that must be taken into account is the limited and varied display sizes of mobile devices. Any information, either input or output, placed outside of these display limitations does not appear. Therefore, you must always keep in mind the display limitations of the target mobile devices when designing screens.

You may use the Class WaveLinkTerminal to obtain display characteristics of the mobile device. If the mobile devices on your network use different screen sizes, you may want to use the Class WaveLinkTerminal to dynamically position screen output. For example, you can program screen output one row from the bottom of the screen rather than setting a static row number.

Error Checking

The second consideration of Wavelink application development concerns method error checking. You must check for these exceptions within your application or your application may fail.

NOTE It is especially important to check for returned error values after making an input call such as RFInput or GetEvent. RF applications are dependent on returned error values to function properly.

For example, on a disconnect command from the Studio Administrator, a specific error message is thrown within your application. This error is thrown by each Wavelink function that communicates with the mobile device after the mobile device is physically disconnected from the network. The first time this error is received, your application should wait for a reconnect or close.

File Extensions

The third programming consideration concerns the use of file extensions used with various Wavelink objects. Certain Wavelink objects save files to mobile devices using the following extensions. The extensions and their associated objects are as follows:

WaveLinkIO	.scr
WaveLinkMenu	.mnu
WaveLinkTone	.ton
WaveLinkBarcode	.bar
WaveLinkScribblePad	.bmp and .jpg

When using these specified objects, it is not necessary to include these extensions when referencing files of the object types. If you reference these files using the WaveLinkFile object, though, it is necessary to include the appropriate file extension (or ".*" for all file extensions).

Additional Information

For more information about the Wavelink Studio product line, see the documentation for these Wavelink products:

- *Wavelink Studio 5.0 User's Guide*
- Wavelink Studio Client documentation

Debugging Java Applications

You can debug a Java-based Wavelink application from any Java IDE by running your application's `main` class. Studio provides this functionality through the Development Server.

The Development Server is a version of the server that initially accepts a single connection, runs the application, and then waits for further connections. By connecting to the Development Server from a mobile device, you can run the application entirely within a standard Java debugger.

The following lines of code must be included in the `main` class of the application:

NOTE In this example, `MyWLApplication` must be the name of the class containing the code.

```
public static void main (String[] args) {
    WaveLinkApplication wlApp = new MyWLApplication();
    wlApp.startServer(args);
}
```

To debug the application:

1. Verify that the application contains the required code.
2. Run the application inside a standard Java debugger.
3. Launch the new application from a mobile device or the Wavelink Windows Client.

   See the *Wavelink Studio 5.0 User's Guide* for more information about setting up the application in Studio.

When you run the application, the Development Server will automatically find an unused port and display it as part of a message in the Studio Administrator. For example:

```
The development server started normally on port XXXX.
```

Additionally, the following command line arguments are recognized by the Development Server:

`-port`	This command line option allows the user to specify a specific port to run on rather than a random unused one.

`-loglevel debug`
`-loglevel info`
`-loglevel warn`
`-loglevel error`
`-loglevel fatal`
	These options define the logging level for the development server (with `debug` logging the most information and `fatal` logging the least).

`-clargs`	This command line option allows the user to specify command line arguments to their Wavelink application. **NOTE** This option must be the last option used.

To shut down the server:

- On the host of the Development Server, type `x` or `X` and press the `ENTER` key.

Overview of Classes

This is an overview of the classes in the Studio Java Development Library.

- **Class ObjectHandle**

  class com.wavelink.clientui.ObjectHandle
  Forms the base of nearly all screen widget methods in Studio.

  - **Class WaveLinkScreen**

    class com.wavelink.clientui.WaveLinkScreen
    Provides the methods for creating widgets.

  - **Class WidgetHandle**

    class com.wavelink.clientui.WidgetHandle
    Includes methods common to all widgets.

    - **Class BitmapHandle**

      class com.wavelink.clientui.BitmapHandle
      Adds a single method to change the bitmap being shown.

    - **Class ButtonHandle**

      class com.wavelink.clientui.ButtonHandle
      Provides a method to change the text shown on a button.

    - **Class HTMLHandle**

      class com.wavelink.clientui.HTMLHandle
      Provides a method to change the text shown on a button.

    - **Class LabelHandle**

      class com.wavelink.clientui.LabelHandle
      Used to change the label text for a label widget.

    - **Class ProgressBarHandle**

      class com.wavelink.clientui.ProgressBarHandle
      Provides a method to change the amount shown on a ProgressBar widget.

- **Class CheckboxHandle**

  class com.wavelink.clientui.CheckboxHandle
  Provides methods for changing the text shown for a checkbox and for obtaining its current value.

- **Class FieldHandle**

  class com.wavelink.clientui.FieldHandle
  Provides the capability to obtain or change the text for a field widget.

- **Class MenuHandle**

  class com.wavelink.clientui.MenuHandle
  Provides the capability to change which WaveLinkMenu file is used and which option in the menu is selected.

  - **Class ComboboxHandle**

    class com.wavelink.clientui.ComboboxHandle
    Provides the capability to get the currently selected item from a combobox widget.

  - **Class ListBoxHandle**

    class com.wavelink.clientui.ListboxHandle
    Provides the capability to get the currently selected item from a listbox widget.

- **Class PopupHandle**

  class com.wavelink.clientui.PopupHandle
  Provides methods to obtain the last option (in a submenu) and title (main menu) references.

- **Class RadioButtonHandle**

  class com.wavelink.clientui.RadioButtonHandle
  Determines which option is selected in a radio button widget.

- **Class Region**

  class com.wavelink.clientui.Region
  Represents a widget region.

- **Class WaveLinkAuxPort**

  class com.wavelink.clientui.WaveLinkAuxPort
  Provides bi-directional access to the serial port of a device.

- **Class WaveLinkBarcode**

  class com.wavelink.clientui.WaveLinkBarcode
  Allows definition of a list of barcode configurations.

- **Class WaveLinkConstants**

  class com.wavelink.clientui.WaveLinkConstants
  Provides special values to control the display of widgets on devices and the input received from field widgets.

- **Class WaveLinkConstants.CheckboxState**

  class com.wavelink.clientui.WaveLinkConstants.CheckboxState
  Provides constants for configuring a checkbox widget.

- **Class WaveLinkConstants.Coord**

  class com.wavelink.clientui.WaveLinkConstants.Coord
  Used to configure a WaveLinkScreen coordinate system.

- **Class WaveLinkConstants.Justify**

  class com.wavelink.clientui.WaveLinkConstants.Justify
  Provides constants used for configuring text alignment on widgets.

- **Class WaveLinkConstants.Repeater**

  class com.wavelink.clientui.WaveLinkConstants.Repeater
  Provides constants for use with a repeater button.

- **Class WaveLinkConstants.State**

  class com.wavelink.clientui.WaveLinkConstants.State
  Used for setting the state of widgets.

- **Class WaveLinkError**

  class com.wavelink.clientui.WaveLinkError
  Notifies the user that an error has occurred and provides information about the specific error.

- **Class WaveLinkEvent**

  class com.wavelink.clientui.WaveLinkEvent
  Provides methods to determine how to handle a WaveLinkListener event.

- **Class WaveLinkFactory**

  class com.wavelink.clientui.WaveLinkFactory
  Contains the methods necessary to build widget objects. DEPRECATED.

- **Class WaveLinkFile**

  class com.wavelink.clientui.WaveLinkFile
  Provides the Wavelink file access interface for both public users and the package classes.

- **Class WaveLinkFont**

  class com.wavelink.clientui.WaveLinkFont
  Represents fonts which are used to render text for widgets on screens.

- **Class WaveLinkIO**

  class com.wavelink.clientui.WaveLinkIO
  Provides the interface to the device for displaying text and querying for input. It also contains the screen image manipulation functionality.

- **Class WaveLinkMenu**

  class com.wavelink.clientui.WaveLinkMenu
  Creates menus on a device.

- **Class WaveLinkMenubarInfo**

  class com.wavelink.clientui.WaveLinkMenubarInfo
  Creates and manipulates a list of menus that can be displayed by a menubar widget.

- **Class WaveLinkMessageBox**

  class com.wavelink.clientui.WaveLinkMessageBox
  Creates, displays, and destroys messages on the device.

- **Class WaveLinkScribblePad**

  class com.wavelink.clientui.WaveLinkScribblePad
  Displays a drawing pad and saves the newly created drawing to a specific (.jpg or .bmp) file.

- **Class WaveLinkTerminal**

  class com.wavelink.clientui.WaveLinkTerminal
  Provides current terminal state information and alters certain terminal options.

- **Class WaveLinkTone**

  class com.wavelink.clientui.WaveLinkTone
  Creates tone configurations for the device.

- **Class WaveLinkWidget**

  class com.wavelink.clientui.WaveLinkWidget
  Builds and manipulates all types of widget objects. DEPRECATED.

- **Class WaveLinkWidgetCollection**

  class com.wavelink.clientui.WaveLinkWidgetCollection
  Logically groups widget objects together. DEPRECATED.

- **Class WidgetOptions**

  class com.wavelink.clientui.WidgetOptions
  Provides options for all the Wavelink widgets.

Class ObjectHandle

`com.wavelink.clientui`

The public Class **ObjectHandle** is supported on: CE

The ObjectHandle class provides a variety of methods used to interact with everything relating to widgets in Studio.

Methods

addListener	getWidgetOptions
changeEnable	setCoordinateType
changeShow	setInitialState
getCoordinateType	setWidgetOptions
getInitialState	

Method Summary

addListener Method

- Adds a new event listener to the widget.

changeEnable Method

- Immediately changes the enabled/disabled state of the widget.

ChangeShow Method

- Immediately changes the show/hide state of the widget.

getCoordinateType Method

- See Class WaveLinkConstants.Coord.

getInitialState Method

- Gets the initial state of an object. See Class WaveLinkConstants.State.

getWidgetOptions Method

- Returns the WidgetOptions object used by this ObjectHandle object.

setCoordinateType Method

- Sets the coordinate type used by an object. See Class WaveLinkConstants.Coord.

setInitialState Method

- Sets the initial state of an object. See Class WaveLinkConstants.State.

setWidgetOptions Method

- Sets the default widget options for the widgets.

addListener Method

This member of Class ObjectHandle is supported on: CE

The addListener Method adds a new event listener to the widget.

Syntax

```
public void addListener(WaveLinkListener listener)
```

Parameters

listener	The object to which events are sent

Remarks

Each ObjectHandle can have its own listener(s) to send events to – including WaveLinkScreen and WidgetHandles.

changeEnable Method

This member of Class ObjectHandle is supported on: CE

The changeEnable Method changes the enabled/disabled state of the widget.

Syntax

```
public abstract void changeEnable(boolean enable)
                                  throws WaveLinkError
```

Parameters

enable	Whether to enable or disable the widget

Throws

WaveLinkError

Remarks

Calling changeEnable(true) will enable an object, while changeEnable(false) will disable it, preventing events from being generated for that object.

changeShow Method

This member of Class ObjectHandle is supported on: CE

The changeShow Method changes the show/hide state of the widget.

Syntax

```
public abstract void changeShow(boolean show)
                                throws WaveLinkError
```

Parameters

show	Whether to show or hide the widget

Throws

WaveLinkError

Remarks

Calling changeShow(true) will make the object visible, while changeShow(false) will hide the object. This applies to both WaveLinkScreen and widgets. If a WaveLinkScreen object is hidden, all the widgets on that screen will be hidden.

getCoordinateType Method

This member of Class ObjectHandle is supported on: CE

The getCoordinateType Method returns the type of coordinates used to position this object.

Syntax

```
public int getCoordinateType()
```

Returns

The coordinate type used (See Class WaveLinkConstants.Coord)

Remarks

Coordinate type can vary between CELL, PIXEL, and PERCENTAGE. The coordinate type defaults to PIXEL for all objects.

getInitialState Method

This member of Class ObjectHandle is supported on: CE

The getInitialState Method gets the initial state of an object.

Syntax

```
public int getInitialState()
```

Returns

The initial state of the object (See Class WaveLinkConstants.State)

Remarks

See setInitialState Method to set the initial state.

getWidgetOptions Method

This member of Class ObjectHandle is supported on: CE

The getWidgetOptions Method gets the default options for the widgets.

Syntax

```
public abstract WidgetOptions getWidgetOptions()
```

Returns

The widget options

Remarks

WidgetOptions are unique to every ObjectHandle, but are inherited by widgets from WaveLinkScreen objects.

setCoordinateType Method

This member of Class ObjectHandle is supported on: CE

The setCoordinateType Method sets the coordinate type used by an object.

Syntax

```
public abstract void setCoordinateType(int coord)
```

Parameters

coord	The new coord type (See Class WaveLinkConstants.Coord)

Remarks

Coordinate type can vary between CELL, PIXEL, and PERCENTAGE. The coordinate type defaults to PIXEL for all objects.

setInitialState Method

This member of Class ObjectHandle is supported on: CE

The setInitialState Method sets the initial state of an object.

Syntax

```
public void setInitialState(int state)
```

Parameters

state	the initial state (See Class WaveLinkConstants.State)

Remarks

The initial state of an object can be set using this method. See the constants list for all the possible states an object can be initially set to.

setWidgetOptions Method

This member of Class ObjectHandle is supported on: CE

The setWidgetOptions Method sets the default widget options for the widgets.

Syntax

```
public abstract void setWidgetOptions(WidgetOptions options)
```

Parameters

options	the widget options

Remarks

Each ObjectHandle (and all classes that inherit from it) can have their unique WidgetOptions set using this method.

Interface WaveLinkListener

The public interface WaveLinkListener is supported on: CE

This interface must be implemented by a class expecting to receive events from widgets. It must implement at least the widgetEvent() method to be able to process events.

Method

widgetEvent

Method Summary

widgetEvent

- Called by Studio when an event is received for which this object is specified as the listener.

Example

See WaveLinkScreen Samples

widgetEvent Method

This method is supported on: CE

The widgetEvent Method is called when an event is received for which this object is specified as the listener.

Syntax

```
public void widgetEvent(WaveLinkEvent e)
```

Parameters

e	WaveLinkEvent object

Class WaveLinkScreen

```
com.wavelink.clientui
```

The public Class **WaveLinkScreen** is supported on: CE

The WaveLinkScreen class is the basis for all widgets in Studio. Once a WaveLinkScreen object is created, widgets can be added using the create functions. After calling saveToDevice(), startEventLoop() can be called to show the widgets on the device screen.

WaveLinkScreen inherits from Class **ObjectHandle**.

Constructors

```
public WaveLinkScreen()
```

```
Used to create a new WaveLinkScreen object to add widgets
to.
```

Methods

changeEnable	createProgressBar
changeShow	createRadioButton
createBitmap	createRepeaterButton
createButton	getCoordinateType
createCheckbox	getCurrentValue
createCombobox	getWidgetOptions
createField	removeFromDevice
createHotspot	saveToDevice
createHTML	setCoordinateType
createImager	setWidgetOptions
createLabel	startEventLoop
createListbox	stopEventLoop
createPopupMenu	widgetEvent

Method Summary

changeEnable Method

- Immediately changes the enabled/disabled state of the widgets on the screen.

changeShow Method

- Displays the screen and processes input events.

createBitmap Method

- Creates a bitmap widget.

createButton Method

- Creates a button widget.

createCheckbox Method

- Creates a checkbox widget.

createCombobox Method

- Creates a combobox widget.

createField Method

- Creates a field widget.

createHotspot Method

- Creates a hotspot widget.

createHTML Method

- Creates an HTML widget.

createLabel Method

- Creates a label widget.

createListbox Method

- Creates a listbox widget.

createPopupMenu Method

- Creates a popup menu widget.

createProgressBar Method

- Creates a progress bar widget.

createRadioButton Method

- Creates a radio button widget.

createRepeaterButton Method

- Creates a repeater button widget.

getCoordinateType Method

- Gets the coordinate type used by this screen – see Class WaveLinkConstants.Coord.

getCurrentValue Method

- Returns the widget's current value.

getWidgetOptions Method

- Gets the default widget options for the widgets.

removeFromDevice Method

- Removes this screen from the device.

saveToDevice Method

- Saves the screen on the device. Note that this method must be called before showing the screen.

setCoordinateType Method

- Sets the coordinate type used by the screen – see Class WaveLinkConstants.Coord.

setWidgetOptions Method

- Sets the default widget options for the widgets.

startEventLoop Method

- Starts processing events for the screen.

stopEventLoop Method

- Stops processing events for the screen.

Remarks

All of the createWidget methods return handles (See Class WidgetHandle) to the widgets that can later be used to manipulate specific options for those widgets after they are displayed. A screen must be saved to the device (using the saveToDevice Method) before most WidgetHandle methods can be called.

Example

See WaveLinkScreen Samples

changeEnable Method

This member of Class WaveLinkScreen is supported on: CE

The changeEnable Method changes the enabled/disabled state of the widgets on the screen.

Syntax

```
public void changeEnable(boolean enable)
                    throws WaveLinkError
```

Parameters

enable	Whether to enable or disable the widget

Throws

WaveLinkError

Remarks

Calling this method will either enable or disable all the widgets on the screen.

changeShow Method

This member of Class WaveLinkScreen is supported on: CE

The changeShow Method displays the screen and processes input events.

Syntax

```
public void changeShow(boolean show)
                throws WaveLinkError
```

Parameters

show	True to show widgets, false to hide widgets

Throws

WaveLinkError

Remarks

Calling this method will either show or hide all the widgets on the screen.

createBitmap Method

This member of Class WaveLinkScreen is supported on: CE

The createBitmap Method creates a bitmap widget.

Syntax

```
public BitmapHandle createBitmap(Region r,
                java.lang.String file,
                WidgetOptions options)
```

Parameters

r	The widget region
file	The bitmap file
options	The widget options or null for the screen default

Returns

BitmapHandle

Throws

WaveLinkError

Remarks

WidgetOptions are inherited from the screen, but can be overridden by the supplied WidgetOptions object.

createButton Method

This member of Class WaveLinkScreen is supported on: CE

The createButton Method creates a button widget.

Syntax

```
public ButtonHandle createButton(Region r,
                java.lang.String text,
                WidgetOptions options,
                WaveLinkListener listener)
```

Parameters

r	The widget region
text	The button text
options	The widget options or null for the screen default
listener	The WaveLinkListener or null for none

Returns

ButtonHandle

Throws

WaveLinkError

Remarks

WidgetOptions are inherited from the screen, but can be overridden by the supplied WidgetOptions object.

createCheckbox Method

This member of Class WaveLinkScreen is supported on: CE

The createCheckbox Method creates a checkbox widget.

Syntax

```
public CheckboxHandle createCheckbox(Region r,
                java.lang.String text,
                WidgetOptions options,
                int state,
                WaveLinkListener listener)
```

Parameters

r	The widget region
text	The display text
options	The widget options or null for the screen default
state	The initial state – see Class WaveLinkConstants.State
listener	The WaveLinkListener or null for none

Returns

CheckboxHandle

Throws

WaveLinkError

Remarks

WidgetOptions are inherited from the screen, but can be overridden by the supplied WidgetOptions object.

createCombobox Method

This member of Class WaveLinkScreen is supported on: CE

The createCombobox Method creates a combobox widget.

Syntax

```
public ComboboxHandle createCombobox(Region r,
                java.lang.String name,
                int selected,
                WidgetOptions options,
                WaveLinkListener listener)
```

Parameters

r	The widget region
name	The menu name
selected	The initial selected index
options	The widget options or null for the screen default
listener	The WaveLinkListener or null for none

Returns

ComboboxHandle

Throws

WaveLinkError

Remarks

WidgetOptions are inherited from the screen, but can be overridden by the supplied WidgetOptions object.

createField Method

This member of Class WaveLinkScreen is supported on: CE

The createField Method creates a field widget.

Syntax

```
public FieldHandle createField(Region r,
                java.lang.String text,
                WidgetOptions options,
                WaveLinkListener listener)
```

Parameters

r	The widget region
text	The field text
options	The widget options or null for the screen default
listener	The WaveLinkListener or null for none

Returns

FieldHandle

Throws

WaveLinkError

Remarks

WidgetOptions are inherited from the screen, but can be overridden by the supplied WidgetOptions object.

createHotspot Method

This member of Class WaveLinkScreen is supported on: CE

The createHotspot Method creates a hotspot widget. Hotspots are not visible but can create events when interacted with by the user.

Syntax

```
public HotspotHandle createHotspot(Region r,
                java.lang.String name,
                WidgetOptions options,
                WaveLinkListener listener)
```

Parameters

r	The widget region
name	The hotspot name
options	The widget options or null for the screen default
listener	The WaveLinkListener or null for none

Returns

HotspotHandle

Throws

WaveLinkError

Remarks

WidgetOptions are inherited from the screen, but can be overridden by the supplied WidgetOptions object.

createHTML Method

This member of Class WaveLinkScreen is supported on: CE

The createHTML Method creates an HTML widget.

Syntax

```
public HTMLHandle createHTML(Region r,
                             java.lang.String text,
                             WidgetOptions options)
```

Parameters

r	The widget region
text	The widget default text
options	The widget options or null for the screen default

Returns

HTMLHandle

Throws

WaveLinkError

Remarks

WidgetOptions are inherited from the screen, but can be overridden by the supplied WidgetOptions object.

The *text* supplied for this object is used directly as HTML to render in the widget. It is recommended, due to size constraints of this field, to use a META REFRESH tag in the value to cause the widget to immediately refresh to show a page from a local web server.

Example

```
String html = "<html><head><meta http-equiv=\"refresh\"
content=\"1;url=http://www.pocketpc.com/\"></head><body>Loading
www.pocketpc.com...</body></html>";
   wlscreen.createHTML(new Region(10, 10, 200, 200), html,
   null);
```

createLabel Method

This member of Class WaveLinkScreen is supported on: CE

The createLabel Method creates a label widget.

Syntax

```
public LabelHandle createLabel(Region r,
                java.lang.String text,
                WidgetOptions options)
```

Parameters

r	The widget region
text	The label text
options	The widget options or null for the screen default

Returns

:LabelHandle

Throws

WaveLinkError

Remarks

WidgetOptions are inherited from the screen, but can be overridden by the supplied WidgetOptions object.

createListbox Method

This member of Class WaveLinkScreen is supported on: CE

The createListbox Method creates a listbox widget.

Syntax

```
public ListboxHandle createListbox(Region r,
                java.lang.String name,
                int selected,
                WidgetOptions options,
                WaveLinkListener listener)
```

Parameters

r	The widget region
name	The menu name
selected	The initial selected index
options	The widget options or null for the screen default
listener	The WaveLinkListener or null for none

Returns

ListboxHandle

Throws

WaveLinkError

Remarks

WidgetOptions are inherited from the screen, but can be overridden by the supplied WidgetOptions object.

createPopupMenu Method

This member of Class WaveLinkScreen is supported on: CE

The createPopupMenu Method creates a popup menu widget.

Syntax

```
public PopupHandle createPopupMenu(WaveLinkMenuList menu,
                WidgetOptions options,
                WaveLinkListener listener)
```

Parameters

menu	The menu list
options	The widget options or null for the screen default
listener	The WaveLinkListener or null for none

Returns

PopupHandle

Throws

WaveLinkError

Remarks

The WaveLinkMenuList is a combination of menus created with WaveLinkMenu. The title of each WaveLinkMenu will be shown in the master popup menu, while the submenus will contain the menu lines of each respective menu. All the menus referenced in the WaveLinkMenuList object must be saved to the device before a PopupMenu is created.

createProgressBar Method

This member of Class WaveLinkScreen is supported on: CE

The createProgressBar Method creates a progress bar widget.

Syntax

```
public ProgressBarHandle createProgressBar(Region r,
                int progress,
                WidgetOptions options)
```

Parameters

r	The widget region
progress	The widget progress
options	The widget options or null for the screen default

Returns

ProgressBarHandle

Throws

WaveLinkError

Remarks

The progress parameter is an integer from 0 to 100. If it is 0, the progress bar will be empty, while at 100 it will be full.

createRadioButton Method

This member of Class WaveLinkScreen is supported on: CE

The createRadioButton Method creates a radio button widget.

Syntax

```
public RadioButtonHandle createRadioButton(Region r,
                java.lang.String name,
                int selected,
                WidgetOptions options,
                WaveLinkListener listener)
```

Parameters

r	The widget region
name	The menu name (see Remarks)
selected	The default initial selected index or 0 for none
options	The widget options or null for the screen default
listener	The WaveLinkListener or null for none

Returns

RadioButtonHandle

Throws

WaveLinkError

Remarks

The second parameter for the menu name is in reference to a WaveLinkMenu that you have previously saved to the device. The menu name used when storing the menu should be used in this parameter.

A RadioButton widget shows the radio buttons horizontally by default. To display vertically, add WidgetOptions with the field RADIO_ALIGNVERT (Class WaveLinkConstants) specified.

Example

```
wlscreen.createRadioButton(new Region(x,y,w,h), "radiomnu",
1, widgOpts,
defaultListener).getWidgetOptions().addFlags(WaveLinkConstan
ts.RADIO_ALIGNVERT);
```

createRepeaterButton Method

This member of Class WaveLinkScreen is supported on: CE

The createRepeaterButton Method creates a repeater button widget.

Syntax

```
public RepeaterButtonHandle createRepeaterButton(Region r,
                int type,
                WidgetOptions options,
                WaveLinkListener listener)
```

Parameters

r	The widget region
type	The repeater type
options	The widget options or null for the screen default
listener	The WaveLinkListener or null for none

Returns

RepeaterButtonHandle

Throws

WaveLinkError

Remarks

The type parameter can be set from WaveLinkConstants.Repeater. A RepeaterButton is simply a button with a pre-drawn up, down, left, or right arrow shown.

getCoordinateType Method

This member of Class WaveLinkScreen is supported on: CE

The getCoordinateType Method gets the coordinate type used by the screen.

Syntax

```
public int getCoordinateType()
```

Returns

The coordinate type used for the screen

Remarks

The default coordinate type is WaveLinkConstants.Coord.PIXEL.

getCurrentValue Method

This member of Class WaveLinkScreen is supported on: CE

The getCurrentValue Method returns the value from the last event directed to the screen. Screens receive events when no widget has focus.

Syntax

```
public java.lang.String getCurrentValue()
```

Returns

The value of the last event

Remarks

The value returned by this method is equivalent to what a WaveLinkEvent object's getEvent Method would return inside a WaveLinkListener object listening for screen events.

getWidgetOptions Method

This member of Class WaveLinkScreen is supported on: CE

The getWidgetOptions Method gets the default widget options used for the screen.

Syntax

```
public WidgetOptions getWidgetOptions()
```

Returns

The widget options

Remarks

All widgets created from this screen object will inherit its WidgetOptions.

removeFromDevice Method

This member of Class WaveLinkScreen is supported on: CE

The removeFromDevice Method removes the screen from the device.

Syntax

```
public void removeFromDevice()
```

Throws

WaveLinkError

Remarks

Call this function to cleanup a screen no longer needed.

saveToDevice Method

This member of Class WaveLinkScreen is supported on: CE

The saveToDevice Method saves the screen on the device. Note that this method must be called before showing the screen.

Syntax

```
public void saveToDevice()
```

Throws

WaveLinkError

Remarks

After all widgets have been added to the screen, this method will save them to the device so they can be shown.

setCoordinateType Method

This member of Class WaveLinkScreen is supported on: CE

The setCoordinateType Method sets the coordinate type used by the screen. See also Class WaveLinkConstants.Coord.

Syntax

```
public void setCoordinateType(int coord)
```

Parameters

coord	The new coord type

Remarks

By default, the coordinate type for a new WaveLinkScreen is WaveLinkConstants.Coord.PIXEL.

setWidgetOptions Method

This member of Class WaveLinkScreen is supported on: CE

The setWidgetOptions Method sets the default widget options.

Syntax

```
public void setWidgetOptions(WidgetOptions options)
```

Parameters

options	The widget options

Remarks

All newly created widgets will inherit from the screen's current WidgetOptions.

startEventLoop Method

This member of Class WaveLinkScreen is supported on: CE

The startEventLoop Method starts processing events for the screen.

Syntax

```
public void startEventLoop()
```

Throws

WaveLinkError

Remarks

Once this method is called, execution from the calling method is halted until the screen's stopEventLoop Method is called. Therefore, it is helpful to have a listener set up to call stopEventLoop when, for example, a "quit" button is pressed. Once stopEventLoop is called, execution will resume from the location startEventLoop was called from.

stopEventLoop Method

This member of Class WaveLinkScreen is supported on: CE

The stopEventLoop Method stops processing events from the screen.

Syntax

```
public void stopEventLoop()
```

Remarks

See startEventLoop for more information.

WaveLinkScreen Samples

The following sample code demonstrates the implementation and capabilities of the Class WaveLinkScreen and its associated methods.

```
WaveLinkScreen wlscreen;
ButtonHandle myButton;

void SingleTest() throws WaveLinkError {
        //Initialize the screen object
        wlscreen = new WaveLinkScreen();

        //Create the widget options to be used for the screen
        WidgetOptions myWidgOpts = new WidgetOptions();
        //Set the font
        myWidgOpts.setFont(new WaveLinkFont("Arial", 12,
        WaveLinkFont.STYLE_BOLD));

        //Set the default justification of text
        myWidgOpts.setJustification(WaveLinkConstants.
        Justify.RIGHT);


        //Set them to the screen
        wlscreen.setWidgetOptions(myWidgOpts);

        //Create an object using our listener class to receive
        //events
        WaveLinkListener wlListener = new myListener();

        //This button will inherit the widget options from the
        //screen since null is specified.
       myButton = wlscreen.createButton(new Region(10, 10, 100,
        50), "Hello!", null, wlListener);


        //Save the screen to the device (must be done before we
        //start the event loop)
        wlscreen.saveToDevice();

        //Show the screen and start listening for events
        wlscreen.startEventLoop();
}
public class myListener implements WaveLinkListener {
        public void widgetEvent(WaveLinkEvent e) {
```

```
            //Will show the text on the widget this listener is
            //for
            System.out.println(e.getEvent());
            try {
                //Default text on the button was Hello!
                myButton.changeText("Hi!");
            } catch (WaveLinkError wlErr) {
                //Unable to change the text, exit the screen
                //because some error happened
                wlscreen.stopEventLoop();
            }
           //If the user has already clicked the button and had
            //the text changed to Hi!, exit.
            if (e.getEvent().equals("Hi!")) {
                wlscreen.stopEventLoop();
            }
        }
    }
```

Class WidgetHandle

`com.wavelink.clientui`

The public Class **WidgetHandle** is supported on:

This class forms the basis of all widgets in Studio and includes the methods that are common to all of them.

WidgetHandle inherits from Class **ObjectHandle**.

Methods

changeEnable	getWidgetOptions
changeShow	grabFocus
equals	setCoordinateType
getCoordinateType	setInitialFocus
getInitialFocus	setWidgetOptions

Method Summary

changeEnable Method

- Immediately changes the enabled/disabled state of the widget. The widget must be saved to the device before calling this method.

changeShow Method

- Immediately changes the show/hide state of the widget. The widget must be saved to the device before calling this method.

equals Method

- Indicates whether an object is "equal" to this WidgetHandle.

getCoordinateType Method

- Used to determine the coordinate type being used to place this widget on the screen. See Class WaveLinkConstants.Coord.

getInitialFocus Method

- Gets the initial focus state.

getWidgetOptions Method

- Gets the default widget options for the widgets.

grabFocus Method

- Grabs the focus of the widget. The widget must be saved to the device before calling this method.

setCoordinateType Method

- Sets the coordinate type used to place this widget. Has no effect if it is different from its parent WaveLinkScreen coordinate type.

setInitialFocus Method

- Sets the initial focus state.

setWidgetOptions Method

- Sets the default widget options for the widgets.

Remarks

Some widgets have special methods in addition to these WidgetHandle methods, such as ButtonHandle or LabelHandle. Since those special handles extend WidgetHandle, the methods in this class can be used on any special handle.

changeEnable Method

This member of Class WidgetHandle is supported on: CE

The changeEnable Method changes the enabled/disabled state of the widget. The widget must be saved to the device before calling this method.

Syntax

```
public void changeEnable(boolean enable)
```

Parameters

enable	Whether to enable or disable the widget

Throws

WaveLinkError

Remarks

Disabling a widget will prevent the widget from receiving events and will "gray" it out (when applicable).

changeShow Method

This member of Class WidgetHandle is supported on: CE

The changeShow Method changes the show/hide state of the widget. The widget must be saved to the device before calling this method.

Syntax

```
public void changeShow(boolean show)
```

Parameters

show	Whether to show or hide the widget

Throws

WaveLinkError

equals Method

This member of Class WidgetHandle is supported on: CE

The equals Method indicates whether an object is "equal" to the WidgetHandle.

Syntax

```
public boolean equals(java.lang.Object obj)
```

Parameters

obj	Another WidgetHandle

Returns

True if the sent object has the same values as this WidgetHandle, false if not.

getCoordinateType Method

This member of Class WidgetHandle is supported on: CE

The getCoordinateType Method is used to determine the coordinate type being used to place this widget on the screen. See Class WaveLinkConstants.Coord.

Syntax

```
public int getCoordinateType()
```

Returns

The coordinate type

Remarks

Widgets will, by default, be positioned by pixels. (See WaveLinkConstants.Coord.PIXEL)

getInitialFocus Method

This member of Class WidgetHandle is supported on: CE

The getInitialFocus Method gets the initial focus state.

Syntax

```
public boolean getInitialFocus()
```

Returns

True if this widget will have the initial focus, false if not

Remarks

Use setInitialFocus() to force this widget to have the initial focus on a screen.

getWidgetOptions Method

This member of Class WidgetHandle is supported on: CE

The getWidgetOptions Method gets the WidgetOptions object being used by this widget.

Syntax

```
public WidgetOptions getWidgetOptions()
```

Returns

The widget options

grabFocus Method

This member of Class WidgetHandle is supported on: CE

The grabFocus Method grabs the focus of the widget. The widget must be saved to the device before calling this method.

Syntax

```
public void grabFocus()
```

Throws

WaveLinkError

setCoordinateType Method

This member of Class WidgetHandle is supported on: CE

The setCoordinateType Method sets the coordinate type used to place this widget. It has no effect if it is different from its parent WaveLinkScreen coordinate type.

Syntax

```
public void setCoordinateType(int coord)
```

Parameters

coord	The new coord type

Remarks

The coordinate type should be specified on the WaveLinkScreen object normally, but it is possible to specify it on a widget-by-widget basis.

setInitialFocus Method

This member of Class WidgetHandle is supported on: CE

The setInitialFocus Method sets the initial focus state.

Syntax

```
public void setInitialFocus(boolean initFocus)
```

Parameters

initFocus	The initial focus flag

Remarks

This will cause the widget to have the initial focus when the screen is saved to the device.

setWidgetOptions Method

This member of Class WidgetHandle is supported on: CE

The setWidgetOptions Method sets the default widget options for the widgets.

Syntax

```
public void setWidgetOptions(WidgetOptions options)
```

Parameters

options	The widgets options

Remarks

Use this method to change specific options for a widget to override settings inherited from the WaveLinkScreen it was created from.

Class BitmapHandle

`com.wavelink.clientui`

The public Class **BitmapHandle** is supported on: CE

This class adds a single method to change the bitmap being shown.

BitmapHandle inherits from Class **WidgetHandle**.

Method Summary

changeBitmap Method

- Changes the bitmap file for the widget. The widget must already be saved to the device.

Throws

WaveLinkError

changeBitmap Method

This member of Class BitmapHandle is supported on: CE

The changeBitmap Method changes the bitmap file for the widget. The widget must already be saved to the device.

Syntax

```
public void changeBitmap(java.lang.String newFile)
```

Parameters

newFile	The new bitmap file

Throws

WaveLinkError

Remarks

Use this function to change the display of a bitmap widget to a different picture.

Class ButtonHandle

`com.wavelink.clientui`

The public class **ButtonHandle** is supported on: CE

This special handle for button widgets provides the changeText method to change the text shown on a button.

ButtonHandle inherits from Class **WidgetHandle**.

Method Summary

changeText Method

- Changes the text on the widget. The widget must be saved to the device before calling this method.

Throws

WaveLinkError

Example

See WaveLinkScreen Samples

changeText Method

This member of Class ButtonHandle is supported on: CE

The changeText Method changes the text on the widget. The widget must be saved to the device before calling this method.

Syntax

```
public void changeText(java.lang.String newText)
```

Parameters

newText	The new text

Throws

WaveLinkError

Class HTMLHandle

`com.wavelink.clientui`

The public Class **HTMLHandle** is supported on: CE

This special handle for button widgets provides the changeText method to change the text shown on a button.

HTMLHandle inherits from Class **WidgetHandle**.

Method Summary

changeText Method

- Changes the text on the widget. The widget must be saved to the device before calling this method.

Throws

WaveLinkError

Example

See createHTML Method

changeText Method

This member of Class HTMLHandle is supported on: CE

The changeText Method changes the text on the widget. The widget must be saved to the device before calling this method.

Syntax

```
public void changeText(java.lang.String newText)
```

Parameters

newText	The new text

Throws

WaveLinkError

Class LabelHandle

`com.wavelink.clientui`

The public class **LabelHandle** is supported on:

This class is used to change the label text for a label widget.

LabelHandle inherits from Class **WidgetHandle**.

Method Summary

changeText Method

- Changes the text on the widget. The widget must be saved to the device before calling this method.

Throws

WaveLinkError

changeText Method

This member of Class LabelHandle is supported on: CE

The changeText Method changes the text on the widget. The widget must be saved to the device before calling this method.

Syntax

```
public void changeText(java.lang.String newText)
```

Parameters

newText	The new text

Throws

WaveLinkError

Class ProgressBarHandle

`com.wavelink.clientui`

The public Class **ProgressBarHandle** is supported on: CE

Provides the changeProgress method to change the amount shown on a ProgressBar widget.

ProgressBarHandle inherits from Class **WidgetHandle**.

Method Summary

changeProgress Method

- Changes the progress on the widget. The widget must be saved to the device before calling this method.

Throws

WaveLinkError

changeProgress Method

This member of Class ProgressBarHandle is supported on: CE

The changeProgress Method changes the progress on the widget. The widget must be saved to the device before calling this method.

Syntax

```
public void changeProgress(int progress)
```

Parameters

progress	The current progress, a number from 0 to 100

Throws

WaveLinkError

Class CheckboxHandle

`com.wavelink.clientui`

The public Class Checkbox Handle is supported on:

This class provides functions for changing the text shown for a checkbox and for obtaining its current value. See Class WaveLinkConstants.CheckboxState.

CheckboxHandle inherits from Class **WidgetHandle**.

Methods

changeText getCurrentValue

Method Summary

changeText Method

- Changes the text on the widget. The widget must be saved to the device before calling this method.

getCurrentValue Method

- Returns the current value of this checkbox. See Class WaveLinkConstants.CheckboxState.

changeText Method

This member of Class CheckboxHandle is supported on: CE

The changeText Method changes the text on the widget. The widget must be saved to the device before calling this method.

Syntax

```
public void changeText(java.lang.String newText)
```

Parameters

newText	The new text

Throws

WaveLinkError

getCurrentValue Method

This member of Class LabelHandle is supported on: CE

The getCurrentValue Method returns the current value of this checkbox. See Class WaveLinkConstants.CheckboxState.

Syntax

```
public int getCurrentValue()
```

Returns

The current state of the checkbox. See the constants list for more information.

Class FieldHandle

`com.wavelink.clientui`

The public Class **FieldHandle** is supported on:

This class provides the capability to obtain or change the text for a field widget.

FieldHandle inherits from Class **WidgetHandle**.

Methods

changeText getCurrentValue

Method Summary

changeText Method

- Changes the text on the widget. The widget must be saved to the device before calling this method.

getCurrentValue Method

- Returns the widget's current value.

changeText Method

This member of Class FieldHandle is supported on: CE

The changeText Method changes the text on the widget. The widget must be saved to the device before calling this method.

Syntax

```
public void changeText(java.lang.String newText)
```

Parameters

newText	The new text

Throws

WaveLinkError

getCurrentValue Method

This member of Class FieldHandle is supported on: CE

The getCurrentValue Method returns the current value of the widget.

Syntax

```
public String getCurrentValue()
```

Returns

The text currently stored in the field

Class MenuHandle

`com.wavelink.clientui`

The public Class **MenuHandle** is supported on: CE

This class provides the capability to change which WaveLinkMenu file is used and which option in the menu is selected.

MenuHandle inherits from Class **WidgetHandle**.

Methods

changeMenu changeSelected

Method Summary

changeMenu Method

- Changes the widgets menu. The widget must be saved to the device before calling this method.

changeSelected Method

- Selects an item in the menu widget. The widget must be saved to the device before calling this method.

changeMenu Method

This member of Class MenuHandle is supported on: CE

The changeMenu Method changes the widget's menu. The widget must be saved to the device before calling this method.

Syntax

```
public void changeMenu(java.lang.String newMenu)
```

Parameters

newMenu	The name of the new menu

Throws

WaveLinkError

changeSelected Method

This member of Class MenuHandle is supported on: CE

The changeSelected Method selects an item in the menu widget. The widget must be saved to the device before calling this method.

Syntax

```
public void changeSelected(int index)
```

Parameters

index	The index of the item to select

Throws

WaveLinkError

Class ComboboxHandle

`com.wavelink.clientui`

The public class **ComboboxHandle** is supported on: CE

This class provides the capability to get the currently selected item from a combobox widget.

ComboboxHandle inherits from Class **MenuHandle**.

Method Summary

getCurrentValue

- Returns the widget's current value.

Throws

WaveLinkError

getCurrentValue Method

This member of Class ComboboxHandle is supported on: CE

The getCurrentValue Method returns the current value of the widget.

Syntax

```
public int getCurrentValue()
```

Returns

The number of the currently selected option

Class ListBoxHandle

`com.wavelink.clientui`

The public Class **ListBoxHandle** is supported on: CE

This class provides the capability to get the currently selected item from a listbox widget.

ListBoxHandle inherits from Class **MenuHandle**.

Method Summary

getCurrentValue

- Returns the widget's current value.

Throws

WaveLinkError

getCurrentValue Method

This member of Class ListBoxHandle is supported on: CE

The getCurrentValue Method returns the current value of the widget.

Syntax

```
public int getCurrentValue()
```

Returns

The number of the currently selected option.

Class PopupHandle

`com.wavelink.clientui`

The public Class **PopupHandle** is supported on:

This class provides methods to obtain the last option (in a submenu) and title (main menu) references.

PopupHandle inherits from Class **WidgetHandle**.

Methods

getOptionIndex getTitleIndex

Method Summary

getOptionIndex

- Returns the last selected option index.

getTitleIndex

- Returns the last selected title index.

getOptionIndex Method

This member of Class PopupHandle is supported on: CE

The getOptionIndex Method returns the last selected option index.

Syntax

```
public int getOptionIndex()
```

Returns

The number of the option chosen in the submenu

getTitleIndex Method

This member of Class PopupHandle is supported on: CE

The getTitleIndex Method returns the last selected title index.

Syntax

```
public int getTitleIndex()
```

Returns

The number of the selected main popup menu

Class RadioButtonHandle

`com.wavelink.clientui`

The public Class **RadioButtonHandle** is supported on: CE

This class provides a getCurrentValue() method to determine which option is selected in a radio button widget.

RadioButtonHandle inherits from Class **WidgetHandle**.

Method Summary

getCurrentValue

- Returns the widget's current value.

Throws

WaveLinkError

getCurrentValue Method

This member of Class RadioButtonHandle is supported on: CE

The getCurrentValue Method returns the current value of the widget.

Syntax

```
public int getCurrentValue()
```

Returns

The number of the selected option in the menu

Class Region

`com.wavelink.clientui`

The public class Region is supported on: CE

The Class Region represents a widget region.

Constructor

```
public Region(int x,int y,int w,int h)
```

Parameters

x	The x position
y	The y position
w	The width
h	The height

Class WaveLinkAuxPort

`com.wavelink.clientui`

The public class **WaveLinkAuxPort** is supported on all OS types.

The Class WaveLinkAuxPort provides bi-directional access to the serial port of a device.

Constructors

```
public WaveLinkAuxPort()
                        throws Exception

public WaveLinkAuxPort(IWaveLinkSession currentSession)
                        throws Exception
```

Fields

BAUD110	HARDWAREFLOWCTL
BAUD150	NOFLOWCTL
BAUD300	PARITYEVEN
BAUD600	PARITYMARK
BAUD1200	PARITYNONE
BAUD1350	PARITYODD
BAUD2400	PARITYSPACE
BAUD4800	SOFTWAREFLOWCTL
BAUD9600	STOPBITS1
BAUD19200	STOPBITS2
BAUD38400	WLCOM1
DATABITS4	WLCOM2
DATABITS5	WLNOCHAR
DATABITS6	WLNOMAXLENGTH
DATABITS7	WLWAITFOREVER
DATABITS8	

Methods

ConfigurePort	QueryPort

Throws

WaveLinkError

Remarks

You must first configure a serial port using the ConfigurePort Method before using the QueryPort Method.

NOTE The main communication port between the server and the client is configured and accessed automatically. The WaveLinkAuxPort object is provided for the auxiliary port on the device, typically a printer port.

Field Detail

BAUD110

```
public static final int BAUD110
```

Baud rate - 110000 baud connection.

BAUD 150

```
public static final int BAUD150
```

Baud rate - 150 baud connection.

BAUD 300

```
public static final int BAUD300
```

Baud rate - 300 baud connection.

BAUD 600

```
public static final int BAUD600
```

Baud rate - 600 baud connection.

BAUD1200

```
public static final int BAUD1200
```

Baud rate - 1200 baud connection.

BAUD 1350

```
public static final int BAUD1350
```

Baud rate - 1350 baud connection.

BAUD 2400

```
public static final int BAUD2400
```

Baud rate - 2400 baud connection.

BAUD 4800

```
public static final int BAUD4800
```

Baud rate - 4800 baud connection.

BAUD 9600

```
public static final int BAUD9600
```

Baud rate - 9600 baud connection.

BAUD 19200

```
public static final int BAUD19200
```

Baud rate - 19200 baud connection.

BAUD 38400

```
public static final int BAUD38400
```

Baud rate - 38400 baud connection.

DATABITS4

```
public static final int DATABITS4
```

Stop Bit Setting - 4 data bits.

DATABITS5

```
public static final int DATABITS5
```

Stop Bit Setting - 5 data bits.

DATABITS6

```
public static final int DATABITS6
```

Stop Bit Setting - 6 data bits.

DATABITS7

```
public static final int DATABITS7
```

Stop Bit Setting - 7 data bits.

DATABITS8

```
public static final int DATABITS8
```

Stop Bit Setting - 8 data bits.

HARDWAREFLOWCTL

```
public static final int
```

Flow Control Setting - Hardware flow control.

NOFLOWCTL

```
public static final int NOFLOWCTL
```

Flow Control Setting - No flow control.

PARITYEVEN

```
public static final int PARITYEVEN
```

Parity Setting - Even parity.

PARITYMARK

```
public static final int PARITYMARK
```

Parity Setting - Mark parity.

PARITYNONE

```
public static final int PARITYNONE
```

Parity Setting - No parity.

PARITYODD

```
public static final int PARITYODD
```

Parity Setting - Odd parity.

PARITYSPACE

```
public static final int PARITYSPACE
```

Parity Setting - Space parity.

SOFTWAREFLOWCTL

```
public static final int SOFTWAREFLOWCTL
```

Flow Control Setting - Software flow control.

STOPBITS1

```
public static final int STOPBITS1
```

Stop Bit Setting - One stop bit.

STOPBITS2

```
public static final int STOPBITS2
```

Stop Bit Setting - Two stop bits.

WLCOM1

```
public static final int WLCOM1
```

COM PORT - COM port 1 (serial port on Palm devices).

WLCOM2

```
public static final int WLCOM2
```

COM PORT - COM port 2 (infrared port on Palm devices).

WLNOCHAR

```
public static final char WLNOCHAR
```

Query Setting - No start/end character.

WLNOMAXLENGTH

```
public static final int WLNOMAXLENGTH
```

Query Setting - No query maximum length for the serial port.

WLWAITFOREVER

```
public static final int WLWAITFOREVER
```

Query Setting - No query timeout for the serial port.

Method Summary

ConfigurePort Method

- Configures UART settings for device's serial port.

QueryPort Method

- Sends and receives data to and from device's serial port.

ConfigurePort Method

This member of Class WaveLinkAuxPort is supported on all OS types.

The ConfigurePort Method configures the basic UART settings for the serial port of a device before calling the QueryPort Method.

Syntax

```
public void ConfigurePort(int port, int baud, int databits,
                          int stopBits, int parity,
                          int flowCtrl)
            throws WaveLinkError
                   IllegalArgumentException
```

Parameters

port	The serial port of the device which UART settings you wish to configure (see Remarks)
baud	The UART baud rate of the serial port (see Remarks)
dataBits	The UART data bit setting of the serial port (see Remarks)
stopBits	The UART stop bit setting of the serial port (see Remarks))
parity	The UART parity setting of the serial port (see Remarks)
flowControl	The UART flow control setting of the serial port (see Remarks)

Throws

WaveLinkError

IllegalArgumentException

Remarks

A serial port must first be configured using the ConfigurePort Method before the port may be queried for input using the QueryPort Method.

The possible values for *port* are:

WLCOM1	- COM port 1 (serial port on Palm devices)
WLCOM2	- COM port 2 (infrared port on Palm devices)

The possible values for *baud* are:

BAUD110	- 110000 BBS baud
BAUD150	- 150 BBS baud

BAUD300	- 300 BBS baud
BAUD600	- 600 BBS baud
BAUD1200	- 1200 BBS baud
BAUD1350	- 1350 BBS baud
BAUD2400	- 2400 BBS baud
BAUD4800	- 4800 BBS baud
BAUD9600	- 9600 BBS baud
BAUD19200	- 19200 BBS baud
BAUD38400	- 38400 BBS baud

The possible values for *dataBits* are:

DATABITS4	- 4 data bits
DATABITS5	- 5 data bits
DATABITS6	- 6 data bits
DATABITS7	- 7 data bits
DATABITS8	- 8 data bits

The possible values for *stopBits* are:

STOPBITS1	- 1 stop bit
STOPBITS2	- 2 stop bits

The possible values for *parity* are:

PARITYEVEN	- Even parity
PARITYMARK	- Mark parity
PARITYNONE	- No parity
PARITYODD	- Odd parity
PARITYSPACE	- Space parity

The possible values for *flowControl* are:

HARDWAREFLOWCTL	- Hardware based flow control
NOFLOWCTL	- No flow control
SOFTWAREFLOWCTL	- Software based flow control

See the Constant Values appendix for the numeric values returned by the function.

Example

See the QueryPort Method

QueryPort Method

This member of Class WaveLinkAuxPort is supported on all OS types.

The QueryPort Method queries a the serial port of a device for input. The QueryPort Method also sends data to the device's serial port.

Syntax

```
public String QueryPort(int port, char start,
                        char end, int maxLength,
                        int timeout, String request)
            throws WaveLinkError
                   IllegalArgumentException
```

Parameters

port	The communicating serial port of the device (see Remarks)
start	The starting character for returned data that is framed. Any data received prior to the starting character is discarded (see Remarks)
end	The ending character for returned data that is framed. Input terminates immediately after the ending character is received (see Remarks)
maxLength	The maximum number of characters that may be received before input automatically returns from the serial port query (see Remarks)
timeout	The maximum amount of time (in seconds) that may pass before input automatically returns from the serial port query (see Remarks)
request	Data that is sent to the serial port before any query input is returned. Set this value to null if you do not wish to send data to the serial port before it is queried (see Remarks)

Returns

The response from the serial port of the device

Throws

WaveLinkError

IllegalArgumentException

Remarks

A serial port must first be configured using the ConfigurePort Method before the port may be queried using the QueryPort Method. This method may be used with RF-based printers when greater control over communications is required.

The possible values for *port* are:

WLCOM1	- COM port 1 (serial port on Palm devices)
WLCOM2	- COM port 2 (infrared port on Palm devices)

For the *start* parameter, enter the decimal equivalent of the ASCII character you wish to designate as the starting character of a data frame. If WLNOCHAR is entered, no starting character is defined.

For the *end* parameter, enter the decimal equivalent of the ASCII character you wish to designate as the ending character of a data frame. If WLNOCHAR is entered, no ending character is defined.

NOTE You can define an ending character without defining a starting character.

If NOMAXLENGTH is entered for the *maxLength* parameter, no maximum length is enforced.

If WLWAITFOREVER is entered for the *timeout* parameter, no timeout value is enforced. If zero is entered, any data that is immediately available is read and returned.

The *request* parameter is useful for sending data to devices that require a triggering event before responding, such as a scale. Set this value to null if you do not wish to send data to the serial port before it is queried.

Example

```
WaveLinkAuxPort AuxIface = new WaveLinkAuxPort();
int nPort = 0;
string sANs;
    // You must set up the port before its first use.
    try {
        AuxIface.ConfigurePort(nPort,
                 WaveLinkAuxPort.BAUD1900,
                 WaveLinkAuxPort.DATABITS7,
                 WaveLinkAuxPort.STOPBITS1,
```

```
                WaveLinkAuxPort.PARITYEVEN,
                 WaveLinkAuxPort.HARDWAREFLOWCTL);
    sAns = AuxIface.QueryPort(nPort, 0, 13, -1, 10,
                                "AS1T");
    // Process sAns
}
catch (WaveLinkError wlErr) {
    //Do error handling
}
```

Class WaveLinkBarcode

`com.wavelink.clientui`

The public class **WaveLinkBarcode** is supported on all OS types.

The Class WaveLinkBarcode provides the methods necessary to define a list of barcode configurations for use within your applications. Each individual barcode configuration defines a valid or invalid barcode for input, based on type, state, and length.

Constructors

```
public WaveLinkBarcode()
```

```
public WaveLinkBarcode(IWaveLinkSession currentSession)
```

Fields

B_UPC
BCDISABLED
BCENABLED
CODABAR
CODE_11
CODE_39
CODE_93
CODE_128
CODE_D25
CODE_I25
D25_IATA
defaultValue
EAN_8
EAN_13
MSI
NO_DEFAULT
PDF_417
TO_39
UCC_128
UPC_A
UPC_E0
UPC_E1
WLNOSYMBOLOGY

Methods

AddBarcode
BarcodeCount
BarcodeFileCount
BarcodeFileName
ClearBarcodes
Decode
Default
DeleteBarcodeFile
Expand
GetBarcodeFile
ListBarcodeFiles
MaxLength
MinLength
PullBarcode
PushBarcode
RemoveBarcode
StoreBarcode
Symbology

Remarks

Individual barcode configurations are stored within a WaveLinkBarcode object. Each barcode is defined by type, decode state, expand state, minimum length, and maximum length.

Once defined, a specific barcode configuration remains within a WaveLinkBarcode object until either the object is released, the barcode configuration is removed from the object using the RemoveBarcode Method, or all barcode configurations are cleared from the object using the ClearBarcodes Method. This allows you to modify a single barcode configuration instead of creating a new barcode object for each individual instance.

Field Detail

B_UPC

```
public static final int B_UPC
```

Barcode Type - This constant represents the barcode type B-UPC.

BCDISABLED

```
public static final int BCDISABLED
```

Default Decode State - All barcodes not defined in configuration are disabled.

BCENABLED

```
public static final int BCENABLED
```

Default Decode State - All barcodes not defined in configuration are enabled.

CODABAR

```
public static final int CODABAR
```

Barcode Type - This constant represents the barcode type CodaBar.

CODE_11

```
public static final int CODE_11
```

Barcode Type - This constant represents the barcode type Code-11.

CODE_39

```
public static final int CODE_39
```

Barcode Type - This constant represents the barcode type Code-39.

CODE_93

```
public static final int CODE_93
```

Barcode Type - This constant represents the barcode type Code-93.

CODE_128

```
public static final int CODE_128
```

Barcode Type - This constant represents the barcode type Code-128.

CODE_D25

```
public static final int CODE_D25
```

Barcode Type - This constant represents the barcode type Code-D25.

CODE_I25

```
public static final int CODE_I25
```

Barcode Type - This constant represents the barcode type Code-I25.

D25_IATA

```
public static final int D25_IATA
```

Barcode Type - This constant represents the barcode type D25-IATA.

defaultValue

```
public int defaultValue
```

EAN_8

```
public static final int EAN_8
```

Barcode Type - This constant represents the barcode type EAN-8.

EAN_13

```
public static final int EAN_13
```

Barcode Type - This constant represents the barcode type EAN-13.

MSI

```
public static final int MSI
```

Barcode Type - This constant represents the barcode type MSI.

NO_DEFAULT

```
public static final int NO_DEFAULT
```

Default Decode State - The default state for barcodes not defined in configuration remains unchanged.

PDF_417

```
public static final int PDF_417
```

Barcode Type - This constant represents the barcode type PDF-417.

TO_39

```
public static final int TO_39
```

Barcode Type - This constant represents the barcode type TO-39.

UCC_128

```
public static final int UCC_128
```

Barcode Type - This constant represents the barcode type UCC-128.

UPC_A

```
public static final int UPC_A
```

Barcode Type - This constant represents the barcode type UPC-A.

UPC_E0

```
public static final int UPC_E0
```

Barcode Type - This constant represents the barcode type UPC-E0.

UPC_E1

```
public static final int UPC_E1
```

Barcode Type - This constant represents the barcode type UPC-E1.

WLNOSYMBOLOGY

```
public static final int WLNOSYMBOLOGY
```

Barcode Type - This constant indicates that no symbology is returned.

Method Summary

AddBarcode Method

- Adds a barcode configuration.

BarcodeCount Method

- Returns the total number of barcode configurations.

BarcodeFileCount Method

- Returns the total number of barcode configuration files.

BarcodeFileName Method

- Returns the name of a barcode configuration file.

ClearBarcodes Method

- Clears all barcode configurations.

Decode Method

- Specifies the decode state of a barcode configuration at the specified index.

Default Method

- Returns the default decode state for barcodes not explicitly defined within the current WaveLinkBarcode object.

DeleteBarcodeFile Method

- Removes barcode files from the device.

Expand Method

- Specifies the expand state of a barcode configuration at the specified index.

GetBarcodeFile Method

- Retrieves a barcode configuration file into the current WaveLinkBarcode object.

ListBarcodeFiles Method

- Returns the total number of barcode configuration files and stores the list in the current WaveLinkBarcode object.

MaxLength Method

- Specifies the maximum input length of the barcode configuration at the specified index.

MinLength Method

- Specifies the minimum input length of the barcode configuration at the specified index.

PullBarcode Method

- Restores a barcode file as the device's default barcode configuration.

PushBarcode Method

- Stores the device's default barcode configuration to a file for later restoration.

RemoveBarcode Method

- Removes a barcode configuration

StoreBarcode Method

- Stores the current WaveLinkBarcode object as a file on the device for later use.

Symbology Method

- Sets the symbology type for a specific barcode configuration within the WaveLinkBarcode object.

AddBarcode Method

This member of Class WaveLinkBarcode is supported on all OS types.

The AddBarcode Method adds a barcode configuration to the WaveLinkBarcode object.

Syntax

```
public void AddBarcode(int bcType)
            throws IllegalArgumentException


public void AddBarcode(int bcType, int bcMinLen,
                       int BcMaxLen)
            throws IllegalArgumentException


public void AddBarcode(int bcType, boolean bcExpand,
                       boolean bcDecode, int bcMinLen,
                       int bcMaxLen)
            throws IllegalArgumentException
```

Parameters

bcType	The symbology type to be added (see Remarks)
bcExpand	A true value expands read symbology; a false value does not expand it
bcDecode	A true value decodes scanned barcode; a false value does not decode it
bcMinLen	The minimum accepted barcode length (see Remarks)
bcMaxlen	The maximum accepted barcode length (see Remarks)

Throws

IllegalArgumentException

Remarks

Input length restrictions for barcode configurations can be eliminated by setting both the *bcMinLen* and *bcMaxLen* parameters to zero (0).

The possible values for the *bcType* parameter are:

B_UPC

CODABAR

CODE_11

CODE_39
CODE_93
CODE_128
CODE_D25
CODE_I25
D25_IATA
EAN_8
EAN_13
MSI
PDF_417
TO_39
UCC_128
UPC_A
UPC_E0
UPC_E1
WLNOSYMBOLOGY

NOTE The *bcExpand* parameter is only valid with the UPC_E0 barcode type which may or may not be further expanded following decode. For all other barcode types pass this parameter as false.

The individual barcode configurations are stored within a WaveLinkBarcode object using a zero-based index. (For example, the fourth barcode configuration has an index value of 3).

Once defined, a specific barcode configuration remains within a WaveLinkBarcode object until either the object is released, the barcode configuration is removed from the object using the RemoveBarcode Method, or all barcode configurations are cleared from the object using the ClearBarcodes Method. This allows you to modify a single barcode configuration instead of creating a new barcode object for each individual instance.

To add an additional barcode configuration to a saved barcode file, make the barcode file the current WaveLinkBarcode object using the GetBarcodeFile Method, then use AddBarcode Method.

Example

See the StoreBarcode Method and WaveLinkBarcode Samples.

BarcodeCount Method

This member of Class WaveLinkBarcode is supported on all OS types.

The BarcodeCount Method returns the number of barcode configurations within the current object.

Syntax

```
public int BarcodeCount()
```

Returns

The number of barcode configurations within the current object

Remarks

To count the number of barcode configurations within a barcode file saved on a device, retrieve the file using GetBarcodeFile Method, then use the BarcodeCount Method.

BarcodeFileCount Method

This member of Class WaveLinkBarcode is supported on all OS types.

The BarcodeFileCount Method returns the number of barcode files found by the previous ListBarcodeFiles Method call.

Syntax

```
public int BarcodeFileCount()
```

Returns

The number of barcode files found

Remarks

The BarcodeFileCount Method returns the file count from the last successful call to the ListBarcodeFiles Method. Use the ListBarcodeFiles Method to return the total number of barcode files that are currently stored on a device.

BarcodeFileName Method

This member of Class WaveLinkBarcode is supported on all OS types.

The BarcodeFileName Method returns the name of the barcode file at the specified index in the barcode file list generated by the last call of the ListBarcodeFileName Method.

Syntax

```
public String BarcodeFileName(int bcIndex)
```

Parameters

bcIndex	The zero-based index value of the target barcode configuration (see Remarks)

Returns

The name of the specified barcode file

Remarks

For a complete listing of all barcode files on a device, simply loop the BarcodeFileName Method for the entire number of files returned by BarcodeFileCount Method.

The first barcode file is indexed as zero. For example, to return the file name of the third barcode file in the current WaveLinkBarcode Object, you pass a 2.

ClearBarcodes Method

This member of Class WaveLinkBarcode is supported on all OS types.

The ClearBarcodes Method removes all barcode configurations from the current object.

Syntax

```
public void ClearBarcodes()
```

Remarks

To clear the barcode configurations from a saved barcode file, use the DeleteBarcodeFile Method.

Example

See the StoreBarcode Method

Decode Method

This member of Class WaveLinkBarcode is supported on: DOS/embedded, Palm

The Decode Method sets the decode flag value for the barcode configuration at the specified index.

Syntax

```
public boolean Decode(int bcIndex)

public void Decode (int bcIndex, boolean newDecode)
            throws IllegalArgumentException
```

Parameters

bcIndex	The zero-based index value for the target barcode configuration
newDecode	A boolean true or false value that determines the decode value (see Remarks)

Returns

The decode flag value as boolean

Throws

WaveLinkError

IllegalArgumentException

Remarks

For the *newDecode* parameter, a true value decodes the scanned barcode, a false value does not decode it.

Example

See WaveLinkBarcode Samples

Default Method

This member of Class WaveLinkBarcode is supported on all OS types.

The Default Method returns the default decode state specified in the last barcode configuration file downloaded.

Syntax

```
public int Default()
```

Returns

The default decode state (see Remarks)

Remarks

The possible decode state values are:

BCDISABLED	- All barcodes not defined within the WaveLinkBarcode object are disabled and are not decoded for input
BCENABLED	- All barcodes not defined within the WaveLinkBarcode object are enabled and are decoded for input
NO_DEFAULT	- The default state for barcodes not defined within the WaveLinkBarcode remains unchanged

The default state for barcodes not defined within the WaveLinkBarcode object is set using the StoreBarcode Method. Therefore, you must be sure that the current WaveLinkBarcode object has been saved to a device before returning the default decode state.

DeleteBarcodeFile Method

This member of Class WaveLinkBarcode is supported on all OS types.

The DeleteBarcodeFile Method removes the specified barcode configuration from the device.

Syntax

```
public void DeleteBarcodeFile(String fileName)
            throws WaveLinkError,
                   IllegalArgumentException
```

Parameters

fileName The barcode configuration file to remove (see Remarks)

Throws

WaveLinkError

IllegalArgumentException

Remarks

You may pass a file name up to eight characters in length. Pass a wildcard symbol ("*") to delete all barcode files from the device.

Expand Method

This member of Class WaveLinkBarcode is supported on all OS types.

The Expand Method sets the expand flag value for the barcode configuration at the specified index.

Syntax

```
public boolean Expand(int bcIndex)

public void Expand(int bcIndex, boolean newExpand)
            throws IllegalArgumentException
```

Parameters

bcIndex	The zero-based index value for the target barcode configuration
newExpand	The boolean expand flag value (see Remarks)

Returns

The expand flag value as boolean

Throws

WaveLinkError

IllegalArgumentException

Remarks

For *newExpand*, a true value expands read symbology; a false value does not expand it.

GetBarcodeFile Method

This member of Class WaveLinkBarcode is supported on all OS types.

The GetBarcodeFile Method retrieves the specified barcode file from the device and sets the current object to that definition.

Syntax

```
public int GetBarcodeFile(String fileName)
            throws WaveLinkError,
                   IllegalArgumentException
```

Parameters

fileName The barcode file to be retrieved

Returns

The total number of barcode configurations

Throws

WaveLinkError

IllegalArgumentException

ListBarcodeFiles Method

This member of Class WaveLinkBarcode is supported on all OS types.

The ListBarcodeFiles Method retrieves a list of all barcode configuration files on the current device and stores it within the current object.

Syntax

```
public int ListBarcodeFiles()
            throws WaveLinkError
```

Returns

The number of barcode configuration files

Throws

WaveLinkError

Remarks

To return the value of the last successful ListBarcodeFiles Method without making an actual call to the device, use the BarcodeFileCount Method.

To list the total number of barcode configurations within a single barcode file use the BarcodeCount Method after making the file the current WaveLinkBarcode object.

NOTE On CE devices, the ListBarcodeFiles Method only lists files with a .bar file extension.

MaxLength Method

This member of Class WaveLinkBarcode is supported on all OS types.

The MaxLength Method sets the maximum barcode length for the barcode configuration at the specified index.

Syntax

```
public int MaxLength(int bcIndex)

public void MaxLength(int bcIndex, int newMaxLen)
            throws IllegalArgumentException
```

Parameters

bcIndex	The zero-based index value of the target barcode configuration
newMaxLen	The new maximum barcode length (see Remarks)

Returns

The maximum barcode length

Throws

IllegalArgumentException

Remarks

The first barcode file is indexed as zero. For example, to alter the maximum length of the third barcode file in the current WaveLinkBarcode object, you pass a 2.

To alter the maximum input length for a barcode configuration stored within a barcode file on a device, use GetBarcodeFile Method to set the barcode file as the current barcode object within the application, then use MaxLength.

Input length restrictions for barcode configurations can be eliminated by setting the minimum and maximum barcode length to zero (0).

MinLength Method

This member of Class WaveLinkBarcode is supported on all OS types.

The MinLength Method sets the minimum barcode length for the barcode configuration at the specified index.

Syntax

```
public int MinLength(int bcIndex)

public void MinLength(int bcIndex, int newMinLen)
         throws IllegalArgumentException
```

Parameters

bcIndex	The zero-based index value of the target barcode configuration (see Remarks)
newMinLen	The new minimum input length for the barcode configuration (see Remarks)

Returns

The minimum barcode length

Throws

IllegalArgumentException

Remarks

The first barcode file is indexed as zero. For example, to alter the maximum length of the third barcode file in the current WaveLinkBarcode object, you pass a 2.

To alter the minimum input length for a barcode configuration stored within a barcode file on a device, first use the GetBarcodeFile Method to set the barcode file as the current barcode object within the application, then use MinLength.

Input length restrictions for barcode configurations can be eliminated by setting the minimum and maximum barcode length to zero (0).

PullBarcode Method

This member of Class WaveLinkBarcode is supported on: DOS/embedded, Palm

The PullBarcode Method restores a barcode file as the current barcode configuration while removing the barcode file from the device.

Syntax

```
public void PullBarcode(String fileName)
            throws WaveLinkError,
                   IllegalArgumentException
```

Parameters

fileName	The barcode configuration file to be pulled

Throws

WaveLinkError

IllegalArgumentException

Remarks

Use the PushBarcode Method to place the current barcode configuration into a specified file for later restoration.

When used in conjunction with the PushBarcode Method, PullBarcode can temporarily change the settings for a device and return the device to its original settings for easy clean up.

Example

See the PushBarcode Method

PushBarcode Method

This member of Class WaveLinkBarcode is supported on: DOS/embedded, Palm

The PushBarcode Method places the current barcode configuration into the specified file for later restoration.

Syntax

```
public void PushBarcode(String fileName)
            throws WaveLinkError,
                   IllegalArgumentException
```

Parameters

fileName	The new barcode configuration file name

Throws

WaveLinkError

IllegalArgumentException

Remarks

Use the PullBarcode Method to restore a previously saved barcode file while removing the file from the device's memory.

When used in conjunction with the PullBarcode Method, PushBarcode can temporarily change the settings of a device and return the device to its original settings for easy clean up.

Example

```
WaveLinkBarcode barcodeIface = new WaveLinkBarcode();
    try {
        barcodeIface.PushBarcode("Defltbar");
        // Create, store, and use barcode configurations in
        // the app (not shown).
.
.
.
        // Use PullBarcode to restore the default barcode at
        // cleanup.
        BarcodeIface.PullBarcode("Defltbar");
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

RemoveBarcode Method

This member of Class WaveLinkBarcode is supported on all OS types.

The RemoveBarcode Method removes the specified barcode symbology from the current object.

Syntax

```
public void RemoveBarcode(int bcIndex)
```

Parameters

bcIndex	The zero-based index value of the target barcode configuration (see Remarks)

Remarks

The first barcode file is indexed as zero. For example, to remove the third barcode file in the current WaveLinkBarcode Object, you pass a 2.

To remove a barcode configuration from a stored barcode file on a device, set the barcode file as the current WaveLinkBarcode object using the GetBarcodeFile Method, then use RemoveBarcode.

StoreBarcode Method

This member of Class WaveLinkBarcode is supported on all OS types.

The StoreBarcode Method stores the specified barcode configuration to the current device.

Syntax

```
public void StoreBarcode(String fileName,
                         int defaultFlag)
            throws WaveLinkError,
                   IllegalArgumentException
```

Parameters

fileName	The barcode configuration file
defaultFlag	The default action to take with barcode definitions not specified (see Remarks)

Throws

WaveLinkError

IllegalArgumentException

Remarks

The possible *defaultFlag* values are:

BCDISABLED	- All barcodes not defined within the WaveLinkBarcode object is disabled and is not be decoded for input
BCENABLED	- All barcodes not defined within the WaveLinkBarcode object is enabled and is not decoded for input
NO_DEFAULT	- The default state for barcodes not defined within the WaveLinkBarcode remains unchanged

To use the current WaveLinkBarcode object with an RFInput call, you must first save the barcode object to the device using the StoreBarcode Method.

Example

```
WaveLinkBarcode barIface = new WaveLinkBarcode();
    Try {
        barIface.PushBarcode("Defltbar");
        barIface.ClearBarcodes();
        barIface.AddBarcode( WaveLinkBarcode.CODE_39,
                    false, true, 1, 32 );
        barIface.AddBarcode( WaveLinkBarcode.UPC_A,
                    false, true, 1, 32 );
        barIface.StoreBarcode("CycCtBar",
                 WaveLinkBarcode.BCENABLED);
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
    // You can use the CycCtBar barcode configuration
    // in an input call such as RFInput.
```

Symbology Method

This member of Class WaveLinkBarcode is supported on all OS types.

The Symbology Method sets the symbology type for the barcode configuration at the specified index.

Syntax

```
public int Symbology(int bcIndex)

public void Symbology(int bcIndex, int newSymbology)
            throws IllegalArgumentException
```

Parameters

bcIndex	The zero-based index value for the target barcode configuration (see Remarks)
newSymbology	The new symbology type (see Remarks)

Throws

WaveLinkError

IllegalArgumentException

Remarks

The first barcode file is indexed as zero. For example, to return the barcode type of the third barcode file in the current WaveLinkBarcode Object, you pass a 2.

The possible symbology types are:

B_UPC

CODABAR

CODE_11

CODE_39

CODE_93

CODE_128

CODE_D25

CODE_I25

D25_IATA

EAN_8

EAN_13

MSI
PDF_417
TO_39
UCC_128
UPC_A
UPC_E0
UPC_E1
WLNOSYMBOLOGY

WaveLinkBarcode Samples

The following sample code demonstrates the implementation and capabilities of the Class WaveLinkBarcode and its associated methods.

Java Sample

```
/*************************************************
/*************************************************
* This source code is for demonstration
* purposes only and should be treated as such.
*
*/package barcodedemo;
import com.wavelink.clientui.*;
public class JavaBarcodeDemo extends WaveLinkApplication {
    private static int CLR = 27;
    public void WaveLinkMain ( String [] args) {
        WaveLinkIO ioIface = new WaveLinkIO ();
        WaveLinkBarcode barcodeIface = new WaveLinkBarcode();
        String returnData;
        // holds the data returned from an RFInput
        int lastExtendedType; // set by RFInput
        try {
            // add two barcodes with
            // type, expand, decode, min, max
            barcodeIface.AddBarcode( WaveLinkBarcode.CODE_39,
                        false, true, 1, 32 );
            barcodeIface.AddBarcode( WaveLinkBarcode.UPC_A,
```

```
                false, true, 1, 32 );
for ( ;; ) {
    ioIface.RFPrint( 0, 0, "Scan CODE_39 ",
         WaveLinkIO.WLNORMAL | WaveLinkIO.WLCLEAR );
    ioIface.RFPrint( 0, 7, " CLR to Exit Demo ",
            WaveLinkIO.WLREVERSE
             | WaveLinkIO.WLFLUSHOUTPUT );
    // enable CODE_39
    barcodeIface.Decode( 0, true );
    // disable UPC_A
    barcodeIface.Decode( 1, false );
    // save config to handheld and disable all
    // other barcodes
    barcodeIface.StoreBarcode( "BCDemo",
                  WaveLinkBarcode.BCDISABLED ); ;
    returnData = ioIface.RFInput( "", 20, 0, 3,
                 "BCDemo",WaveLinkIO.WLNORMALKEYS,
                  WaveLinkIO.WLBACKLIGHT );
    // check for the CLR ( escape ) key
    if ( ioIface.LastInputType() ==
         WaveLinkIO.WLCOMMANDTYPE &&
         returnData.charAt(0) == (char)CLR )
        return;
    lastExtendedType = ioIface.LastExtendedType();
   if ( lastExtendedType != WaveLinkBarcode.CODE_39)
   {
        WaveLinkMessageBox wlmb = new
                                WaveLinkMessageBox( );
        wlmb.SetMessageLine( "Scan CODE_39", 0 );
        wlmb.Display( 5 );
        continue;
    }
    ioIface.RFPrint( 0, 0, "Scan UPC_A ",
         WaveLinkIO.WLNORMAL | WaveLinkIO.WLCLEAR );
    // disable CODE_39
    barcodeIface.Decode( 0, false );
    // enable UPC_A
    barcodeIface.Decode( 1, true );
    // save config to handheld and disable all
    // other barcodes
    barcodeIface.StoreBarcode( "BCDemo",
                  WaveLinkBarcode.BCDISABLED );
    returnData = ioIface.RFInput( "", 20, 0, 3,
              "BCDemo",
```

```
                                        WaveLinkIO.WLNORMALKEYS,
                                         WaveLinkIO.WLBACKLIGHT );
                        // check for the CLR ( escape ) key
                        if ( ioIface.LastInputType() ==
                             WaveLinkIO.WLCOMMANDTYPE &&
                             returnData.charAt(0) == (char)CLR )
                            return;
                         lastExtendedType = ioIface.LastExtendedType();
                       if ( lastExtendedType != WaveLinkBarcode.UPC_A )
                       {
                            WaveLinkMessageBox wlmb = new
                                                  WaveLinkMessageBox( );
                            wlmb.SetMessageLine( "Scan UPC_A", 0 );
                            wlmb.Display( 5 );
                            continue;
                        }
                    }
                } catch ( WaveLinkError wlErr ) {
                    // Do error handling
                }
            }
        }
```

Class WaveLinkConstants

```
com.wavelink.clientui
```

The public class WaveLinkConstants is supported on: CE

WaveLinkConstants provides special values to control the display of widgets on devices and the input received from Field widgets. Used with the WidgetOptions setFlags()/addFlags() Methods, these can be combined to only allow certain input from a user.

Constructor

```
public WaveLinkConstants()
```

Fields

AUTOSIZE	FIELD_MULTILINE
FIELD_ALPHA	FIELD_NOECHO
FIELD_ASCII	FIELD_NUMERIC
FIELD_DISABLEKEY	FIELD_PASSWORD
FIELD_DISABLESCAN	FIELD_UPPER
FIELD_FORCEENTRY	FIELD_WANTRETURN
FIELD_LOWER	RADIO_ALIGNVERT

Throws

Remarks

Field Detail

AUTOSIZE

```
public static final int AUTOSIZE
```

Sizes a field (or other widgets) according to the size of the text that it contains.

FIELD_ALPHA

```
public static final long FIELD_ALPHA
```

Only allows alphabetic characters to be entered.

FIELD_ASCII

```
public static final long FIELD_ASCII
```

Only allows characters from the lower ASCII range to be entered.

FIELD_DISABLEKEY

```
public static final long FIELD_DISABLEKEY
```

Disables normal key entry in a field, though scans are still available.

FIELD_DISABLESCAN

```
public static final long FIELD_DISABLESCAN
```

Disables scanning in a field.

FIELD_FORCEENTRY

```
public static final long FIELD_FORCEENTRY
```

Forces a field to have data in it before the user can complete entry.

FIELD_LOWER

```
public static final long FIELD_LOWER
```

Forces lowercase characters in a field.

FIELD_MULTILINE

```
public static final long FIELD_MULTILINE
```

Allows multiple lines to be entered in a field.

FIELD_NOECHO

```
public static final long FIELD_NOECHO
```

Text entered is not shown in the field.

FIELD_NUMERIC

```
public static final long FIELD_NUMERIC
```

Only allows numbers to be entered in a field.

FIELD_PASSWORD

```
public static final long FIELD_PASSWORD
```

Shows asterisks instead of normal characters in a field.

FIELD_UPPER

```
public static final long FIELD_UPPER
```

Forces all text to be uppercase in a field.

FIELD_WANTRETURN

```
public static final long FIELD_WANTRETURN
```

Field entry will allow Carriage Returns (CRLF) instead of ending.

RADIO_ALIGNVERT

```
public static final long RADIO_ALIGNVERT
```

Aligns a radio button list vertically instead of horizontally (default).

Class WaveLinkConstants.CheckboxState

`com.wavlink.clientui`

The public class **WaveLinkConstants.CheckboxState** is supported on: CE

This class is a collection of constants used for configuring a checkbox widget and determining its state. Use when creating a checkbox widget using the WaveLinkScreen.createCheckbox() Method, and use when evaluating an event from a checkbox widget to determine what state the checkbox is currently in.

Fields

CHECKED UNDETERMINED
UNCHECKED

Field Detail

CHECKED

```
public static final int CHECKED
```

When creating a checkbox, this flag will force it to start initially checked. When receiving an event from a checkbox widget, this means it is now checked.

UNCHECKED

```
public static final int UNCHECKED
```

When creating a checkbox, this flag will force it to start initially unchecked. When receiving an event from a checkbox widget, this means it is now unchecked.

UNDETERMINED

```
public static final int UNDETERMINED
```

When creating a checkbox, this will force it to start initially in a checked gray state. When receiving an event from a checkbox widget, this means it is in the checked gray state.

Class WaveLinkConstants.Coord

```
com.wavelink.clientui
```

The public class **WaveLinkConstants.Coord** is supported on: CE

Used to configure a WaveLinkScreen coordinate system, by using setCoordinateType(). Also used when calling WaveLinkTerminal.TerminalHeight and TerminalWidth to find out exact pixel or cell amounts available. For a WaveLinkScreen, PIXEL is the default coordinate type.

Fields

CELL PIXEL
PERCENTAGE

Field Detail

CELL

```
public static final int CELL
```

This signifies that the screen is broken up into cells, when used to configure a WaveLinkScreen. When calling TerminalWidth or TerminalHeight from the WaveLinkTerminal class, this will return the width or height in cells.

PERCENTAGE

Allows widgets to be placed at certain percentage values from the edge of the screen rather than at absolute cells or pixels. Not valid for WaveLinkTerminal calls.

PIXEL

```
public static final int PIXEL
```

This signifies that the screen is broken up into pixels, when used to configure a WaveLinkScreen. When calling TerminalWidth or TerminalHeight from the WaveLinkTerminal class, this will return the width or height in pixels.

Class WaveLinkConstants.Justify

`com.wavelink.clientui`

The public class **WaveLinkConstants.Justify** is supported on: CE

Constants used for configuring text alignment on widgets. Use with the WidgetOptions setJustification() Method.

Fields

CENTER	NONE
LEFT	RIGHT

Field Detail

CENTER

```
public static final int CENTER
```

Specifies center justification.

LEFT

```
public static final int LEFT
```

Specifies left justification.

NONE

```
public static final int NONE
```

No justification specified.

RIGHT

```
public static final int RIGHT
```

Specifies right justification.

Class WaveLinkConstants.Repeater

`com.wavelink.clientui`

The Class WaveLinkConstants.Repeater is supported on: CE

This class has constants for use with the WaveLinkScreen **createRepeaterButton() Method**. A repeater button is an arrow pointing in the direction of the constant used, but functions like a button otherwise.

Fields

DOWN	RIGHT
LEFT	UP

Field Detail

DOWN

```
public static final int DOWN
```

Creates a repeater button pointing down.

LEFT

```
public static final int LEFT
```

Creates a repeater button pointing left.

RIGHT

```
public static final int RIGHT
```

Creates a repeater button pointing right.

UP

```
public static final int UP
```

Creates a repeater button pointing up.

Class WaveLinkConstants.State

`com.wavelink.clientui`

The public Class WaveLinkConstants.State is supported on: CE

The constants contained in this class are used for setting the state of widgets, using the WidgetHandle setInitialState() method.

Fields

DISABLED	HIDDEN_DISABLED
HIDDEN	STANDARD

Field Detail

DISABLED

```
public static final int DISABLED
```

The widget will be disabled on the screen.

HIDDEN

```
public static final int HIDDEN
```

The widget will be hidden on the screen.

HIDDEN_DISABLED

```
public static final int HIDDEN_DISABLED
```

The widget will be both hidden and disabled.

STANDARD

```
public static final int STANDARD
```

The widget will be shown as normal.

Class WaveLinkError

`com.wavelink.clientui`

The public class **WaveLinkError** is supported on all OS types.

The Class WaveLinkError notifies the user that an error has occurred and provides information about the specific error.

Fields

WLBARCODELIMITEXCEEDED	WLMEMORYERROR
WLBCADDERROR	WLNOERROR
WLBCCLEARERROR	WLNOINPUTTYPE
WLBCPARSEFAILURE	WLNOTINITIALIZED
WLCOMMERROR	WLPORTTIMEOUT
WLDTOUTOFSYNC	WLREQUESTFAIL
WLERROR	WLTIMEOUT
WLFUNCTIONFAILED	WLTONEADDERROR
WLINVALIDPARAMS	WLTONECLEARERROR
WLINVALIDRETURN	WLUNKNOWNERROR
WLIOQUEUEERROR	

Constructors

```
public WaveLinkError()
```

```
public WaveLinkError(String arg0)
```

```
public WaveLinkError(String arg0, int errorNum)
```

Methods

WLErrorCode

Field Detail

WLBARCODELIMITEXCEEDED

```
public static final int WLBARCODELIMITEXCEEDED
```

Exceeded number of allowable barcode configurations.

WLBCADDERROR

```
public static final int WLBCADDERROR
```

Failure adding barcode configuration.

WLBCCLEARERROR

```
public static final int WLBCCLEARERROR
```

Failure clearing barcode configuration list.

WLBCPARSEFAILURE

```
public static final int WLBCPARSEFAILURE
```

Failure parsing barcode file.

WLCOMMERROR

```
public static final int WLCOMMERROR
```

Communication error.

WLERROR

```
public static final int WLERROR
```

Error occurred.

WLFUNCTIONFAILED

```
public static final int WLFUNCTIONFAILED
```

Function call failed at device.

WLINVALIDPARAMS

```
public static final int WLINVALIDPARAMS
```

Invalid function parameters.

WLINVALIDRETURN

```
public static final int WLINVALIDRETURN
```

Invalid function return packet.

WLIOQUEUEERROR

```
public static final int WLIOQUEUEERROR
```

IO queue buffer error.

WLMEMORYERROR

```
public static final int WLMEMORYERROR
```

Memory allocation error.

WLNOERROR

```
public static final int WLNOERROR
```

No error.

WLNOIPUTTYPE

```
public static final int WLNOIPUTTYPE
```

Unable to determine input type.

WLNOTINITIALIZED

```
public static final int WLNOTINITIALIZED
```

Connection not initialized.

WLPORTTIMEOUT

```
public static final int WLPORTTIMEOUT
```

Aux port timeout.

WLREQUESTFAIL

```
public static final int WLREQUESTFAIL
```

Input request error.

WLDTOUTOFSYNC

```
public static final int WLDTOUTOFSYNC
```

Terminal Date/Time is out of sync by more that 5 seconds.

WLTIMEOUT

```
public static final int WLTIMEOUT
```

RFInput timeout has occurred.

WLTONEADDERROR

```
public static final int WLTONEADDERROR
```

Failure adding tone configuration.

WLTONECLEARERROR

```
public static final int WLTONECLEARERROR
```

Failure clearing tone configuration list.

WLUNKNOWNERROR

```
public static final int WLUNKNOWNERROR
```

Unknown error occurred.

Method Summary

WLErrorCode Method

- Returns the value of the error.

WLErrorCode Method

This member of Class WaveLinkError is supported on all OS types.

The WLErrorCode Method returns the error value.

Syntax

```
public int WLErrorCode()
```

Returns

The error code value

Remarks

See the Constant Values appendix for the numeric equates returned by the function.

Class WaveLinkEvent

`com.wavelink.clientui`

The public class WaveLinkEvent is supported on:

When receiving an event in a WaveLinkListener class, the parameter passed to the widgetEvent Method will be an object of type WaveLinkEvent. This object has several methods that will help you determine how to handle the event.

Methods

getBarcodeType	getHandle
getEvent	getInputType
getExtendedType	isBarcode

Method Summary

getBarcodeType

- Returns the barcode type.

getEvent

- Returns the event data.

getExtendedType

- Returns the extended type.

getHandle

- Returns the handle of the widget that triggered the event. Depending on the type of widget, it may be needed to cast as a more specific widget type than WidgetHandle.

getInputType

- Returns the input type.

isBarcode

- Returns if a barcode was scanned.

Example

```
public class myListener implements WaveLinkListener {
  public void widgetEvent(WaveLinkEvent e) {
    System.out.println(e.getEvent());
```

For another example, see WaveLinkScreen Samples.

getBarcodeType Method

This member of Class WaveLinkEvent is supported on: CE

The getBarcodeType Method returns the barcode type.

Syntax

```
public int getBarcodeType()
```

Returns

The barcode type that was scanned (See Class WaveLinkBarcode for constant values)

getEvent Method

This member of Class WaveLinkEvent is supported on: CE

The getEvent Method returns the event data.

Syntax

```
public java.lang.String getEvent()
```

Returns

Returns the text from the widget or keystroke that generated the event.

getExtendedType Method

This member of Class WaveLinkEvent is supported on: CE

The getExtendedType Method returns the extended type.

Syntax

```
public int getExtendedType()
```

Returns

The extended type of an event (See WaveLinkIO.LastExtendedType for possible types)

getHandle Method

This member of Class WaveLinkEvent is supported on: CE

The getHandle Method returns the handle of the widget that triggered the event. Depending on the type of widget, it may be needed to cast as a more specific widget type than WidgetHandle, such as a ButtonHandle or ListboxHandle.

Syntax

```
public ObjectHandle getHandle()
```

Returns

The handle to the widget that triggered the event.

getInputType Method

This member of Class WaveLinkEvent is supported on: CE

The getInputType Method returns the input type.

Syntax

```
public int getInputType()
```

Returns

The input type of the event that was triggered (See WaveLinkIO.LastInput-Type for possible types)

isBarcode Method

This member of Class WaveLinkEvent is supported on: CE

The isBarcode Method returns if a barcode was scanned.

Syntax

```
public boolean isBarcode()
```

Returns

True if a barcode was scanned into the widget, false if not

Class WaveLinkFactory

`com.wavelink.clientui`

The public class **WaveLinkFactory** is supported on: Palm, CE

The WaveLinkFactory object contains the methods necessary to build all types of widget objects.

NOTE Deprecated.

Constructors

```
public WaveLinkFactory()

public WaveLinkFactory(WaveLinkIO ioIface)

public WaveLinkFactory(IWaveLinkSession currentSession)
```

Properties

DefaultCoordinateType
DefaultBackColor
DefaultDisplayFlags
DefaultFontSize
DefaultFontType
DefaultForeColor

Methods

CreateBitmap
CreateButton
CreateCheckbox
CreateField
CreateHotspot
CreateLabel
CreateMenubar
CreatePopupTrigger
CreatePushButton
CreateRepeaterButton
CreateSelectorTrigger

Remarks

The WaveLinkFactory object automates the process of creating widgets. Use the WaveLinkWidget object to specify widget characteristics not available in the WaveLinkFactory object.

To display widgets to the device, you must use the StoreWidgets Method.

To return input from a widget, use the RFInput Method or GetEvent Method. These two methods automatically return the widget ID.

To process input from the widget based on the widget ID, use the LastExtendedType Method.

To process input from the widget based on its general input type (scanned, keyed, widget input) use the LastInputType Method.

Property Summary

DefaultCoordinateType Property

- Stores and retrieves the default coordinate type of the WaveLinkFactory object.

DefaultBackColor Property

- Stores and retrieves the default background color of the WaveLinkFactory object.

DefaultDisplayFlags Property

- Stores and retrieves the default label formatting flags of the WaveLinkFactory object.

DefaultFontSize Property

- Stores and retrieves the size of the default display font of the WaveLinkFactory object.

DefaultFontType Property

- Stores and retrieves the name of the default display font of the WaveLinkFactory object.

DefaultForeColor Property

- Stores and retrieves the default foreground color of the WaveLinkFactory object.

Method Summary

CreateBitmap Method

- Creates an image based on a bitmap file.

CreateButton Method

- Creates a standard button that triggers an action when the user clicks it.

CreateCheckbox Method

- Creates a checkbox that stores an on/off state.

CreateField Method

- Creates an input box.

CreateHotspot Method

- Creates an invisible area that triggers a specific action when the user taps it.

CreateLabel Method

- Creates positioned text in a proportionally-spaced font.

CreateMenubar Method

- Creates a menubar containing a set of menus.

CreatePopupTrigger Method

- Creates a popup trigger that displays a pop-up list when the user taps it.

CreatePushButton Method

- Creates a set of buttons that store an on/off state.

CreateRepeaterButton Method

- Creates a button that implies that something can be scrolled or incremented when the user clicks it.

CreateSelectorTrigger Method

- Creates a button that implies that a dialog box opens when the user clicks it.

DefaultCoordinateType Property

This member of Class WaveLinkFactory is supported on: Palm, CE

The DefaultCoordinateType Property sets and retrieves the default coordinate type for the WaveLinkFactory object.

Syntax

```
public int getDefaultCoordinateType()

public void setDefaultCoordinateType(int newCoordType)
```

Parameters

newCoordType	The variable that specifies the default coordinate type (see Remarks)

Returns

The default coordinate type (see Remarks)

Remarks

The possible values for *newCoordType* are:

BYCELL	- Position the widget using cell coordinates (rows and columns)
BYPIXEL	- Position the widget using pixel coordinates
BYPERCENTAGE	- Position the widget using percentage coordinates

The DefaultCoordinateType Property determines how the values entered for the positioning and size of the widget are interpreted. BYCELL is the default value.

Example

See the CreateBitmap Method

DefaultBackColor Property

This member of Class WaveLinkFactory is supported on: CE

The DefaultDisplayBackcolor Property sets and retrieves the default background color of the WaveLinkFactory object.

Syntax

```
public long getDefaultBackColor()

public void setDefaultBackColor(long newBackColor)
```

Parameters

newBackColor The variable that stores the default background color

Returns

The default background color

Remarks

The background color must be specified in six-byte RGB format (example, FFFFFF).

Pass a value of -1 to use the default system background color.

DefaultDisplayFlags Property

This member of Class WaveLinkFactory is supported on: Palm, CE

The DefaultDisplayFlags Property sets and retrieves the default label formatting flags of the WaveLinkFactory object.

Syntax

```
public long getDefaultDisplayFlags()

public void setDefaultDisplayFlags(long newDisplayFlags)
```

Parameters

newDisplayFlags	The variable that specifies the default label formatting flags (see Remarks)

Returns

The default label formatting flags (see Remarks)

Remarks

Valid settings for *newDisplayFlags* include:

CENTERJUST	- Center-justified label text
LEFTJUST	- Left-justified label text
RIGHTJUST	- Right-justified label text

DefaultFontSize Property

This member of Class WaveLinkFactory is supported on: CE

The DefaultFontSize Property sets and retrieves the default font size of the WaveLinkFactory object.

Syntax

```
public int getDefaultFontSize()
```

```
public void setDefaultFontSize(int newFontSize)
```

Parameters

newFontSize Specifies the default font size

Returns

The default font size

Remarks

The font size must be specified in points. See Client documentation for more information about specifying font size.

Pass a value of -1 to use the default system font size.

DefaultFontType Property

This member of Class WaveLinkFactory is supported on: CE

The DefaultFontType Property sets and retrieves the default font of the WaveLinkFactory object.

Syntax

```
public String getDefaultFontType()
```

```
public void setDefaultFontType(String newFontType)
```

Parameters

newFontType The variable that specifies the default font (see Remarks)

Returns

The default display font

Remarks

Pass value of null to use the system default font.

For details on specifying font name for CE devices, see client documentation.

The possible values for *newFontType* for Palm devices:

BOLD	- Bold font
FIXED	- Monospace font
LARGE	- Large font
LARGEBOLD	- Large bold font
STANDARD	- Standard Palm OS font

DefaultForeColor Property

This member of Class WaveLinkFactory is supported on: CE

The DefaultForeColor Property sets and retrieves the default foreground color of the WaveLinkFactory object.

Syntax

```
public long getDefaultForeColor()
```

```
public void setDefaultForeColor(long newForeColor)
```

Parameters

newForeColor The variable that specifies the default foreground color

Returns

The default foreground color

Remarks

Pass a value of -1 to use default system foreground color.

The color must be specified in six-byte RGB format (example, FFFFFF).

CreateBitmap Method

This member of Class WaveLinkFactory is supported on: Palm, CE

The CreateBitmap Method creates a image based on a bitmap file.

Syntax

```
public WaveLinkWidget CreateBitmap(int XCoord, int YCoord,
                                   int Width, int Height,
                                   String bitmapFile,
                                   WaveLinkWidgetCollection
                                   targetCollection)
                    throws WaveLinkError
```

Parameters

XCoord	The starting left coordinate of the widget (see Remarks)
YCoord	The starting top coordinate of the widget (see Remarks)
Width	The horizontal extent of the widget (see Remarks)
Height	The vertical extent of the widget (see Remarks)
bitmapFile	The file name of the bitmap image (must include the full path)
targetCollection	The name of the collection that will contain the widget

Returns

A pointer to the bitmap widget

Throws

WaveLinkError

Remarks

The DefaultCoordinateType Property determines how the application interprets the values entered for the position, width, and height of the widget.

For the *Width* and *Height* parameters, pass a value of 0 to set the width or height automatically. Set the size of the widget to AUTOSIZE by setting both the width and height to zero (0).

The CreateBitmap Method automatically stores the path for the bitmap image in the SpecialString Property.

Example

```
WaveLinkFactory factoryIface = new WaveLinkFactory();
WaveLinkWidgetCollection colIface = new
```

```
    WaveLinkWidgetCollection();
WaveLinkIO ioIface = new WaveLinkIO();
WaveLinkFile fileIface = new WaveLinkFile();
    try {
        // The following code illustrates an easy way to
        // transfer the image file.
        fileIface.RFTransferFile("C:\temp\logo.bmp",
                  "logo.bmp", true);
       // You may choose to clear the screen before creating
        // the bitmap.
        ioIface.RFPrint(0, 35, "", WaveLinkIO.WLCLEAR);
        factoryIface.setDefaultCoordinateType
                    (WaveLinkWidget.BYPIXEL);
        factoryIface.CreateBitmap(20, 20, 35, 25,
                                  "logo.bmp",
                                  colIface);
        colIface.StoreWidgets();
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

CreateButton Method

This member of Class WaveLinkFactory is supported on: Palm, CE

The CreateButton Method creates a standard labeled button that triggers a specific action when the user "clicks" it.

Syntax

```
public WaveLinkWidget CreateButton(int XCoord, int YCoord,
                                   int Width, int Height,
                                   String displayText,
                                   WaveLinkWidgetCollection
                                   targetCollection)
                    throws WaveLinkError
```

Parameters

XCoord	The starting left coordinate of the widget (see Remarks)
YCoord	The starting top coordinate of the widget (see Remarks)
Width	The horizontal extent of the widget (see Remarks)
Height	The vertical extent of the widget (see Remarks)
displayText	The display text of the widget
targetCollection	The name of the widget collection

Returns

A pointer to the button widget

Throws

WaveLinkError

Remarks

The default value returned by a button widget through an RFInput Method or GetEvent Method call is the display text.

The DefaultCoordinateType Property determines how the application interprets the values entered for the position, width, and height of the widget.

For the *Width* and *Height* parameters, pass a value of 0 to set the width or height automatically. Set the size of the widget to AUTOSIZE by setting both the width and height to zero (0). AUTOSIZE sizes the button widget according to the size of the text that it contains.

Example

```
WaveLinkFactory factoryIface = new WaveLinkFactory();
WaveLinkWidgetCollection colIface = new
    WaveLinkWidgetCollection();
WaveLinkIO ioIface = new WaveLinkIO();
    try {
        ioIface.RFPrint(0, 0, "WaveLink Corporation",
                WaveLinkIO.WLCLEAR | WaveLinkIO.WLREVERSE);
        ioIface.RFPrint(0, 1, "   Auto Detailing   ",
                 WaveLinkIO.WLNORMAL);
        factoryIface.CreateButton(4, 12, 0, 0, " Start ",
                     colIface);
        colIface.StoreWidgets();
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

CreateCheckbox Method

This member of Class WaveLinkFactory is supported on: Palm, CE

The CreateCheckbox Method creates a widget that stores an on/off state. The on/off state switches when the user taps it.

Syntax

```
public WaveLinkWidget CreateCheckbox(int XCoord, int
                                     YCoord,
                                     int Width, int Height,
                                     String displayText,
                                     int initialState,
                                     WaveLinkWidgetCollection
                                     targetCollection)
                        throws WaveLinkError
```

Parameters

XCoord	The starting left coordinate of the widget (see Remarks)
YCoord	The starting top coordinate of the widget (see Remarks)
Width	The horizontal extent of the widget (see Remarks)
Height	The vertical extent of the widget (see Remarks)
displayText	The display text of the widget
initialState	The initial state of the checkbox (see Remarks)
targetCollection	The name of the collection that will contain the widget

Returns

A pointer to the checkbox widget

Throws

WaveLinkError

Remarks

The default value returned by a checkbox widget through an RFInput or GetEvent call is the checkbox state. The possible values for the checkbox state are:

0 - UNCHECKED	- This constant represents an unchecked checkbox state
1 - CHECKED	- This constant represents a checked checkbox state

2 - UNDETERMINED - This constant represents an undetermined checkbox state

The DefaultCoordinateType Property determines how the application interprets the values entered for the position, width, and height of the widget.

For the *Width* and *Height* parameters, pass a value of 0 to set the width or height automatically. Set the size of the widget to AUTOSIZE by setting both the width and height to zero (0).

We recommend that you useAUTOSIZE setting for the height and width of the checkbox. Using a value other than AUTOSIZE does change the size of the widget, only the extent of the clickable area surrounding it.

Example

```
WaveLinkFactory factoryIface = new WaveLinkFactory();
WaveLinkWidgetCollection colIface = new
    WaveLinkWidgetCollection();
WaveLinkIO ioIface = new WaveLinkIO();
    try {
        // Give the widget some context
        ioIface.RFPrint(0, 0, "WaveLink Corporation",
                WaveLinkIO.WLCLEAR
                        | WaveLinkIO.WLREVERSE);
        ioIface.RFPrint(0, 1, "   Auto Detailing   ",
                WaveLinkIO.WLNORMAL);
        ioIface.RFPrint(0, 3, "   Select Vehicle   ",
                WaveLinkIO.WLNORMAL
                 | WaveLinkIO.WLFLUSHOUTPUT);
        factoryIface.CreateCheckbox(0, 5, 0, 0,
                 "Passenger Auto", WaveLinkWidget.UNCHECKED
                 colIface);
        colIface.StoreWidgets();
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

CreateField Method

This member of Class WaveLinkFactory is supported on: Palm, CE

The CreateField Method creates an input box.

Syntax

```
public WaveLinkWidget CreateField(int XCoord, int YCoord,
                                  int Width, int Height,
                                  String fieldText,
                                  WaveLinkWidgetCollection
                                  targetCollection)
                        throws WaveLinkError
```

Parameters

XCoord	The starting left coordinate of the widget (see Remarks)
YCoord	The starting top coordinate of the widget (see Remarks)
Width	The horizontal extent of the widget (see Remarks)
Height	The vertical extent of the widget (see Remarks)
fieldText	The default text contained in the field
targetCollection	The name of collection that will contain the widget

Returns

A pointer to the field widget

Throws

WaveLinkError

Remarks

The DefaultCoordinateType Property determines how the application interprets the values entered for the position, width, and height of the widget.

For the *Width* and *Height* parameters, pass a value of 0 to set the width or height automatically. Set the size of the widget to AUTOSIZE by setting both the width and height to zero (0).

Each returned field is a separate input event. To return additional events requires subsequent calls to either RFInput Method or GetEvent Method in the same WaveLinkIO object. Each subsequent callto RFInput or GetEvent removes one additional event from the input stack.

Example

```
WaveLinkFactory factoryIface = new WaveLinkFactory();
WaveLinkWidgetCollection colIface = new
    WaveLinkWidgetCollection();
WaveLinkIO ioIface = new WaveLinkIO();
    try {
        ioIface.RFPrint(0, 0, "Enter your name:",
                WaveLinkIO.WLCLEAR);
        factoryIface.CreateField(3, 3, 12, 1, "your name",
                    colIface);
        colIface.StoreWidgets();
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

CreateHotspot Method

This member of Class WaveLinkFactory is supported on: Palm, CE

The CreateHotspot Method creates an invisible area that triggers a specific action when the user taps it.

Syntax

```
public WaveLinkWidget CreateHotspot(int XCoord, int
                                    YCoord,
                                    int Width, int Height,
                                    WaveLinkWidgetCollection
                                    targetCollection)
                        throws WaveLinkError
```

Parameters

XCoord	The starting left coordinate of the widget (see Remarks)
YCoord	The starting top coordinate of the widget (see Remarks)
Width	The horizontal extent of the widget (This value must be greater than zero)
Height	The vertical extent of the widget (This value must be greater than zero)
targetCollection	The name of the collection that will contain the widget

Returns

A pointer to the hotspot widget

Throws

WaveLinkError

Remarks

The DefaultCoordinateType Property determines how the application interprets the values entered for the position, width, and height of the widget.

For the *Width* and *Height* parameters, the value must be greater than zero (0).

Example

```
WaveLinkFactory factoryIface = new WaveLinkFactory();
WaveLinkWidgetCollection colIface = new
```

```
    WaveLinkWidgetCollection();
WaveLinkIO ioIface = new WaveLinkIO();
    try {
        // In this example, clear the screen and hide
        // the cursor.
        ioIface.RFPrint(0, 35, "", WaveLinkIO.WLCLEAR);
        factoryIface.setDefaultCoordinateType
                    (WaveLinkWidget.BYPIXEL);
        // Give some context for the hotspot.
      factoryIface.CreateBitmap(20, 20, 35, 25, "logo.bmp",
                   colIface);
        // Create the widget.
        factoryIface.CreateHotspot(20, 20, 35, 25,
                    "your name", colIface);
        colIface.StoreWidgets();
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

CreateLabel Method

This member of Class WaveLinkFactory is supported on: Palm, CE

The CreateLabel Method creates positioned text that is in a porportionally spaced font and has the same "lifetime" as the other widgets.

Syntax

```
public WaveLinkWidget CreateLabel(int XCoord, int YCoord,
                                  int Width, int Height,
                                  String displayText,
                                  WaveLinkWidgetCollection
                                  targetCollection)
                        throws WaveLinkError
```

Parameters

XCoord	The starting left coordinate of the widget (see Remarks)
YCoord	The starting top coordinate of the widget (see Remarks)
Width	The horizontal extent of the widget (see Remarks)
Height	The vertical extent of the widget (see Remarks)
displayText	The display text of the widget
targetCollection	The name of the collection that will contain the widget

Returns

A pointer to the label widget

Throws

WaveLinkError

Remarks

The DefaultCoordinateType Property determines how the application interprets the values entered for the position, width, and height of the widget.

For the *Width* and *Height* parameters, pass a value of 0 to set the width or height automatically. Set the size of the widget to AUTOSIZE by setting both the width and height to zero (0).

Example

```
    WaveLinkFactory factoryIface = new WaveLinkFactory();
    WaveLinkWidgetCollection colIface = new
        WaveLinkWidgetCollection();
```

.

```
.
.
        try {
            factoryIface.CreateLabel(10, 10, 0, 0,
                        "Version 1.5", colIface);
            colIface.StoreWidgets();
        }
        catch (WaveLinkError wlErr) {
            //Do error handling
        }
```

CreateMenubar Method

This member of Class WaveLinkFactory is supported on: Palm, CE

The CreateMenubar Method creates a menubar containing a set of menus. Each menu is associated with a pop-up list that activates when the user taps it.

Syntax

```
public WaveLinkWidget CreateMenubar(WaveLinkMenubarInfo
                                    menuInfo,
                                    WaveLinkWidgetCollection
                                    targetCollection)
                        throws WaveLinkError
```

Parameters

menuInfo The name of the WaveLinkMenubarInfo object to associate with the widget

targetCollection The name of the collection that will contain the widget

Returns

A pointer to the menubar widget

Throws

WaveLinkError

Remarks

The menubar displays menu names based on stored WaveLinkMenu configurations. The menu configuration used by the menubar must be stored on the device before the menubar can access it. The menubar widget returns the menu index of the user-selected option.

Use the WaveLinkMenubarInfo object to create a list of menus for the menubar widget.

While creating the widget object, the CreateMenubar Method automatically calls the SetSpecialInfo Method which assigns the formatted list of menus to the SpecialString Property. If you want to modify the list of menus for the current menubar widget, use the SpecialString Property.

Example

```
WaveLinkFactory factoryIface = new WaveLinkFactory();
WaveLinkWidgetCollection colIface = new
    WaveLinkWidgetCollection();
```

```
WaveLinkMenubarInfo infoIface = new WaveLinkMenubarInfo();
WaveLinkMenu mnuIface = new WaveLinkMenu();
    try {
    // Create the menus that appear in the menubar.
    mnuIface.ResetMenu();
    mnuIface.AddTitleLine("Widget Demo's");
    mnuIface.AddOption("Buttons");
    mnuIface.AddOption("Pen Capture");
    mnuIface.StoreMenu("MenOne");
    mnuIface.ResetMenu();
    mnuIface.AddTitleLine("Exit");
    mnuIface.AddOption("Quit");
    mnuIface.StoreMenu("MenTwo");
    // Insert the menus into the WaveLinkMenubarInfo object,
    // using a -1 to automatically tack the
    // menu to the end of the list.
    infoIface.Insert(-1, "MenOne");
    infoIface.Insert(-1, "MenTwo");
.
.
.
        factoryIface.CreateMenubar(myMenInfo,
                    colIface);
        colIface.StoreWidgets();
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

CreatePopupTrigger Method

This member of Class WaveLinkFactory is supported on: Palm, CE

The CreatePopupTrigger Method creates a popup trigger that displays a pop-up list when the user taps it. The currently selected item appears to the right of the trigger.

Syntax

```
public WaveLinkWidget CreatePopupTrigger(int XCoord,
                                         int YCoord,
                                         intWidth, int Height,
                                         StringmenuName,
                                         int initialOption,
                                         WaveLinkWidgetCollection
                                         targetCollection)
                      throws WaveLinkError
```

Parameters

XCoord	The starting left coordinate of the widget (see Remarks)
YCoord	The starting top coordinate of the widget (see Remarks)
Width	The horizontal extent of the widget (see Remarks)
Height	The vertical extent of the widget (see Remarks)
menuName	The name of the WaveLinkMenu configuration associated with the widget
initialOption	The initial menu-index value of the widget (for example, passing a 1 indicates the first menu entry in the WaveLinkMenu configuration)
targetCollection	The name of the collection that will contain the widget

Returns

A pointer to the popup trigger widget

Throws

WaveLinkError

Remarks

The popup trigger populates its menu options with a stored WaveLinkMenu configuration. The menu must be stored on the device before the popup trigger can access it. The widget returns the menu index of the user-selected option.

The CreatePopupTrigger Method automatically stores the menu name associated with the current widget in the SpecialString Property of the Class WaveLinkWidget.

For the *Width* and *Height* parameters, pass a value of 0 to set the width or height automatically. Set the size of the widget to AUTOSIZE by setting both the width and height to zero (0).

We recommend that you use the AUTOSIZE setting for the height and width of the popup trigger. Using a value other than AUTOSIZE does not change the size of the widget, only the extent of the clickable area surrounding it.

Example

```
WaveLinkFactory factoryIface = new WaveLinkFactory();
WaveLinkWidgetCollection colIface = new
    WaveLinkWidgetCollection();
WaveLinkIO ioIface = new WaveLinkIO();
WaveLinkMenu mnuIface = new WaveLinkMenu();
    try {
        // Create the menu that appears in the widget
        mnuIface.ResetMenu();
        mnuIface.MenuWidth(18);
        mnuIface.AddOption("Basic Car Wash");
        mnuIface.AddOption("Basic Wash/Wax");
        mnuIface.AddOption("Carnuba Wax");
        mnuIface.AddOption("Interior Clean");
        mnuIface.AddOption("Full Detail");
        mnuIface.AddOption("Special Detail");
        mnuIface.AddOption("Device Version");
        mnuIface.AddOption("Exit");
        mnuIface.StoreMenu("MainMenu");
.
.
.
        ioIface.RFPrint(0, 0, "Select Wash:",
                WaveLinkIO.WLCLEAR);
        factoryIface.CreatePopupTrigger(3, 5, 0, 0,
                    "MainMenu", 1, colIface);
        colIface.StoreWidgets();
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

CreatePushButton Method

This member of Class WaveLinkFactory is supported on: Palm, CE

The CreatePushButton Method creates a set of buttons that store an on/off state. When the user selects a button, the selected button stores the "on" state, and all other buttons store the "off" state. Only one button can store the "on" state at any time.

Syntax

```
public WaveLinkWidget CreatePushButton(int XCoord,
                                       int YCoord,
                                       int Width, int Height,
                                       String menuName,
                                       int initialOption,
                                       WaveLinkWidgetCollection
                                       targetCollection)
                        throws WaveLinkError
```

Parameters

XCoord	The starting left coordinate of the widget (see Remarks)
YCoord	The starting top coordinate of the widget (see Remarks)
Width	The horizontal extent of the widget (see Remarks)
Height	The vertical extent of the widget (set Remarks)
menuName	The name of the WaveLinkMenu configuration associated with the widget
initialOption	The initial menu-index value of the widget (For example, passing a 1 indicates the first menu entry in the WaveLinkMenu configuration)
targetCollection	The name of the collection that will contain the widget

Returns

A pointer to the push button widget

Throws

WaveLinkError

Remarks

The push button widget populates its menu options with a stored WaveLinkMenu configuration. The menu must be stored on the device before

the push button can access it. The widget returns the menu index of the user-selected option.

You can set additional display properties for the push button widget using the DisplayFlags Property of the Class WaveLinkWidget.

The CreatePushButton Method automatically stores the menu name associated with the current widget in the SpecialString Property of the Class WaveLinkWidget.

The DefaultCoordinateType Property determines how the application interprets the values entered for the position, width, and height of the widget.

For the *Width* and *Height* parameters, pass a value of 0 to set the width or height automatically. Set the size of the widget to AUTOSIZE by setting both the width and height to zero (0).

We recommend that you use the AUTOSIZE setting for the height and width of the push button. Changing this value from its default does not change the size of the widget, only the extent of the clickable area surrounding it.

Example

```
WaveLinkFactory factoryIface = new WaveLinkFactory();
WaveLinkWidgetCollection colIface = new
    WaveLinkWidgetCollection();
WaveLinkIO ioIface = new WaveLinkIO();
WaveLinkMenu mnuIface = new WaveLinkMenu();
WaveLinkWidget myWidget;
    try {
        mnuIface.ResetMenu();
        mnuIface.MenuWidth(18);
        mnuIface.AddOption("1");
        mnuIface.AddOption("2");
        mnuIface.AddOption("3");
        mnuIface.StoreMenu("DegofDif");
.
.
.
        // In this example, show some context for the
        // widget.
        ioIface.RFPrint(0, 0, "Degree of  ",
                WaveLinkIO.WLCLEAR);
        ioIface.RFPrint(0, 0, "Difficulty:",
                WaveLinkIO.WLNORMAL);
       myWidget = factoryIface.CreatePushButton(3, 5, 0, 0,
                  "DegofDif", 1, colIface);
        // Format the push button
        myWidget.setDisplayFlags(WaveLinkWidget.PBFIXED);
        colIface.StoreWidgets();
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

CreateRepeaterButton Method

This member of Class WaveLinkFactory is supported on: Palm, CE

The CreateRepeaterButton Method creates a button that implies that something can be scrolled or incremented. The four available buttons are: left arrow, right arrow, up arrow, and down arrow.

Syntax

```
public WaveLinkWidget CreateRepeaterButton(int XCoord,
                                           int YCoord,
                                          int Width, int Height,
                                           int repeaterType,
                                           WaveLinkWidgetCollection
                                           targetCollection)
                        throws WaveLinkError
```

Parameters

XCoord	The starting left coordinate of the widget (see Remarks)
YCoord	The starting top coordinate of the widget (see Remarks)
Width	The horizontal extent of the widget (see Remarks)
Height	The vertical extent of the widget (see Remarks)
repeaterType	The type of repeater button (see Remarks)
targetCollection	The name of the collection that will contain the widget

Returns

A pointer to the repeater button widget

Throws

WaveLinkError

Remarks

For the *Width* and *Height* parameters, pass a value of 0 to set the width or height automatically. Set the size of the widget to AUTOSIZE by setting both the width and height to zero (0).

We recommend that you use the AUTOSIZE setting for the height and width of the repeater button. Changing this value from its default does not change the size of the widget, only the extent of the clickable area surrounding it.

The possible values for *repeaterType* are:

DOWNREPEATER - down arrow button

LEFTREPEATER	- left arrow button
RIGHTREPEATER	- right arrow button
UPREPEATER	- up arrow button

Example

```
WaveLinkFactory factoryIface = new WaveLinkFactory();
WaveLinkWidgetCollection colIface = new
    WaveLinkWidgetCollection();
.
.
.
    try {
        factoryIface.CreateRepeaterButton(1, 7, 0, 0,
                     WaveLinkWidget.UPREPEATER, colIface);
        factoryIface.CreateRepeaterButton(19, 7, 0, 0,
                    WaveLinkWidget.DOWNREPEATER, colIface);
        colIface.StoreWidgets();
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

CreateSelectorTrigger Method

This member of Class WaveLinkFactory is supported on: Palm, CE

The CreateSelectorTrigger Method creates a button that implies that a new screen appears when the user "clicks" it.

Syntax

```
public WaveLinkWidget CreateSelectorTrigger(int XCoord,
                                            int YCoord,
                                            int Width, int Height,
                                            String displayText,
                                            WaveLinkWidgetCollection
                                            targetCollection)
                      throws WaveLinkError
```

Parameters

XCoord	The starting left coordinate of the widget (see Remarks)
YCoord	The starting top coordinate of the widget (see Remarks)
Width	The horizontal extent of the widget (see Remarks)
Height	The vertical extent of the widget (see Remarks)
displayText	The display text of the widget
targetCollection	The name of the collection which will contain the widget

Returns

A pointer to the selector trigger widget

Throws

WaveLinkError

Remarks

For the *Width* and *Height* parameters, pass a value of 0 to set the width or height automatically. Set the size of the widget to AUTOSIZE by setting both the width and height to zero (0). AUTOSIZE sizes the selector trigger widget according to the size of the text that it contains.

Example

```
WaveLinkFactory factoryIface = new WaveLinkFactory();
WaveLinkWidgetCollection colIface = new
    WaveLinkWidgetCollection();
WaveLinkIO ioIface = new WaveLinkIO();
WaveLinkTerminal termIface = new WaveLinkTerminal();
```

```
.
.
.
        try {
            ioIface.RFPrint(0, 0, "WaveLink Corporation",
                    WaveLinkIO.WLCLEAR | WaveLinkIO.WLREVERSE);
            ioIface.RFPrint(0, 0, "     Test Demo      ",
                     WaveLinkIO.NORMAL);
            factoryIface.CreateSelectorTrigger(4,
                        (termIface.TerminalHeight() - 2), 0, 0,
                         " Configure ", colIface);
            colIface.StoreWidgets();
        }
        catch (WaveLinkError wlErr) {
            //Do error handling
        }
```

Class WaveLinkFile

`com.wavelink.clientui`

The public class **WaveLinkFile** is supported on all OS types.

The Class WaveLinkFile provides the WaveLink file access interface for both public users and the package classes.

Constructors

```
public WaveLinkFile()
```

```
public WaveLinkFile(IWaveLinkSession currentSession)
```

Methods

RFDeleteFile
RFFileCount
RFFileDate
RFFileName
RFFileSize
RFFileTime
RFGetFile
RFListFiles
RFListFilesEx
RFStoreFile
RFTransferFile

Remarks

The WaveLinkFile Object uses the standard DOS 8.3 file and wild card ("*") naming convention.

Method Summary

RFDeleteFile Method

- Deletes DOS files from the device.

RFFileCount Method

- Returns the total number of files returned by the last call to the RFListFiles Method.

RFFileDate Method

- Returns the last modification date of a specific file.

RFFileName Method

- Returns the name of a specific file.

RFFileSize Method

- Returns the size of a specific file in bytes.

RFFileTime Method

- Returns the last modification time of a specific file.

RFGetFile Method

- Retrieves a specific file into the current WaveLinkFile object.

RFListFilesEx Method

- Returns the number of files on the device and stores the file names in the current WaveLinkFile object.This method also stores the size, the last modification date, and the last modification time of the file.

RFStoreFile Method

- Stores data as a DOS file on the device.

RFTransferFile Method

- Transfers files between a network host and the device.

RFDeleteFile Method

This member of Class WaveLinkFile is supported on all OS types.

The RFDeleteFile Method removes files from a device.

Syntax

```
public void RFDeleteFile(String fileMask)
                    throws WaveLinkError,
                           IllegalArgumentException
```

Parameters

fileMask	The delete file mask ("*.*" deletes all files while "*.ton" deletes only tone files)

Throws

WaveLinkError

IllegalArgumentException

Remarks

The WaveLinkFile Object uses the standard DOS 8.3 file and wild card ("*") naming convention. For example, to delete all files from an RF devices memory, pass "*.*" for the *fileMask* parameter.

Example

See the WaveLinkFile Samples

RFFileCount Method

This member of Class WaveLinkFile is supported on all OS types.

The RFFileCount Method returns the total number of files returned by the last successful call to the RFListFiles Method.

Syntax

```
public int RFFileCount()
```

Returns

The number of files returned by the last call to the ListFiles Method

Remarks

The RFFileCount Method returns the file count from the last successful call to the RFListFiles Method. Use the RFListFiles Method to return the total number of files that are currently stored on a device.

Example

See the WaveLinkFile Samples

RFFileDate Method

This member of Class WaveLinkFile is supported on all OS types.

The RFFileDate Method returns the last file modification date of the file in the position specified by *fileIndex*.

Syntax

```
public String RFFileDate(int fileIndex)
```

Parameters

fileIndex	The zero-based index value of the file on the device

Returns

The date the file was last modified, returned in "YYYYMMDD" format

Remarks

The first file in the list is indexed as zero. For example, to return the file date for the third file in the current WaveLinkFile Object, you pass a 2.

To store the additional file time information in the WaveLinkFile object, you must use the RFListFilesEx Method instead of the RFListFiles Method.

Use the RFFileTime Method to return the time a file was last modified.

Use the RFFileSize Method to return the size of the file.

RFFileName Method

This member of Class WaveLinkFile is supported on all OS types.

The RFFileName Method returns the name of a file saved on a device.

Syntax

```
public String RFFileName(int fileIndex)
```

Parameters

fileIndex	The zero-based index value of the file on the device

Returns

The file name as a string

Remarks

The first file in the list is indexed as zero. For example, to return the file name for the third file in the current WaveLinkFile Object, you pass a 2.

For a complete listing of all files on a device, first retrieve the list using the RFListFiles Method, then loop the RFFileName Method for the entire number of files returned by RFFileCount Method.

Example

See the WaveLinkFile Samples

RFFileSize Method

This member of Class WaveLinkFile is supported on all OS types.

The RFFileSize Method returns the exact size of the specified file.

Syntax

```
public long RFFileSize(int fileIndex)
```

Parameters

fileIndex The zero-based index value of the file on the device

Returns

The size of the file in bytes

Remarks

The first file in the list is indexed as zero. For example, to return the file size for the third file in the current WaveLinkFile Object, you pass a 2.

Use the RFFileSize Method instead of the RFListFiles Method to store the additional file size information within the WaveLinkFile object.

Use the RFFileDate Method to return the date a file was last modified.

Use the RFFileTime Method to return the time a file was last modified.

RFFileTime Method

This member of Class WaveLinkFile is supported on all OS types.

The RFFileTime Method returns the time that the specified file was last modified.

Syntax

```
public String RFFileTime(int fileIndex)
```

Parameters

FileIndex	The zero-based index value of the file on the device

Returns

The time the file was last modified, returned in "HHMMSS" format

Remarks

The first file in the list is indexed as zero. For example, to return the file size for the third file in the current WaveLinkFile Object, you pass a 2.

You must use the RFListFilesEx Method instead of the RFListFiles Method to store the additional file time information within the WaveLinkFile object.

Use the RFFileDate Method to return the date a file was last modified.

Use the RFFileSize Method to return the size of file.

RFGetFile Method

This member of Class WaveLinkFile is supported on all OS types.

The RFGetFile Method returns the contents of the specified device file.

Syntax

```
public String RFGetFile(String fileName)
                    throws WaveLinkError,
                           IllegalArgumentException
```

Parameters

fileName	The base name of the device resident file

Returns

The contents of the device resident file

Throws

WaveLinkError

IllegalArgumentException

Remarks

The RFGetFile object uses the standard DOS 8.3 file naming convention

Example

See the WaveLinkFile Samples

RFListFiles Method

This member of Class WaveLinkFile is supported on all OS types.

The RFListFiles Method returns the total number of files saved on a device that match the input mask and stores the file names in the current object.

Syntax

```
public int RFListFiles(String fileMask)
                    throws WaveLinkError,
                           IllegalArgumentException
```

Parameters

fileMask	The list file mask (for example, "*.*" returns all files while "*.ton" returns only tone files)

Returns

The number of files on the device that match the file mask

Throws

WaveLinkError

IllegalArgumentException

Remarks

To return the value of the last successful RFListFiles Method without making an actual call to the device, use the RFFileCount Method.

Use the RFListFilesEx Method to store the file modification date and time, the file size, and the DOS 8.3 file name within the WaveLinkFile Object.

Example

See the WaveLinkFile Samples

RFListFilesEx Method

This member of Class WaveLinkFile is supported on all OS types.

Like the RFListFiles Method, the RFListFilesEx Method returns the total number of files saved on a device that match the input mask and stores the file names in the current object. In addition, it stores the date and time the file was last modified and its size.

Syntax

```
public int RFListFilesEx(String fileMask)
                    throws WaveLinkError,
                           IllegalArgumentException
```

Parameters

fileMask	The list file mask (for example, "*.*" returns all files while "*.ton" returns only tone files)

Throws

WaveLinkError

IllegalArgumentException

Returns

The number of files on the device that match the file mask

Remarks

Unlike the RFListFiles method, the RFListFilesEx method stores the complete DOS 8.3 file name in the WaveLinkFile Object.

Use the RFFileDate Method to return the last modification date of a file stored within the WaveLinkFile object.

Use the RFFileTime Method to return the last modification time of a file stored within the WaveLinkFile object.

Use the RFFileSize Method to return the size of a file stored within the WaveLinkFile object.

RFStoreFile Method

This member of Class WaveLinkFile is supported on all OS types.

The RFStoreFile Method saves data to the device in the specified file.

Syntax

```
public void RFStoreFile(String fileName,
                        String fileData)
                    throws WaveLinkError,
                           IllegalArgumentException
```

Parameters

fileName	The base name of the file, including the file extension
fileData	The data to be stored in the file

Throws

WaveLinkError

IllegalArgumentException

Remarks

The WaveLinkFile object uses the standard DOS 8.3 file naming convention.

You must specify a 3-character file extension when passing *fileName*.

NOTE Do not use a WaveLinkFile object to store the other WaveLink objects (menus, tones, and barcodes). Each of these has its own methods for interacting with the device.

Example

```
WaveLinkFile fileIface = new WaveLinkFile();
string strBuffer;
strBuffer = "A string representing data to be
            stored on the device!";
    try {
        fileIface.RFStoreFile("FileDemo.dat", strBuffer);
    }
    catch (WaveLinkError wlErr) {
        // Do error handling
    }
```

RFTransferFile Method

This member of Class WaveLinkFile is supported on all OS types.

The RFTransferFile Method transfers files bi-directionally between a network host and device.

Syntax

```
public void RFTransferFile(String sourceFile,
                           String targetFile,
                           boolean toDevice)
                    throws WaveLinkError,
                           IllegalArgumentException
```

Parameters

sourceFile	The source file name to be transferred (see Remarks)
targetFile	The name of the target file (see Remarks)
toDevice	A boolean true or false value that determines which direction the file is transferred (see Remarks)

Throws

WaveLinkError

IllegalArgumentException

Remarks

If the *toDevice* value is true, the source file transfers from the host to the RF client. If this value is set to false, the source file transfers from the RF client to the host.

If the file resides on the host, enter the complete path for the file (for example, “C:\Temp\File.txt) or enter a UNC path. If the file resides on the local device, enter only the file name.

Example

```
WaveLinkFile fileIface;

fileIface.RFTransferFile("C:\temp\logo.bmp",
"logo.bmp", true);
```

WaveLinkFile Samples

The following sample code demonstrates the implementation and capabilities of the Class WaveLinkFile and its associated methods.

Java Sample

```
/**
* This source code is for demonstration
* purposes only and should be treated as such.
*
* This application demonstrates the use of the various
* WaveLinkFile methods. It assumes a display area of 20x8.
* In practice, you should use a WaveLinkTerminal
* object to format output to fit the current device.
*
*/
import com.wavelink.clientui.*;
import java.io.*;
public class JavaFileDemo extends WaveLinkApplication {
    public void WaveLinkMain(String[] args)
    {
        WaveLinkIO ioIface = new WaveLinkIO();
        WaveLinkFile fileIface = new WaveLinkFile();
        String strBuffer, csBuffer;
        int nTmp;
        // The following lines clear the display and output the
        // app title in reverse video.
        try {
            ioIface.RFPrint(0, 0, " Welcome To The ",
                    WaveLinkIO.WLCLEAR | WaveLinkIO.WLREVERSE);
            ioIface.RFPrint(0, 1, " WaveLinkFile Demo ",
                    WaveLinkIO.WLREVERSE);
            ioIface.RFPrint(0, 3, "Hit a key to proceed",
                     WaveLinkIO.WLNORMAL);
        // Hide the cursor for a cleaner display
            ioIface.RFPrint(0, 20, "", WaveLinkIO.WLNORMAL);
            ioIface.GetEvent();
        // The WaveLinkFile object provides an easy way for
        // applications to store, retrieve, list,
        // and delete files from the device.
        // Sometimes it is desirable to determine if a
        // particular file resides on the device.
```

```
// The use ofRFListFiles Method is the quickest and
// easiest method to do this.
    fileIface.RFListFiles("*.*");
// To find a particular file, pass the file name as
// the parameter to RFListFiles:
// nFound = rfFil.RFListFiles ("Target.dat")
// Since we just wish to list the files on the current RF
// device, we used the wildcard notation ("*.*").
    nTmp = fileIface.RFFileCount();
// It would have been quicker to use the return value
// from RFListFiles instead of an additional
// function call, but RFFileCount was used for
// demonstration purposes.
    for ( int lcv = 0; lcv < nTmp; lcv++ ) {
        if ( (lcv % 7) == 0 )
            ioIface.GetEvent();
        ioIface.RFPrint(0, 0, " RF Device Files: ",
               WaveLinkIO.WLCLEAR
               | WaveLinkIO.WLFLUSHOUTPUT);
        ioIface.RFPrint(0, lcv+1,
                fileIface.RFFileName(lcv),
                WaveLinkIO.WLNORMAL);
    }
        ioIface.GetEvent();
// When it is necessary to store data files on the
// device, the WaveLinkFile object provides
// methods to store, retrieve, and delete them.
    ioIface.RFPrint(0, 2, "Storing data file...",
            WaveLinkIO.WLCLEAR
            | WaveLinkIO.WLFLUSHOUTPUT);
    strBuffer = "A string representing data to be"
                +"stored on a device!";
                fileIface.RFStoreFile("FLDemo.dat",
                            strBuffer);
    ioIface.RFPrint (0, 3, "Getting data file...",
            WaveLinkIO.WLNORMAL
             | WaveLinkIO.WLFLUSHOUTPUT);
    csBuffer = fileIface.RFGetFile("FLDemo.dat");
    if ( strBuffer.equals(csBuffer) )
        ioIface.RFPrint(0, 4, "Data Read Success!",
                 WaveLinkIO.WLNORMAL);
    else
        ioIface.RFPrint(0, 4, "Data Read Failure!",
                WaveLinkIO.WLNORMAL);
    ioIface.RFPrint(0, 6, " Hit a key! ",
            WaveLinkIO.WLREVERSE);
```

```
                ioIface.RFPrint(0, 9, "", WaveLinkIO.WLNORMAL);
                ioIface.GetEvent();
                // Anytime an application changes the configuration
                // of a device, it should return it to its
                // original values during cleanup
                fileIface.RFDeleteFile("FLDemo.dat");
               // NOTE : Do NOT use a WaveLinkFile object to store
               // other WaveLink objects (menus, tones, barcodes).
               // Each of these has its own methods for interacting
               // with the device.
            }
            catch ( WaveLinkError wlErr ) {
            //Do error handling
            }
        }
    }
```

Class WaveLinkFont

```
com.wavelink.clientui
```

The public class WaveLinkFont is supported on: CE

The WaveLinkFont class represents fonts which are used to render text for widgets on screens. One a WaveLinkFont object has been created, it should be added to a WidgetOptions object using setFont().

Constructors

```
public WaveLinkFont()
Creates a font object with default settings.

public WaveLinkFont(java.lang.String face,int size,int
style)
Creates a font object with the specified attributes.

Parameters:
    face - The font face
    size - The size to display the font
    style - The font style
```

Fields

STYLE_BOLD	STYLE_STRIKEOUT
STYLE_ITALICS	STYLE_UNDERLINE
STYLE_STANDARD	

Methods

clone	hashCode
equals	setFace
getFace	setSize
getSize	setStyle
getStyle	

Field Detail

STYLE_BOLD

```
public static final int STYLE_BOLD
```

Uses the bold font style. Styles can be combined by using or.

STYLE_ITALICS

```
public static final int STYLE_ITALICS
```

Uses the italics font style. Styles can be combined by using or.

STYLE_STANDARD

```
public static final int STYLE_STANDARD
```

Uses the standard font style.

STYLE_STRIKEOUT

```
public static final int STYLE_STRIKEOUT
```

Uses the strikeout font style. Styles can be combined by using or.

STYLE_UNDERLINE

```
public static final int STYLE_UNDERLINE
```

Uses the underline font style. Styles can be combined by using or.

Method Summary

clone

- Returns an exact copy of a font object.

equals

- Used to determine if two font objects have the same values.

getFace

- Gets the font face.

getSize

- Gets the font size.

getStyle

- Gets the font style.

setFace

- Sets the font face.

setSize

- Sets the font size.

setStyle

- Sets the font style.

Example

```
WaveLinkFont myFont = new WaveLinkFont("Arial", 10,
WaveLinkFont.STYLE_STANDARD);
```

clone Method

This member of Class WaveLinkFont is supported on: CE

The clone Method returns an exact copy of a font object.

Syntax

```
public java.lang.Object clone()
```

Returns

An object that is a copy of this WaveLinkFont

equals Method

This member of Class WaveLinkFont is supported on: CE

The equals Method is used to determine if two font objects have the same values.

Syntax

```
public boolean equals(WaveLinkFont f)
```

Parameters

F	Another WaveLinkFont object to compare against

Returns

True if the two objects have the same values (font, size, color), false if not

getFace Method

This member of Class WaveLinkFont is supported on: CE

The getFace Method gets the font face.

Syntax

```
public java.lang.String getFace()
```

Returns

The font name.

getSize Method

This member of Class WaveLinkFont is supported on: CE

The getSize Method returns the font size.

Syntax

```
public int getSize()
```

Returns

The font size

getStyle Method

This member of Class WaveLinkFont is supported on: CE

The getStyle Method gets the font style.

Syntax

```
public int getStyle()
```

Returns

The font style. See Class WaveLinkFont constants.

setFace Method

This member of Class WaveLinkFont is supported on: CE

The setFace Method sets the font face.

Syntax

```
public void setFace(java.lang.String face)
```

Parameters

face	new face

setSize Method

This member of Class WaveLinkFont is supported on: CE

The setSize Method sets the font size.

Syntax

```
public void setSize(int size)
```

Parameters

size	the font size

setStyle Method

This member of Class WaveLinkFont is supported on: CE

The setStyle Method sets the font style.

Syntax

```
public void setStyle(int style)
```

Parameters

style	the font style

Class WaveLinkIO

`com.wavelink.clientui`

The public class **WaveLinkIO** extends WaveLinkProto and is supported on all OS types.

The Class WaveLinkIO provides the interface to the device for displaying text and querying for input. It also contains the screen image manipulation functionality (PushScreen, PullScreen, and RestoreScreen).

Fields

WLALPHA_ONLY
WLBACKLIGHT
WLCAPSLOCK
WLCLEAR
WLCLR_INPUT_BUFFER
WLCLREOLN
WLCLREOS
WLCOMMANDTYPE
WLDISABLE_FKEYS
WLDISABLE_SCAN
WLECHO_ASTERISK
WLFLUSHOUTPUT
WLFORCE_ENTRY
WLIGNORE_CRLF
WLIGNORE_KEY
WLIGNORE_WIDGETS
WLINCLUDE_DATA
WLINCLUDE_WIDGETS
WLKEYTYPE
WLMAXLENGTH
WLMENUBARTYPE
WLNO_NONPRINTABLE
WLNO_RETURN_BKSP
WLNO_RETURN_FILL
WLNORMAL
WLNORMALKEYS
WLNUMERIC_ONLY
WLPOPUPTYPE
WLREVERSE
WLSCANTYPE
WLSOFT_TRIGGER
WLSUPPRESS_ECHO
WLWIDGETTYPE

Constructors

`public WaveLinkIO(IWaveLinkSession currentSession)`

Methods

ActivateScanner	RestoreScreen
AddHotKey	RFAux
ClearHotKeys	RFFlushoutput
EventCount	RFInput
GetEvent	RFPrint
LastBarcodeType	RFSetFill
LastExtendedType	RFSpool
LastInputType	SetInputTimeout
PullScreen	TellEvent
PushScreen	WaitForReconnect

Field Detail

WLALPHA_ONLY

```
public static final long WLALPHA_ONLY
```

RFInput Input Mode - Accept only alphabetical characters as input.

WLBACKLIGHT

```
public static final long WLBACKLIGHT
```

RFInput Input Mode - Turn the backlight on for the current input.

WLCAPSLOCK

```
public static final int WLCAPSLOCK
```

RFInput Key Mode - Set the device keyboard into capslock mode.

WLCLEAR

```
public static final long WLCLEAR
```

RFPrint Output Mode - Clear the contents of the display before displaying the output text.

WLCLR_INPUT_BUFFER

```
public static final long WLCLR_INPUT_BUFFER
```

RFInput Input Mode - Clear any unused data from the input buffer before accepting input from the user.

WLCLREOLN

```
public static final long WLCLREOLN
```

RFPrint Output Mode - Clear the content of the specified row from the specified column before displaying output text.

WLCLREOS

```
public static final long WLCLREOS
```

RFPrint Output Mode - Clear the content of the display from the specified row to the bottom before displaying output text.

WLCOMMANDTYPE

```
Public static final int WLCOMMANDTYPE
```

Input Type – Function key input.

WLDISABLE_FKEYS

```
public static final long WLDISABLE_FKEYS
```

RFInput Input Mode - Disable the function keys for the current input.

WLDISABLE_SCAN

```
public static final long WLDISABLE_SCAN
```

RFInput Input Mode - Disable the scanner for the current input.

WLECHO_ASTERISK

```
public static final long WLECHO_ASTERISK
```

RFInput Input Mode - Display an '*' for each key pressed for the current input.

WLFLUSHOUTPUT

```
public static final long WLFLUSHOUTPUT
```

RFPrint Output Mode - Send the current output text along with anything currently in the output queue.

WLFORCE_ENTRY

```
public static final long WLFORCE_ENTRY
```

RFInput Input Mode - Do not accept empty inputs.

WLIGNORE_CRLF

```
public static final long WLIGNORE_CRLF
```

RFInput Input Mode - Includes any carriage returns or line feeds embedded within the returned input. By default, returned input strings containing these characters are broken into separate input packets.

WLIGNORE_KEY

```
public static final long WLIGNORE_KEY
```

RFInput Input Mode - Disable the normal keys for the current input.

WLIGNORE_WIDGETS

```
public static final int WLIGNORE_WIDGETS
```

Push/Pull Screen Mode - Do not push widgets into *.scr file when Push/Pull Screen is called.

WLINCLUDE_DATA

```
public static final long WLINCLUDE_DATA
```

RFInput Input Mode - If the input is terminated with a function key and there is data within the field, return the data as well.

WLINCLUDE_WIDGETS

```
public static final int WLINCLUDE_WIDGETS
```

Push/Pull Screen Mode - Include widgets in *.scr file when Push/Pull Screen is called.

WLKEYTYPE

```
Public static final int WLKEYTYPE
```

Input Type – Standard key input.

WLMAXLENGTH

```
public static final long WLMAXLENGTH
```

RFInput Input Mode - Limit keyed input to the length of the field but do not return until entered.

WLMENUBARTYPE

```
public static final int WLMENUBARTYPE
```

Input Type – Input from a menubar widget.

WLNO_NONPRINTABLE

```
public static final long WLNO_NONPRINTABLE
```

RFInput Input Mode - Do not display non-printable characters in the input field.

WLNO_RETURN_FILL

```
public static final long WLNO_RETURN_FILL
```

RFInput Input Mode - Do not return from current input when input length is reached.

WLNO_RETURN_BKSP

```
public static final long WLNO_RETURN_BKSP
```

RFInput Input Mode - Do not exit input when a backspace is pressed at the first cell.

WLNORMAL

```
public static final long WLNORMAL
```

RFPrint Output Mode - Display the output text using normal video.

WLNORMALKEYS

```
public static final int WLNORMALKEYS
```

RFInput Key Mode - Set the device keyboard into non-capslock mode.

WLNUMERIC_ONLY

```
public static final long WLNUMERIC_ONLY
```

RFInput Input Mode - Accept numeric characters only.

WLPOPUPTYPE

```
Public static final int WLPOPUPTYPE
```

Input Type – Input from a popup trigger widget.

WLREVERSE

```
public static final long WLREVERSE
```

RFPrint Output Mode - Display the output text using reverse video.

WLSCANTYPE

```
Public static final int WLSCANTYPE
```

Input Type – Scanner input.

WLSOFT_TRIGGER

```
public static final long WLSOFT_TRIGGER
```

RFInput Input Mode - Trigger the scanner for the input automatically.

WLSUPPRESS_ECHO

```
public static final long WLSUPPRESS_ECHO
```

RFInput Input Mode - Suppress the echoing of key presses.

WLWIDGETTYPE

```
Public static final int WLWIDGETTYPE
```

Input Type – Input from a widget.

Method Summary

ActivateScanner Method

- Activates the device's scanner without the prompt of an RFInput or GetEvent call.

AddHotKey Method

- Defines custom command keys for your applications.

ClearHotKeys Method

- Clears custom command keys from the WaveLinkIO object.

EventCount Method

- Returns the number of input events in the input queue.

GetEvent Method

- Returns a value after a single key, function key combination, or scan input event has occurred.

LastBarcodeType Method

- Returns the last barcode type scanned by the device.

LastExtendedType Method

- Returns the last extended type, either a barcode type or widget ID, of the last input call.

LastInputType Method

- Returns the last input type.

PullScreen Method

- Restores a previously saved screen while deleting the screen file from the device memory.

PushScreen Method

- Saves the current displayed screen for later restoration.

RestoreScreen Method

- Restores a previously saved screen and does *not* remove the screen file.

RFAux Method

- Sends data directly to the device serial port.

RFFlushoutput Method

- Sends all data in the output buffer to the device.

RFInput Method

- Returns user input from a device.

RFPrint Method

- Prints data to the device's display.

RFSetFill Method

- Defines the fill character for RFInput input prompts.

RFSpool Method

- Spools label data and a copy count to the device's serial port.

SetInputTimeout Method

- Defines the amount of time that elapses before an RFInput input prompt expires.

TellEvent Method

- Checks the input queue for data sent from the device without removing the data from the input queue.

WaitForReconnect Method

- Defines the number of seconds to wait before the application attempts to reconnect to the device.

ActivateScanner Method

This member of Class WaveLinkIO is supported on all OS types.

The ActivateScanner Method turns on the device scanner with the specified barcode configuration.

Syntax

```
public void ActivateScanner(String barcodeCfg)
                    throws WaveLinkError
```

Parameters

barcodeCfg The barcode configuration used by the input call

Throws

WaveLinkError

Remarks

Enter null ("") for the *barcodeCfg* parameter to use the default configuration.

This function has been designed primarily for polling scenarios, such as those in which TellEvent is used.

AddHotKey Method

This member of Class WaveLinkIO is supported on all OS types.

The AddHotKey Method includes the specified key as an input command type.

Syntax

```
public void AddHotKey(char hotKey)
```

Parameters

hotKey	The character to be added, passed in single quotations

Remarks

Use the ClearHotKeys Method to clear any custom command keys you have created.

The return type for a custom command key is WLCOMMANDTYPE. Use the LastInputType Method to process input based on the return type.

Example

```
WaveLinkIO IOIface = new WaveLinkIO();
.
.
.
   try {
        IOIface.AddHotKey(Asc('Z'));
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

ClearHotKeys Method

This member of Class WaveLinkIO is supported on all OS types.

The ClearHotKeys Method removes the application specified hotkeys from the WaveLinkIO object.

Syntax.

```
public void ClearHotKeys()
```

Remarks

Use the AddHotKey Method to create custom command keys for your applications.

EventCount Method

This member of Class WaveLinkIO is supported on all OS types.

The EventCount Method returns the number of input events currently in the input queue.

Syntax

```
public int EventCount()
```

Remarks

If multiple input events are in the input queue, subsequent calls to either RFInput Method or GetEvent Method in the same WaveLinkIO object must be made to return the additional events. Each subsequent call that you make to RFInput or GetEvent removes one additional event from the input stack.

GetEvent Method

This member of Class WaveLinkIO is supported on all OS types.

The GetEvent Method requests and returns a single event from the device. Single events are defined as either a key stroke, a function key sequence, or a scanned barcode.

Syntax

```
public String GetEvent()
                    throws WaveLinkError
```

Returns

The single key, function key, or scan input value

Throws

WaveLinkError

Remarks

Unlike a standard input, which only returns after a carriage return or the maximum input length has been reached, GetEvent immediately returns a single character, function-key combination, or a scanner event.

GetEvent only returns the first key stroke or function key sequence but returns the entire value for a scan input.

Use the LastInputType Method to return the origin of the last input type (i.e., scanner, keypad, or function key).

Example

```
WaveLinkIO IOIface = new WaveLinkIO();
WaveLinkTerminal termIface = new WaveLinkTerminal();
String pszVerifyIn;
.
.
.
    try {
        // Do something to instruct the user to enter input.
        IOIface.RFPrint(0, (termIface.TerminalHeight() - 1),
                 "Verify y/n ? ",
               WaveLinkIO.WLREVERSE | WaveLinkIO.WLCLREOLN);
         pszVerifyIn = IOIface.Getvent();
    }
    catch (WaveLinkError wlErr) {
    }
```

LastBarcodeType Method

This member of Class WaveLinkIO is supported on all OS types.

The LastBarcodeType Method returns the barcode type of the last input call.

Syntax

```
public int LastBarcodeType()
```

Returns

The barcode type of the last input call. If the last input returned is not a barcode, the method returns a -1.

Remarks

The possible barcode types include:

B_UPC

CODABAR

CODE_11

CODE_39

CODE_93

CODE_128

CODE_D25

CODE_I25

D25_IATA

EAN_8

EAN_13

MSI

PDF_417

TO_39

UCC_128

UPC_A

UPC_E0

UPC_E1

WLNOSYMBOLOGY

See the Constant Values appendix for the numeric equates returned by the function.

Example

```
WaveLinkIO ioIface = new WaveLinkIO();
WaveLinkBarcode barcodeIface = new WaveLinkBarcode();
WaveLinkFactory factoryIface = new WaveLinkFactory();
String resultStr;
    // Create a barcode configuration and store it as
    //"BCDemo"(not shown).
.
.
.
    try {
        //Use RFInput to obtain input
        resultStr = ioIface.RFInput("", 20, 0, 3, "BCDemo",
                    WaveLinkIO.WLNORMALKEYS,
                    WaveLinkIO.WLDISABLE_KEY);
        switch(ioIface.LastInputType())
        {
            case ioIface.WLCOMMANDTYPE:
                //Process command type
                break;

            case ioIface.WLKEYTYPE:
                //Process keyed command
                break;
            case ioIface.WLSCANTYPE:
                switch(ioIface.LastBarcodeType())
                {
                   case WaveLinkBarcode.CODE_39:
                        // Process Code_39
                    break;
                   case WaveLinkBarcode.UPC-A:
                        // Process UPC_A
                    break;
                }
                break;
        }
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

LastExtendedType Method

This member of Class WaveLinkIO is supported on all OS types.

The LastExtendedType Method returns the extended type of the last input call. The extended type can be a barcode type or a widget ID.

Syntax

```
public int LastExtendedType()
```

Returns

The barcode type or widget ID of the last input call (see Remarks)

Remarks

When processing input from a widget, use the LastExtendedType Method to obtain the widget ID. The widget ID is returned through a previous RFInput Method call or a GetEvent Method call.

The possible barcode types include:

B_UPC

CODABAR

CODE_11

CODE_39

CODE_93

CODE_128

CODE_D25

CODE_I25

D25_IATA

EAN_8

EAN_13

MSI

PDF_417

TO_39

UCC_128

UPC_A

UPC_E0

UPC_E1

WLNOSYMBOLOGY

See the Constant Values appendix for the numeric equates returned by the function.

Example

```
/**
* This source code is for demonstration
* purposes only and should be treated as such.
*
* NOTE: For this demo, we are assuming a 20x8 display area.
*/
package samplecode;
import com.wavelink.clientui.*;
import java.io.*;
public class IOSampleLastType extends WaveLinkApplication {
    WaveLinkIO ioIface = new WaveLinkIO();
    WaveLinkFactory factoryIface = new WaveLinkFactory();
    WaveLinkWidgetCollection colIface = new
WaveLinkWidgetCollection();
    WaveLinkWidget widgetIfaceQuit;
    WaveLinkWidget widgetIfaceKeepOn;
    int quitID;
    String resultStr;
    public IOSampleLastType() throws WaveLinkError    {
    }
    private void initialize() throws WaveLinkError {
    // Create "Quit" widget and set the integer quitID
    // to its widget ID.
        widgetIfaceQuit = factoryIface.CreateButton(2, 2, 8,
                          1, "Quit", colIface);
        widgetIfaceKeepOn = factoryIface.CreateButton(2, 5,
                            8, 1, "Keep on", colIface);
        quitID = widgetIfaceQuit.getWidgetID();
        colIface.StoreWidgets();
    }
    private void start() throws WaveLinkError{
        try {
            evaluate:
            for ( ;; ) {
            //Use RFInput or GetEvent to obtain input
                resultStr = ioIface.GetEvent();
                switch ( ioIface.LastInputType() ) {
                case WaveLinkIO.WLCOMMANDTYPE:
                //Process command type
```

```
                        McastSend.send("case WLCOMMANDTYPE") ;
                        break;
                    case WaveLinkIO.WLKEYTYPE:
                    //Process keyed command
                        McastSend.send("case WLKEYTYPE") ;
                        break;
                    case WaveLinkIO.WLWIDGETTYPE:
                        McastSend.send("case WLWIDGETTYPE") ;
                       if ( ioIface.LastExtendedType() == quitID ) {
                            //Quit button pressed, exit process
                            McastSend.send("LastExtendedType()
                                           == quitID") ;
                            break evaluate;
                        }
                        else {
                            McastSend.send("LastExtendedType()
                                          != quitID") ;
                            break;
                        }
                    }
                }
            }
            catch ( WaveLinkError wlErr ) {
            // Do error handling
            }
        }// initialize
        public void WaveLinkMain( String arg[] )
        {
            try {
                McastSend.send("Starting application 1" );
                IOSampleLastType localObj = new IOSampleLastType();
                localObj.initialize();
                localObj.start();
            }//try
            catch ( WaveLinkError wlErr ) {
            } // catch (WaveLinkError wlErr)
            McastSend.send("Exiting application") ;
        } // End of main
    }
```

LastInputType Method

This member of Class WaveLinkIO is supported on all OS types.

The LastInputType Method returns the type of the last input call.

Syntax

```
public int LastInputType()
```

Returns

The input type constant (see Remarks)

Remarks

The possible return constants for input types are:

WLCOMMANDTYPE	- Function Key input
WLKEYTYPE	- Standard Key input
WLMENUBARTYPE	- Menubar input
WLPOPUPTYPE	- Popup trigger input
WLSCANTYPE	- Scanner input
WLTIMEDOUT	- RFInput input prompt expired
WLWIDGETTYPE	- Widget input

See the Constant Values appendix for the numeric equates returned by the function.

Example

```
WaveLinkIO ioIface = new WaveLinkIO();
String resultStr;
.
.
.
    try {
        //Use RFInput or GetEvent to obtain input
        resultStr = ioIface.GetEvent();
        switch(ioIface.LastInputType())
        {
            case ioIface.WLCOMMANDTYPE:
                //Process command type
                break;
            case ioIface.WLKEYTYPE:
                //Process keyed command
                break;
            case ioIface.WLSCANTYPE:
                // Process scanned input
                break;
        }
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

PullScreen Method

This member of Class WaveLinkIO is supported on all OS types.

The PullScreen Method restores a previously saved screen as the current screen while simultaneously deleting the screen file from the RF device's memory.

Syntax

```
public void PullScreen(String fileName)
                    throws WaveLinkError

public void PullScreen(String fileName,
                       int screenFlags)
                    throws WaveLinkError
```

Parameters

fileName	The file containing the image
screenFlags	Determines whether widgets are present in the image file (see Remarks)

Throws

WaveLinkError

Remarks

The possible values for *screenFlags* are:

WLIGNORE_WIDGETS - Includes widgets in image file

WLINCLUDE_WIDGETS - Does not include widgets in image file

Use the PushScreen Method to save a current screen file to device memory.

Use the RestoreScreen Method if you wish to restore a screen without deleting it from the device memory.

When used in conjunction, PushScreen and PullScreen or RestoreScreen can temporarily interrupt a displayed screen within your application, then restore the screen to its original state. This creates faster display when re-using screens.

Example

See the PushScreen Method

PushScreen Method

This member of Class WaveLinkIO is supported on all OS types.

The PushScreen Method saves the current displayed screen directly to the device's non-volatile memory for later restoration.

Syntax

```
public void PushScreen(String fileName)
                    throws WaveLinkError

public void PushScreen(String fileName,
                       int screenFlags)
                    throws WaveLinkError
```

Parameters

fileName	The file in which to store the screen image
screenFlags	A flag that determines whether widgets are pushed with the screen (see Remarks)

Throws

WaveLinkError

Remarks

The possible values for *screenFlags* are:

WLIGNORE_WIDGETS - Includes widgets in image file

WLINCLUDE_WIDGETS - Does not include widgets in image file

Use the PushScreen Method to save a current screen file to device memory.

Use the RestoreScreen Method if you wish to restore a screen without deleting it from the device memory.

When used in conjunction, PushScreen and PullScreen or RestoreScreen can temporarily interrupt a displayed screen within your application, then restore the screen to its original state. This creates faster display when re-using screens.

NOTE We recommend that you clear the output queue immediately before using the PushScreen Method.

Example

```
WaveLinkIO IOIface = new WaveLinkIO();
.
```

```
.
.
    try {
        // Flush the output before using PushScreen.
        // You can flush the output with:
        // 1. An input call
        // 2. TheRFFlushoutput Method.
        // 3. The RFPrint RFPrint Method when WLFLUSHOUTPUT
        //    is set as the display mode.
        IOIface.RFPrint(0, 7, " Version 3.01 ",
                WaveLinkIO.WLNORMAL
                | WaveLinkIO.WLFLUSHOUTPUT);
        IOIface.PushScreen("Version");
.
.
.
        // Re-display the Version screen later.
        IOIface.PullScreen("Version");
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

RestoreScreen Method

This member of Class WaveLinkIO is supported on all OS types.

The RestoreScreen Method paints the screen with the specified stored image without deleting the image file.

Syntax

```
public void RestoreScreen(String fileName)
                    throws WaveLinkError

public void RestoreScreen(String fileName,
                          int screenFlags)
                    throws WaveLinkError
```

Parameters

fileName	The file containing the image
screenFlags	A flag that determines whether widgets are pushed with the screen (see Remarks)

Throws

WaveLinkError

Remarks

The possible values for *screenFlags* are:

WLIGNORE_WIDGETS - Includes widgets in image file

WLINCLUDE_WIDGETS - Does not include widgets in image file

Use the PushScreen Method to save a current screen file to device memory.

Use the RestoreScreen Method if you wish to restore a screen without deleting it from the device memory.

When used in conjunction, PushScreen and PullScreen or RestoreScreen can temporarily interrupt a displayed screen within your application, then restore the screen to its original state. This creates faster display when re-using screens.

Example

```
WaveLinkIO IOIface = new WaveLinkIO();
.
.
.
    try {
        // Flush the output before using PushScreen.
        // You can flush the output with:
```

```
            // 1. An input call
            // 2. The RFFlushoutput Method.
            // 3. The RFPrint Method when WLFLUSHOUTPUT is
            //    set as the display mode.
            IOIface.RFPrint(0, 7, " Version 3.01 ",
                    WaveLinkIO.WLNORMAL
                    | WaveLinkIO.WLFLUSHOUTPUT);
            IOIface.PushScreen("Version");
.
.
.
            // Re-display the Version screen later.
            IOIface.RestoreScreen("Version");
        }
        catch (WaveLinkError wlErr) {
            //Do error handling
        }
```

RFAux Method

This member of Class WaveLinkIO is supported on all OS types.

NOTE Deprecated - Please use Class WaveLinkAuxPort instead.

The RFAux Method sends data directly to the device for routing to its COM port (usually a printer).

Syntax

```
public void RFAux(String outputData)
                    throws WaveLinkError
```

Parameters

outputData	A byte array containing the target data

Throws

WaveLinkError

Remarks

Use the Class WaveLinkAuxPort instead of the RFAux Method for serial port communication. The RFAux Method is retained only for compatibility purposes.

RFFlushoutput Method

This member of Class WaveLinkIO is supported on all OS types.

The RFFlushoutput Method clears the output queue sending all waiting data to the device.

Syntax

```
public void RFFlushoutput()
                    throws WaveLinkError
```

Throws

WaveLinkError

Remarks

For wireless network efficiency purposes, data in an output queue is not sent to an RF device until either an input call is made or the maximum output size (100 bytes) of a radio packet has been reached. RFFlushoutput forces data in the output queue.

RFInput Method

This member of Class WaveLinkIO is supported on all OS types.

The RFInput Method returns user input events from a mobile device over a wireless network.

Syntax

```
public String RFInput(String defaultString, int nLength,
                      int nCol, int nRow,
                      String barcodeCfg,
                      int keyMode, long inputMode)
                     throws WaveLinkError
```

Parameters

defaultString	Default text that appears in the input prompt (“”). The default value is not be included in returned input (see Remarks)
nLength	The length, in characters, of the input field
nCol	The x-coordinate of the input field. Columns are numbered from the left starting with column 0
nRow	The y-coordinate of the input field. Rows are numbered from the top starting with row 0
barcodeCfg	The barcode configuration to use with this input. Enter null ("") for this parameter to specify the default configuration
keyMode	The default keyboard mode (see Remarks)
inputMode	The input characteristics (see Remarks)

Returns

The event’s contents as a string

Throws

WaveLinkError

Remarks

The characters entered in the *defaultString* parameter *are not* included with the returned input. For example, if the user presses Enter without changing the default value, the ASCII code for Enter (Chr$(13)) is returned.

The possible values for *keyMode* are:

WLCAPSLOCK - Caps lock on

WLNORMALKEYS - Caps lock off

The possible values for *inputMode* are:

WLALPHA_ONLY - Accepts only alphabetic input

WLBACKLIGHT - Enables the device display backlight

WLCLR_INPUT_BUFFER - Clears the input buffer of any pending data

WLDISABLE_SCAN - Disable the device scanner

WLDISABLE_FKEYS - Disable function keys

WLDISABLE_KEY - Disable the device keypad

WLECHO_ASTERISK - Echos an asterisk (“*) to device screen with each key entered

WLFORCE_ENTRY - Input is not returned unless an actual value is entered in the input prompt

WLINGORE_CRLF - Includes any carriage returns or line feeds embedded within the returned input string. By default, returned input strings containing these characters are broken into separate input packets

WLINCLUDE_DATA - RFInput returns both data entered at an input prompt and any additional function key input

WLMAXLENGTH - RFInput does not accept more characters than defined in the *nLength* parameter

WLNO_NONPRINTABLE - Suppresses non-printable characters. This input mode is used primarily when using foreign character sets

WLNO_RETURN_BKSP - Input is not returned if the Backspace key is pressed in the first position of the input prompt

WLNO_RETURN_FILL - Input is not returned until the Enter key is pressed, even if the maximum input length is reached

WLNUMERIC_ONLY - Accepts only numeric input

WLSOFT_TRIGGER - Automatically activates the device’s scanner when the RFInput function is called

WLSUPPRESS_ECHO - Disable display echo of input

NOTE Multiple input mode options may be used by "ORing" the desired values (for example, WLDISABLE_SCAN | WLDISABLE_FKEYS).

See the Constant Values appendix for the numeric equates returned by the function.

RFInput returns input immediately after either the Enter key is pressed, the maximum input length has been reached, the Backspace key is pressed in the first position, or a function key is pressed.

If a function key is pressed, the RFInput only returns the value for the function key pressed, discarding any data entered in the input prompt. Use WLINCLUDE_DATA to return both the data entered in the input prompt and the function key value). Note that if you use WLINCLUDE_DATA, only the data entered in the input prompt is removed from the input stack on the initial call to RFInput. A subsequent call to either RFInput or GetEvent Method within the same WaveLinkIO object is required to remove the function key from the input stack.

You may further limit barcode input types through the use of Barcode Configuration files. Please see Class WaveLinkBarcode for more information.

To return a single key, function key combination, or scan input use the GetEvent Method.

Use the TellEvent Method to check for user input without removing the input from the input queue and pausing your application.

Example

```
WaveLinkIO IOIface = new WaveLinkIO();
WaveLinkTerminal termIface = new WaveLinkTerminal();
string pszVerifyIn;
string pszVerifyInDefault = "";
.
.
.
    try {
        // Do something to instruct the user to enter input.
        // This example uses a call to RFPrint.
       IOIface.RFPrint(0, (termIface.TerminalHeight() - 1),
                 "Verify y/n ? ",
                WaveLinkIO.WLREVERSE | WaveLinkIO.WLCLREOLN);
       pszVerifyIn = IOIface.RFInput(pszVerifyInDefault, 1,
                    0, (termIface.TerminalHeight() + 2), "",
                       WaveLinkIO.WLDISABLESCAN
                       | WaveLinkIO.WLNO_RETURN_BKSP);
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

RFPrint Method

This member of Class WaveLinkIO is supported on all OS types.

The RFPrint Method displays the output text at the specified coordinates using the specified display mode.

Syntax

```
public void RFPrint(int nCol, int nRow, String outputText,
                    long outputMode)
                     throws WaveLinkError,
                            IllegalArgumentException
```

Parameters

nCol	The x-coordinate for the output. Columns are numbered from the left starting with column 0
nRow	The y-coordinate for the output. Rows are numbered from the top starting with row 0
outputText	The data to be displayed
outputMode	The display mode of the output (see Remarks)

Throws

WaveLinkError

InvalidArgumentException

Remarks

Multiple options may be used by "ORing" the desired constants (For example, WLCLEAR | WLNORMAL).

The possible values for *outputMode* are:

WLCLEAR	- Clears the device's display
WLCLREOLN	- Clear text to the end of current line
WLFLUSHOUTPUT	- Automatically flushes the output buffer
WLNORMAL	- Normal display of text
WLREVERSE	- Text is displayed in reverse colors

See the Constant Values appendix for the numeric equates returned by the function.

Example

```
WaveLinkIO ioIface = new WaveLinkIO();
WaveLinkTerminal termIface = new WaveLinkTerminal();
string pszVerifyInput;

    try {
        // Note: Example uses the WaveLinkTerminal object
        // to dynamically position the screen output.
        ioIface.RFPrint(0, 0, "WaveLink Corporation",
                WaveLinkIO.WLCLEAR | WaveLinkIO.WLREVERSE);
        ioIface.RFPrint(0, 1, "   Auto Detailing   ",
                WaveLinkIO.WLNORMAL);
       ioIface.RFPrint(0, (termIface.TerminalHeight() - 1),
                "  Press any key   ", WaveLinkIO.WLNORMAL);
        // Flush the output buffer and await keystroke/scan.
        pszVerifyInput = ioIface.GetEvent();
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

RFSetFill Method

This member of Class WaveLinkIO is supported on all OS types.

The RFSetFill Method sets the default fill character for RFInput calls.

Syntax

```
public void RFSetFill(char newFillChar)
                    throws WaveLinkError
```

Parameters

newfillChar The fill character, passed in single quotation marks

Throws

WaveLinkError

RFSpool Method

This member of Class WaveLinkIO is supported on DOS/embedded, Palm

The RFSpool Method sends the specified data to the printer port with the specified format file, the specified number of times. The format of the data is specified using a previously stored printer format file.

Syntax

```
public void RFSpool(String fileName, int nCopies,
                    String outputData)
                     throws WaveLinkError
```

Parameters

fileName	The local printer format file (see Remarks)
nCopies	The number of copies to make
outputData	The specific data to print

Throws

WaveLinkError

Remarks

NOTE Printer format files are printer-specific. See your printer documentation for more information.

SetInputTimeout Method

This member of Class WaveLinkIO is supported on all OS types.

The SetInputTimeout Method defines the amount of time, in seconds, that elapses before an RFInput input prompt expires.

Syntax

```
public void SetInputTimeout(int numberOSecs)
```

Parameters

numberOSecs The number of seconds before an RFInput prompt expires

Remarks

NOTE By default, RFInput prompts do not expire.

An expired RFInput prompt returns an input type of WLTIMEDOUT. Use the LastInputType Method to check for the returned input type.

TellEvent Method

This member of Class WaveLinkIO is supported on all OS types.

The TellEvent Method requests and returns a single event from the device without removing it from the input queue. Single events are defined as either a key stroke, a function key sequence, or a scanned barcode.

Syntax

```
public String TellEvent()
                    throws WaveLinkError
```

Returns

The returned input event

Throws

WaveLinkError

Remarks

TellEvent *does not* remove any returned input values from the input queue. This method simply checks for user input without pausing your application.

To actually remove returned input from the input queue, a subsequent call to another input method, such as GetEvent Method or RFInput Method must be made following TellEvent. Use a call to the LastInputType Method to return the origin of the last input type (i.e., scan, keypad, or function key input).

NOTE If the local input queue is empty, TellEvent also checks the output queue on the mobile device before returning.

WaitForReconnect Method

This member of Class WaveLinkIO requires a Wavelink Studio 4.x or newer Client.

The WaitForReconnect Method determines if the application connection has been restored within a specified number of seconds after a WaveLinkError occurs (see Remarks).

Syntax

```
public boolean WaitForReconnect(int seconds)
```

Returns

A boolean true or false value that describes whether the device has successfully re-connected to the Wavelink application

Parameters

seconds	The amount of time to wait for the device to reconnect before returning false. The seconds begin counting down from the time that the broken connection is detected

Remarks

A WaveLinkError occurs when the connection to the mobile device is broken. Wavelink applications typically clean up any resources and exit after this type of communication failure. The WaitForReconnect Method is an alternative way to handle a WaveLinkError. WaitForReconnect suspends the Wavelink application until the device can re-establish a connection.

WaitForReconnect works by sending a cookie to the mobile device. This cookie is included in the connect process and used to re-connect a device to an active Wavelink application.

NOTE Special design considerations must be kept in mind for applications that use WaitForReconnect; it is not a transparent function. Care must be taken to ensure that the appropriate screen is redisplayed after a successful reconnect.

Example

```
package reconnect;
import com.wavelink.clientui.*;
public class Reconnect extends WaveLinkApplication {
    private static int CLR = 27;
    public void WaveLinkMain( String arg[] ) {
        WaveLinkIO ioIface = new WaveLinkIO();
        WaveLinkSignon signonObj;
        try {
            // The constructor parameter turns encryption on or
off.
            signonObj = new WaveLinkSignon( true );
        } catch ( WaveLinkError e ) {
            // failed initializing Signon Object
            // exit the WaveLinkApplication
            return;
        }
        // set the text of the WaveLinkSignon cancel button
        signonObj.CancelButton().setDisplayText("Quit");
        int loginFailureCounter = 1;
        int exitCode;
        boolean loggedIn = false;
        // loop until the user logs in or presses cancel
        do {
            try {
                // Defined exitCodes are CANCEL and NORMAL.
                exitCode = signonObj.DisplayDialog();
            } catch ( WaveLinkError e ) {
                try {
                    // Redisplay the signonObj if a terminal
                    // successfully reconnects after a
```

```
communication
                    // failure.
                    if ( ioIface.WaitForReconnect( 60 ) )
                        continue;
                } catch ( Exception err ) {
                }
                // A terminal did not reconnect to this WaveLink
                // application. Exit the WaveLinkApplication
                return;
            }
            // Check to see if CANCEL was selected.
            if ( exitCode == WaveLinkSignon.CANCEL )
                return;
            // Check to see if Login was selected.
            if ( exitCode == signonObj.NORMAL ) {
                // retrieve the userName and Password
                String userName = signonObj.getUserName();
                String password = signonObj.getPassword();
                // compare to the valid userName and Password
                if ( userName != null && password != null
                     && userName.equalsIgnoreCase("a")
                     && password.equalsIgnoreCase( "a") ) {
                    loggedIn = true;
                } else {
                    signonObj.FeedbackLabel().setDisplayText(
                           "Login Incorrect #" +
loginFailureCounter++ );
                }
            }
        }while ( !loggedIn );
        // Display the "End of Demo.  Type "exit"" message and
wait for
        // the user to enter the word "exit".
        String returnData = null;
        do {
            try {
                ioIface.RFPrint( 0, 0, "End of Demo",
                       WaveLinkIO.WLNORMAL | WaveLinkIO.WLCLEAR
);
                ioIface.RFPrint( 0, 1, "Type \"exit\"",
                        WaveLinkIO.WLNORMAL );
                returnData = ioIface.RFInput( "", 4, 0, 3, "",
                             WaveLinkIO.WLNORMALKEYS,
```

```
                                WaveLinkIO.WLBACKLIGHT );
                } catch ( WaveLinkError e ) {
                    try {
                      // Redisplay the RFPrint message if a terminal
                        // successfully reconnects after a
communication
                        // failure.
                        if ( ioIface.WaitForReconnect( 60 ) )
                            continue;
                    } catch ( Exception err ) {
                    }
                   // A terminal did not reconnect to this WaveLink
                    // application. Exit the WaveLinkApplication
                    return;
                }
                // check for the keyed data "exit" then exit the
Wavelink
                // application
            } while ( returnData == null
                      || ioIface.LastInputType() !=
WaveLinkIO.WLKEYTYPE
                      || !returnData.equalsIgnoreCase( "exit" ) );

            // end of WaveLinkApplication
        }
}
```

WaveLinkIO Samples

The following sample code demonstrates the implementation and capabilities of the Class WaveLinkIO and its associated methods.

Java Sample

```
/**
* This source code is for demonstration
* purposes only and should be treated as such.
*
* NOTE: For this demo, we are assuming a 20x8 display area.
*/
package samplecode;
import com.wavelink.clientui.*;
import java.io.*;
public class IOSample extends WaveLinkApplication {
    // Member variables
    WaveLinkIO ioIface;
    WaveLinkBarcode bcIface;
    WaveLinkMessageBox msgIface;
    boolean promptStored;
    RandomAccessFile raf;
    File fileTemp;
    public IOSample() throws WaveLinkError    {
    }
    private void initialize() throws WaveLinkError {
        McastSend.send( "Initializing" );
        // Initialize state variables
        promptStored = false;
        ioIface = new WaveLinkIO ();
        bcIface = new WaveLinkBarcode ();
        msgIface = new WaveLinkMessageBox ();
        // Store allowable barcodes
        bcIface.AddBarcode (WaveLinkBarcode.CODE_39, 7, 10);
        bcIface.AddBarcode (WaveLinkBarcode.UPC_A, 12, 12);
        bcIface.AddBarcode (WaveLinkBarcode.UPC_E0, 6, 6);
        bcIface.AddBarcode (WaveLinkBarcode.UPC_E1, 6, 6);
```

```
        bcIface.StoreBarcode ("IOSAMPLE",
                WaveLinkBarcode.BCDISABLED);
    }
/**
*  DisplayWelcomeScreen - Displays welcome message to the user.
*/
    private void DisplayWelcomeScreen () throws WaveLinkError
    {
        // Send the welcome message to the device ...
        ioIface.RFPrint (0, 0, " Welcome to the ",
                WaveLinkIO.WLCLEAR | WaveLinkIO.WLREVERSE);
        ioIface.RFPrint (0, 1, " WaveLink IO Sample ",
                WaveLinkIO.WLREVERSE);
        ioIface.RFPrint (0, 2, " Application! ",
                WaveLinkIO.WLREVERSE);
        ioIface.RFPrint (0, 7, "Hit A Key To Proceed",
                WaveLinkIO.WLREVERSE);
        // ... and wait for a key stroke.
        ioIface.GetEvent ();
    } // End of DisplayPromptScreen
/**
* DisplayPromptScreen - Prompts the user for input. Note the use
* of PushScreen and RestoreScreen to minimize radio traffic.
*/
    private void DisplayPromptScreen () throws WaveLinkError
    {
        if ( promptStored == false ) {
            ioIface.RFPrint (0, 0, " Enter data to be ",
                    WaveLinkIO.WLCLEAR | WaveLinkIO.WLREVERSE);
            ioIface.RFPrint (0, 1, " evaluated: ",
                    WaveLinkIO.WLREVERSE);
            ioIface.RFPrint (0, 7, " Ctrl+X To Exit ",
                    WaveLinkIO.WLREVERSE
                    |WaveLinkIO.WLFLUSHOUTPUT);
            try {
                ioIface.PushScreen ("IOSAMPLE");
                promptStored = true;
            } // try
            catch ( WaveLinkError wlErr ) {
                if ( wlErr.WLErrorCode () !=
                     WaveLinkError.WLFUNCTIONFAILED )
                     throw wlErr;
```

```
            } // catch (WaveLinkError wlErr)
        } // if (promptStored == false)
        else
            ioIface.RestoreScreen ("IOSAMPLE");
    } // End of DisplayPromptScreen
/**
* EvaluateInput - Reads and processes the user's input by
* displaying a message containing the input's relevant
* information.
*/
    private boolean EvaluateInput () throws WaveLinkError
    {
        String inputData;
        int inputType;
        int barcodeType;
        StringBuffer exitBuf = new StringBuffer ().append
                              ((char)24);
        StringBuffer lineOne = new StringBuffer ();
        StringBuffer lineTwo = new StringBuffer ();
        StringBuffer lineThree = new StringBuffer ();
        inputData = ioIface.RFInput (null, 16, 2, 2, "IOSAMPLE",
                    WaveLinkIO.WLNORMALKEYS,
                    WaveLinkIO.WLBACKLIGHT);
        inputType = ioIface.LastInputType ();
        barcodeType = ioIface.LastExtendedType ();
        switch ( inputType ) {
        case WaveLinkIO.WLCOMMANDTYPE:
           if ( inputData.equals (exitBuf.toString ()) == true )
                return false;
            lineOne.append ("Type: Command");
            lineTwo.append ("Key: ").append (inputData);
             break;
        case WaveLinkIO.WLKEYTYPE:
            lineOne.append ("Type: Normal");
            lineTwo.append ("Key: ").append (inputData);
             break;
        case WaveLinkIO.WLSCANTYPE:
            lineOne.append ("Type: Scanned");
            lineTwo.append (inputData);
```

```
            lineThree.append ("Symbology: ");
            switch ( barcodeType ) {
            case WaveLinkBarcode.CODE_39:
                lineThree.append ("Code 39");
                 break;
            case WaveLinkBarcode.UPC_A:
                lineThree.append ("UPC A");
                 break;
            case WaveLinkBarcode.UPC_E0:
                lineThree.append ("UPC E0");
                 break;
            case WaveLinkBarcode.UPC_E1:
                lineThree.append ("UPC E1");
                 break;
            } // switch (barcodeType)
            break;
        default:
            return false;
        } // switch (inputType)
        msgIface.ClearMessage ();
        msgIface.SetMessageLine (lineOne.toString(), 0);
        msgIface.SetMessageLine (lineTwo.toString(), 1);
        if ( lineThree.length() > 0 )
            msgIface.SetMessageLine (lineThree.toString(), 2);
        msgIface.Display (0);
        return true;
    } // End of EvaluateInput
    public void WaveLinkMain( String arg[] )
    {
        try {
            McastSend.send("Starting application 1" );
            IOSample localObj = new IOSample();
            localObj.initialize();
            localObj.DisplayWelcomeScreen ();
            while ( true ) {
                localObj.DisplayPromptScreen ();
                if ( localObj.EvaluateInput () == false )
                    break;
            } // while (true)
            // Clean up configuration files
            WaveLinkFile flIface = new WaveLinkFile ();
            flIface.RFDeleteFile ("IOSAMPLE.BAR");
            flIface.RFDeleteFile ("IOSAMPLE.SCR");
        }//try
        catch ( WaveLinkError wlErr ) {
        } // catch (WaveLinkError wlErr)
```

```
    McastSend.send("Exiting application") ;
} // End of main}
```

Class WaveLinkMenu

```
com.wavelink.clientui
```

The public class **WaveLinkMenu** is supported on all OS types.

The Class WaveLinkMenu lets you easily create and use menus on a device. WaveLinkMenu objects may also be saved directly into the memory of network mobile devices for true client-server efficiency within your applications.

Fields

WLBORDER	WLNO_BORDER

Constructors

```
public WaveLinkMenu()
                    throws WaveLinkError

public WaveLinkMenu(int bogus)

publicWaveLinkMenu(IWaveLinkSession currentSession)
                    throws WaveLinkError

publicWaveLinkMenu(WaveLinkIO ioIface)
                    throws WaveLinkError
```

Methods

AddOption	MenuFileName
AddTitleLine	MenuHeight
ClearOptions	MenuWidth
ClearTitle	ResetMenu
DeleteMenu	SetCoordinates
DoMenu	SetMenuStyle
GetMenuOption	StartColumn
ListMenuFiles	StartRow
MenuFileCount	StoreMenu

Throws

WaveLinkError

Remarks

A WaveLinkMenu object is comprised of titles and options. A title is a line or lines of text displayed at the top of a menu. A WaveLinkMenu object may

contain more than one title line. Options are the choices available from the menu that may be returned by the menu.

When selected, an option returns the numeric value of its option line. For example, selecting the second option in a three option menu return the value 2 to your application.

Field Detail

WLBORDER

```
public static final long WLBORDER
```

Menu Style - Automatically creates a border around your menu.

WLNO_BORDER

```
public static final long WLNO_BORDER
```

Menu Style - Does not create a border around your menu.

Method Summary

AddOption Method

- Adds a menu option to the WaveLinkMenu object.

AddTitleLine Method

- Adds a single title line to an WaveLinkMenu object.

ClearOptions Method

- Removes all menu options from the WaveLinkMenu object.

ClearTitle Method

- Clears all title lines from the WaveLinkMenu object.

DeleteMenu Method

- Removes menu files from a device.

DoMenu Method

- Executes a menu file from a device's non-volatile memory and returns the selected option to your application.

GetMenuOption Method

- Returns the name of the selected menu option.

ListMenuFiles Method

- Returns the total number of menu files on the device and stores the list of names in the WaveLinkMenu object.

MenuFileCount Method

- Returns the total number of menu files returned by the last call to the ListMenuFiles Method.

MenuFileName Method

- Returns the name of a specific menu file.

MenuHeight Method

- Sets the height of the WaveLinkMenu object.

MenuWidth Method

- Sets the width of the WaveLinkMenu object.

ResetMenu Method

- Resets the current WaveLinkMenu object.

SetCoordinates Method

- Sets the position, height, and width of the WaveLinkMenu object.

SetMenuStyle Method

- Sets the menu style of the WaveLinkMenu object.

StartColumn Method

- Sets the starting left column of the WaveLinkMenu object.

StartRow Method

- Sets the starting top row of the WaveLinkMenu object.

StoreMenu Method

- Saves the current WaveLinkMenu object as a menu file to the device for later use.

AddOption Method

This member of Class WaveLinkMenu is supported on all OS types.

The AddOption Method adds additional options to a WaveLinkMenu object.

Syntax

```
public void AddOption(String optionText)
                    throws IllegalArgumentException
```

Parameters

optionText The text for the menu option

Throws

IllegalArgumentException

Remarks

Options are the choices that may be returned by the menu. The first file in the list is indexed as zero. For example, to return the file name for the third file in the current WaveLinkMenu Object, you pass a 2.

Example

See the DoMenu Method

AddTitleLine Method

This member of Class WaveLinkMenu is supported on all OS types.

The AddTitleLine Method adds a single title line to a WaveLinkMenu object.

Syntax

```
public void AddTitleLine(String titleText)
                    throws IllegalArgumentException
```

Parameters

titleText	Menu title line text

Throws

IllegalArgumentException

Remarks

Titles are always printed at the top of a menu. For multi-lined titles, a WaveLinkMenu object may contain additional title lines.

Example

See the DoMenu Method

ClearOptions Method

This member of Class WaveLinkMenu is supported on all OS types.

The ClearOptions Method clears all options from the WaveLinkMenu object.

Syntax

```
public void ClearOptions()
```

Remarks

ClearOptions removes *all* options from a WaveLinkMenu object.

ClearTitle Method

This member of Class WaveLinkMenu is supported on all OS types.

The ClearTitle Method clears all title lines from the WaveLinkMenu object.

Syntax

```
public void ClearTitle()
```

Remarks

ClearTitle removes *all* titles from a WaveLinkMenu object.

DeleteMenu Method

This member of Class WaveLinkMenu is supported on all OS types.

The DeleteMenu Method removes a menu file from a device.

Syntax

```
public void DeleteMenu(String fileName)
                    throws WaveLinkError,
                           IllegalArgumentException
```

Parameters

fileName The menu file to be deleted

Throws

WaveLinkError

IllegalArgumentException

Remarks

You may pass a file name up to eight characters in length. Pass a wildcard symbol ("*") to delete all menu files from a device.

Example

See WaveLinkMenu Samples

DoMenu Method

This member of Class WaveLinkMenu is supported on all OS types.

The DoMenu Method executes a menu file from a device's non-volatile memory and returns the selected option to your application.

Syntax

```
public int DoMenu(String fileName)
                    throws WaveLinkError,
                           IllegalArgumentException
```

Parameters

fileName	The menu file to be executed (see Remarks)

Returns

The numeric value of the option selected (see Remarks)

Throws

WaveLinkError

IllegalArgumentException

Remarks

You may pass a file name up to eight characters in length.

The DoMenu Method returns -1 if CLR is pressed and -2 if an error is encountered

The DoMenu Method returns the numeric value of the option line selected by the user to your application. For example, if you have a menu with a total of three options and the user selects the second option in the menu, the number 2 is returned to your application.

Example

```
WaveLinkIO ioIface = new WaveLinkIO();
WaveLinkMenu mnuIface = new WaveLinkMenu();
string nTmp;

    try {
        mnuIface.ResetMenu();
        mnuIface.MenuWidth(18);
        mnuIface.AddOption("Option 1");
        mnuIface.AddOption("Option 2");
        mnuIface.AddOption("Option 3");
        mnuIface.AddOption("Option 4");
```

```
            mnuIface.AddTitleLine("Demonstration");
            mnuIface.SetCoordinates(0, 1, 20, 6);
            //The menu must be stored on the device
            // before it may be used.
            mnuIface.StoreMenu("MainMenu");
.
.
.
            nTmp = mnuIface.DoMenu("MainMenu");
        }
        catch (WaveLinkError wlErr) {
            //Do error handling
        }
```

GetMenuOption Method

This member of Class WaveLinkMenu is supported on all OS types.

The GetMenuOption Method returns the option name of a given menu option.

Syntax

```
public String GetMenuOption(int mnuIndex)
```

Parameters

mnuIndex	The numeric value of the option selected (see Remarks)

Returns

The option text

Remarks

Options are the choices that may be returned by the menu. The first file in the list is indexed as zero. For example, to return the file name for the third file in the current WaveLinkMenu Object, you pass a 2.

ListMenuFiles Method

This member of Class WaveLinkMenu is supported on all OS types.

The ListMenuFiles Method returns the total number of menu files saved on a device and stores the list of file names in the current object.

Syntax

```
public int ListMenuFiles()
                    throws WaveLinkError
```

Returns

The menu file count

Throws

WaveLinkError

Remarks

To return the value of the last successful ListMenuFiles Method without making an actual call to the device, use the MenuFileCount Method.

MenuFileCount Method

This member of Class WaveLinkMenu is supported on all OS types.

The MenuFileCount Method returns the total number of menu files successfully returned by the last call to ListMenuFiles.

Syntax.

```
public int MenuFileCount()
```

Returns

The menu file count (see Remarks)

Remarks

The MenuFileCount Method returns the file count from the last successful call to the ListMenuFiles Method. That is, MenuFileCount returns the last known number of files stored on a device without actually communicating via RF to the device.

Use the ListMenuFiles Method to return the total number of menu files that are currently stored on a device.

MenuFileName Method

This member of Class WaveLinkMenu is supported on all OS types.

The MenuFileName Method returns the name of the specified menu file.

Syntax

```
public String MenuFileName(int mnuIndex)
```

Parameters

mnuIndex	The zero-based index value of the file name being returned (see Remarks)

Returns

The menu file name

Remarks

The first file in the list is indexed as zero. For example, to return the file name for the third file in the current WaveLinkMenu Object, you pass a 2.

For a complete listing of all menu files on a device simply loop the MenuFileName Method for the entire number of files returned by MenuFileCount Method.

MenuHeight Method

This member of Class WaveLinkMenu is supported on all OS types.

The MenuHeight Method sets or returns the height of the current menu object.

Syntax

```
public int MenuHeight()
```

```
public void MenuHeight(int newHeight)
```

Parameters

newHeight	The new height value, in number of characters (see Remarks)

Remarks

When defining the coordinates for a menu, keep in mind the limited size of an RF device's display.

If the menu height and width are set to zero or not passed, the Wavelink Studio automatically scales the menu to either the size of the screen or the sum of the number of options and title lines (whichever is smaller).

MenuWidth Method

This member of Class WaveLinkMenu is supported on all OS types.

The MenuWidth Method sets or returns the width of the current menu object.

Syntax

```
public int MenuWidth()
```

```
public void MenuWidth(int newWidth)
```

Parameters

newWidth	The new width value, in number of characters

Remarks

When defining the coordinates for a menu, keep in mind the limited size of an RF device's display.

If the menu height and width are set to zero or not passed, Studio automatically scales the menu to either the size of the screen or the sum of the number of options and title lines (whichever is smaller).

Example

See the DoMenu Method

ResetMenu Method

This member of Class WaveLinkMenu is supported on all OS types.

The ResetMenu Method resets all properties for the current WaveLinkMenu object.

Syntax

```
public void ResetMenu()
```

Remarks

ResetMenu clears all menu options and titles and resets all menu coordinates to zero. This allows you to re-use a menu object instead of creating a new one.

Example

See the DoMenu Method

SetCoordinates Method

This member of Class WaveLinkMenu is supported on all OS types.

The SetCoordinates Method sets the position and size of a WaveLinkMenu object.

Syntax

```
public void SetCoordinates(int mnuCol, int mnuRow,
                           int mnuWidth, int mnuHeight)
                    throws IllegalArgumentException
```

Parameters

mnuCol	The on-screen column coordinate where the top left corner of the menu begins (starting with column 0)
mnuRow	The on-screen row coordinate where the top left corner of the menu begins (starting with row 0)
mnuWidth	The width of the entire menu. Pass a value of 0 to automatically set the width of the menu
mnuHeight	The height of the entire menu. Pass a value of 0 to automatically set the height of the menu

Throws

IllegalArgumentException

Remarks

When defining the coordinates for a menu, keep in mind the limited size of an RF device's display.

If the menu coordinates are set to zero or not passed, the Wavelink Studio automatically scales the menu to either the size of the screen or the sum of the number of options and title lines (whichever is smaller).

Example

See the DoMenu Method

SetMenuStyle Method

This member of Class WaveLinkMenu is supported on all OS types.

The SetMenuStyle Method sets the menu style for the current menu.

Syntax

```
public void SetMenuStyle(long newStyle)
```

Parameters

newStyle	The new style value (see Remarks)

Remarks

The possible values for the menu style are:

WLBORDER	- A border surrounds the menu
WLNO_BORDER	- No menu border

StartColumn Method

This member of Class WaveLinkMenu is supported on all OS types.

The StartColumn Method sets or returns the starting column for the current WaveLinkMenu object.

Syntax

```
public int StartColumn()
```

```
public void StartColumn(int newColumn)
```

Parameters

newColumn The new starting column coordinate (see Remarks)

Returns

The starting column coordinate

Remarks

Columns are numbered from the left, starting with column 0.

StartRow Method

This member of Class WaveLinkMenu is supported on all OS types.

The StartRow Method sets or returns the starting row for the current WaveLinkMenu object.

Syntax

```
public int StartRow()
```

```
public void StartRow(int newRow)
```

Parameters

newRow	The new starting row coordinate (see Remarks)

Returns

The starting row coordinate

Remarks

Rows are numbered from the top, starting with row 0.

StoreMenu Method

This member of Class WaveLinkMenu is supported on all OS types.

The StoreMenu Method saves the current WaveLinkMenu object as a menu file on a device for future use.

Syntax

```
public void StoreMenu(String fileName)
                    throws WaveLinkError,
                           IllegalArgumentException
```

Parameters

fileName	The file in which the WaveLinkMenu object is saved (see Remarks)

Throws

IllegalArgumentException

WaveLinkError

Remarks

You may pass a file name up to eight characters in length.

Example

See the DoMenu Method

WaveLinkMenu Samples

The following sample code demonstrates the implementation and capabilities of the Class WaveLinkMenu and its associated methods.

Java Samples

```
/*************************************************
/*************************************************
* MenuSample.java - This sample describes the usage of the
* WaveLinkMenu, WaveLinkTone, WaveLinkIO, and WaveLinkTerminal
* objects. This source code is for demonstration
* purposes only and should be treated as such.
*
*/
package samplecode;
import com.wavelink.clientui.*;
import java.io.*;
import java.util.*;
public class MenuSample extends WaveLinkApplication {
// Member variables
    WaveLinkIO ioIface;
    WaveLinkMenu mnIface;
    WaveLinkTone tnIface;
    WaveLinkTerminal tmIface;
    public MenuSample () {}
    public void initialize() throws WaveLinkError{
    // Initialize WaveLink objects for use
        ioIface = new WaveLinkIO ();
        mnIface = new WaveLinkMenu ();
        tnIface = new WaveLinkTone ();
        tmIface = new WaveLinkTerminal ();
    // Store menu definition
        mnIface.AddTitleLine (" Menu Sample");
        mnIface.AddTitleLine ("------------------");
        mnIface.AddOption ("Play Music");
        mnIface.AddOption ("Show Corners");
        mnIface.AddOption ("Terminal ID");
        mnIface.AddOption ("WaveLink Version");
        mnIface.AddOption ("Send 100 Pings");
        mnIface.AddOption ("Exit Sample");
        mnIface.StoreMenu ("MNSAMPLE");
```

```
        for ( int lcv = 1; lcv <= 5; ++lcv )
            tnIface.AddTone (lcv * 200, 200);
    // Store tone definition
        tnIface.StoreTone ("MNSAMPLE");
    } // End of initialize
/**
* DisplayWelcomeScreen - Displays a welcome message to the user.
*/
    private void DisplayWelcomeScreen () throws WaveLinkError
    {
// Send the welcome message to the device ...
        ioIface.RFPrint (0, 0, " Welcome to the ",
                 WaveLinkIO.WLCLEAR | WaveLinkIO.WLREVERSE);
        ioIface.RFPrint (0, 1, "WaveLink Menu Sample",
                WaveLinkIO.WLREVERSE);
        ioIface.RFPrint (0, 2, " Application! ",
                WaveLinkIO.WLREVERSE);
        ioIface.RFPrint (0, 7, "Hit A Key To Proceed",
                WaveLinkIO.WLREVERSE);
// ... and wait for a key stroke.
        ioIface.GetEvent ();
    } // End of DisplayWelcomeScreen
/**
* DisplayPromptScreen - Prompts the user for input. Note the use
* of PushScreen and RestoreScreen to minimize radio traffic.
*/
    private int DisplayPromptScreen () throws WaveLinkError
    {
        return mnIface.DoMenu ("MNSAMPLE");
    } // End of DisplayPromptScreen
/**
* EvaluateInput - Processes the user's input by performing the
* functionality specified by the menu option.
*/
    private boolean EvaluateInput (int menuChoice)
                                            throws WaveLinkError
    {
        StringBuffer tmpStr = new StringBuffer ();
        switch ( menuChoice ) {
        case 1: // Play Music
            tnIface.PlayTone ("MNSAMPLE");
             break;
        case 2: // Show Corners
           ioIface.RFPrint (0, 0, "(0,0)",
                   WaveLinkIO.WLREVERSE | WaveLinkIO.WLCLEAR);
           ioIface.RFPrint (0, tmIface.TerminalHeight () - 1,
```

```
                "(0," + Integer.toString() - 1) + ")",
                 WaveLinkIO.WLREVERSE);
            tmpStr.append ("(").append (Integer.toString
                 (tmIface.TerminalWidth () - 1)).append(",0)");
            ioIface.RFPrint (tmIface.TerminalWidth ()
                 - tmpStr.length (), 0, tmpStr.toString (),
                 WaveLinkIO.WLREVERSE);
            tmpStr.setLength (0);
            tmpStr.append ("(").append (Integer.toString
                 (tmIface.TerminalWidth () - 1)).append
                 (",").append (Integer.toString
                 (tmIface.TerminalHeight () - 1)).append(")");
            ioIface.RFPrint (tmIface.TerminalWidth ()
                 - tmpStr.length (), tmIface.TerminalHeight ()
                 - 1, tmpStr.toString (), WaveLinkIO.WLREVERSE);
            ioIface.GetEvent ();
             break;
        case 3: // Terminal ID
            ioIface.RFPrint (0, 0, " Terminal ID:  ",
                 WaveLinkIO.WLREVERSE | WaveLinkIO.WLCLEAR);
            ioIface.RFPrint (0, 1, tmIface.TerminalID (),
                  WaveLinkIO.WLNORMAL);
            ioIface.GetEvent ();
             break;
        case 4: // WaveLink Version
            ioIface.RFPrint (0, 0, " WaveLink Version:  ",
                 WaveLinkIO.WLREVERSE | WaveLinkIO.WLCLEAR);
            ioIface.RFPrint (0, 1, tmIface.WaveLinkVersion (),
                  WaveLinkIO.WLNORMAL);
            ioIface.GetEvent ();
             break;
        case 5: { // 100 Pings
                ioIface.RFPrint (0, 0, "Execute 100 Pings - ",
                      WaveLinkIO.WLREVERSE | WaveLinkIO.WLCLEAR
                        | WaveLinkIO.WLFLUSHOUTPUT);
                Date startTime = new Date ();
                tmIface.Ping ("PING", 100);
                Date endTime = new Date ();
                ioIface.RFPrint(0,2,"TotalTime:  "
                        + String.valueOf(endTime.getTime
                        () - startTime.getTime ()),
                         WaveLinkIO.WLNORMAL);
                ioIface.RFPrint(0,3,"Time Per Ping:  "
                        + String.valueOf((endTime.getTime ()
```

```
                                - startTime.getTime ()) / 100),
                                 WaveLinkIO.WLNORMAL);
                            ioIface.RFPrint (0, 5, "(in milliseconds)",
                                    WaveLinkIO.WLNORMAL);
                            ioIface.GetEvent ();
                             break;
                        } //case 5:
                    case -1: // Ctrl + X
                    case -2: // Error
                    case 6: // Exit
                        return false;
                    } // switch (menuChoice)
                    return true;
                } // End of EvaluateInput
                public void WaveLinkMain( String arg[] ) {
                    try {
                        MenuSample localObj = new MenuSample ();
                        localObj.initialize();
                        localObj.DisplayWelcomeScreen ();
                        while ( true ) {
                            if ( localObj.EvaluateInput
                               (localObj.DisplayPromptScreen()) == false )
                                break;
                        } // while (true)
                        localObj.mnIface.DeleteMenu ("MNSAMPLE");
                        localObj.tnIface.DeleteToneFile ("MNSAMPLE");
                    }//try
                    catch ( WaveLinkError wlErr ) {
                    } // catch (WaveLinkError wlErr)
                }
            }
```

Class WaveLinkMenubarInfo

```
com.wavelink.clientui
```

The public class **WaveLinkMenubarInfo** is supported on: Palm, CE

The Class WaveLinkMenubarInfo contains the methods necessary to create and manipulate a list of menus that can be displayed by a menubar widget.

Methods

Clear	Remove
Insert	Replace

Constructors

```
public WaveLinkMenubarInfo()
```

Remarks

Use the CreateMenubar Method of the WaveLinkFactory object to create a menubar widget.

The menu names used by WaveLinkMenubarInfo methods must match the names of menu configurations created using the Class WaveLinkMenu.

Method Summary

Clear Method

- Clears all menu names from the menu list.

Insert Method

- Inserts a menu name at a specific index in the menu list.

Remove Method

- Removes the menu name from a specific index in the menu list.

Replace Method

- Replaces the menu name at a specific index in the menu list.

Clear Method

This member of Class WaveLinkMenubarInfo is supported on: Palm, CE

The Clear Method clears all menu names from the menu list.

Syntax

```
public void Clear()
```

Insert Method

This member of Class WaveLinkMenubarInfo is supported on: Palm, CE

The Insert Method inserts a menu name at the specified index.

Syntax

```
public void Insert(int inDex, String menuName)
                    throws WaveLinkError
```

Parameters

inDex	The zero-based index for the new menu (see Remarks)
menuName	The name of the WaveLinkMenu configuration

Throws

WaveLinkError

Remarks

The first menu file in the list is indexed as zero. For example, to remove for the third file in the current WaveLinkMenubarInfo object, you pass a 2.

We recommend that you pass an index value of –1 to insert the menu name at the end of the menu list.

Example

```
WaveLinkMenubarInfo infoIface = new WaveLinkMenubarInfo();
WaveLinkMenu mnuIface = new WaveLinkMenu();

    // Create the menus that appear in the menubar.
    mnuIface.ResetMenu();
    mnuIface.AddTitleLine("Widget Demo's");
    mnuIface.AddOption("Buttons");
    mnuIface.AddOption("Pen Capture");
    mnuIface.StoreMenu("MenOne");
    mnuIface.ResetMenu();
    mnuIface.AddTitleLine("Exit");
    mnuIface.AddOption("Quit");
    mnuIface.StoreMenu("MenTwo");
    // Insert the menus into the WaveLinkMenubarInfo object,
    // using a -1 to automatically tack the
    // menu to the end of the list.
    infoIface.Insert(-1, "MenOne");
    infoIface.Insert(-1, "MenTwo");
```

Remove Method

This member of Class WaveLinkMenubarInfo is supported on: Palm, CE

The Remove Method removes the menu name at the specified index.

Syntax

```
public void Remove(int inDex)
                    throws WaveLinkError
```

Parameters

inDex	The zero-based index of the menu to be removed (see Remarks)

Throws

WaveLinkError

Remarks

The first menu file in the list is indexed as zero. For example, to remove for the third file in the current WaveLinkMenubarInfo object, you pass a 2.

Replace Method

This member of Class WaveLinkMenubarInfo is supported on: Palm, CE

The Replace Method replaces the menu name at the specified index.

Syntax

```
public void Replace(int inDex, String menuName)
                    throws WaveLinkError
```

Parameters

inDex	The zero-based index value of the menu to be replaced (see Remarks)
menuName	The name of the new WaveLinkMenu configuration

Throws

WaveLinkError

Remarks

The first menu file in the list is indexed as zero. For example, to remove for the third file in the current WaveLinkMenubarInfo object, you pass a 2.

Class WaveLinkMessageBox

```
com.wavelink.clientui
```

The public class **WaveLinkMessageBox** is supported on all OS types.

The Class WaveLinkMessageBox provides methods for creating, displaying, and destroying error messages on the device.

Constructors

```
public WaveLinkMessageBox()
                            throws WaveLinkError

public WaveLinkMessageBox(IWaveLinkSession
                          currentSession)
                            throws WaveLinkError

public WaveLinkMessageBox(WaveLinkIO ioIface)
                            throws WaveLinkError
```

Methods

ClearMessage Display
SetMessageLine

Remarks

Once defined, an error message remains in the WaveLinkMessageBox object until either the object is released, a new message is defined over an existing line of error message using SetMessageLine, or all message lines are cleared using ClearMessage. This allows you to use a single error object within your application which may be modified for each error message instead of creating a new error object for each individual error message.

Method Summary

ClearMessage Method

- Clears error messages from the WaveLinkMessageBox object.

Display Method

- Displays error messages.

SetMessageLine Method

- Defines error message text at a specific row on the device.

ClearMessage Method

This member of Class WaveLinkMessageBox is supported on all OS types.

The ClearMessage Method clears the list of error lines displayed by the current object.

Syntax

```
public void ClearMessage()
```

Remarks

Once defined, a message remains within a WaveLinkMessageBox object until either the object is released, a new message is defined over an existing line of error message using SetMessageLine Method, or all message lines are cleared using ClearMessage.

This allows you to modify a single object for each message in the application instead of creating a new messagebox object for each individual message.

Example

See the Display Method

Display Method

This member of Class WaveLinkMessageBox is supported on all OS types.

The Display Method sends the error message to the device for display.

Syntax

```
public void Display(int timeOut)
                    throws WaveLinkError

public void Display(int timeOut, boolean showBox)
                    throws WaveLinkError
```

Parameters

timeOut	The number of seconds to display the error
showBox	A boolean true or false value that indicates whether a border displays around the error message

Throws

WaveLinkError

Remarks

For the *timeOut* parameter, pass a value of 0 to prompt the user to acknowledge the error.

For the *showBox* parameter, true value displays a border around the error message; a false value does not display the border.

Example

```
WaveLinkMessageBox messageIface = new WaveLinkMessageBox();
.
.
.
    try {
        messageIface.ClearMessage();
        messageIface.SetMessageLine("  Please have   ", 0);
        messageIface.SetMessageLine(" customer sign  ", 0);
        messageIface.SetMessageLine("Work Order sheet", 0);
        messageIface.Display(0);
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

SetMessageLine Method

This member of Class WaveLinkMessageBox is supported on all OS types.

The SetMessageLine Method adds an error line string to the current error message.

Syntax

```
public void SetMessageLine(String msgLine,
                           int lineNumber)
                    throws WaveLinkError
```

Parameters

msgLine	The display text for the message line
lineNumber	The zero-based index value of the error line to be added (see Remarks)

Throws

WaveLinkError

Remarks

The first message line in the list is indexed as zero. For example, to set the message line as the third line in the current WaveLinkMessageBox object, you pass a 2.

Once defined, a message remains within a WaveLinkMessageBox object until either the object is released, a new message is defined over an existing line of message using the SetMessageLine Method, or all message lines are cleared using ClearMessage Method.

This allows you to modify a single object for each message in the application instead of creating a new messagebox object for each individual message.

Example

See the Display Method

WaveLinkMessageBox Samples

The following sample code demonstrates the implementation and capabilities of the Class WaveLinkMessageBox and its associated methods.

Java Sample

```
package samplecode;

import com.wavelink.clientui.*;
import java.io.*;
public class MessageBoxDemo extends WaveLinkApplication {
    public void WaveLinkMain(String[] args) {
    // This application demonstrates the use of the various
    // WaveLinkMessageBox methods.
    // It assumes a display area of 20x8. In practice, you
    // use a WaveLinkTerminal object to format output to fit
    // the current device.
    // The following lines clear the display and output the app
    // title in reverse video.
        try {
            WaveLinkIO ioIface = new WaveLinkIO();
            WaveLinkMessageBox msgIface = new
                                        WaveLinkMessageBox();
            String strBuffer;
            ioIface.RFPrint(0, 0, " Welcome To The ",
                    WaveLinkIO.WLCLEAR | WaveLinkIO.WLREVERSE);
            ioIface.RFPrint(0, 1, " MessageBox Demo ",
                    WaveLinkIO.WLREVERSE);
            ioIface.RFPrint(0, 3, "Hit a key to proceed",
                    WaveLinkIO.WLNORMAL);
            // Hide the cursor for a cleaner display
            ioIface.RFPrint(0, 20, "", WaveLinkIO.WLNORMAL);
            ioIface.GetEvent();
           // During the course of any application, there are
           // times when you wish to notify a user of an error
          // condition which has arisen. The WaveLinkMessageBox
           // object is the preferred method
           // for doing this over anRF network.
           // The first sample defines and displays an error
           // message for 2 seconds
            msgIface.SetMessageLine(" This error will ", 0);
            msgIface.SetMessageLine(" display for 2 ", 1);
            msgIface.SetMessageLine(" seconds ", 2);
            msgIface.Display(2);
           // Now we define and display an error message, but
           // require the user to acknowledge the error
```

```
                // by pressing the CLR (Escape) key by setting a
                // zero (0) timeout.
                 msgIface.ClearMessage();
                 msgIface.SetMessageLine("Acknowledge this", 1);
                 msgIface.SetMessageLine("error, please. ", 2);
                 msgIface.SetMessageLine(" ", 3);
                 msgIface.Display(0);
                // When an application must perform an extended
                // operation, it is usually a good idea to provide
                // the user with a visual indicator of progress.
                 ioIface.RFPrint(0, 0, " Performing a time- ",
                          WaveLinkIO.WLCLEAR);
                 ioIface.RFPrint(0, 1, " consuming job! ",
                          WaveLinkIO.WLFLUSHOUTPUT);
                 msgIface.ClearMessage();
                 for ( int lcv = 25; lcv < 100; lcv += 25 ) {
                     strBuffer = "" + Integer.toString(lcv)
                                 + "% Complete ";
                     msgIface.SetMessageLine(strBuffer, 0);
                     msgIface.Display(1);
                 }
              }
              catch ( WaveLinkError wlErr ) {
             //Do error handling
              }
        }
}
```

Class WaveLinkScribblePad

```
com.wavelink.clientui
```

The public class **WaveLinkScribblePad** is supported on: Palm, CE

The Class WaveLinkScribblePad contains the properties and methods necessary to display a drawing pad and save the newly created drawing to a specific (.jpg or .bmp) file.

Constructors

```
public WaveLinkScribblePad()
                    throws WaveLinkError
```

Properties

CancelButton	OkButton
ClearButton	Title

Methods

DisplayDialog

Throws

WaveLinkError

Remarks

Use the Class WaveLinkScribblePad for signature capture in your applications.

Property Summary

CancelButton Property

- Stores and retrieves the pre-defined widget that maintains the Cancel button functionality of the scribble pad.

ClearButton Property

- Stores and retrieves the pre-defined widget that maintains the Clear button functionality of the scribble pad.

OkButton Property

- Stores and retrieves the pre-defined widget that maintains the OK button functionality of the scribble pad.

Title Property

- Stores and retrieves the pre-defined widget that maintains the title bar functionality of the scribble pad.

Method Summary

DisplayDialog Method

- Displays the scribble pad dialog box and, when the user presses the OK button, saves the picture.

CancelButton Property

This member of Class WaveLinkScribblePad is supported on: Palm, CE

The CancelButton Property stores and retrieves a pre-defined WaveLinkWidget object for the ScribblePad object. The CancelButton Property maintains the Cancel functionality of the WaveLinkScribblePad object.

Syntax

```
public WaveLinkWidget CancelButton()

public void CancelButton(WaveLinkWidget newCancelButton)
```

Returns

The WaveLinkWidget object

Parameters

newCancelButton The name of the WaveLinkWidget object

Remarks

Use the DisplayText Property of the Class WaveLinkWidget to change the display text that appears on the Cancel button. Currently, changing other WaveLinkWidget properties has no effect.

Example

```
WaveLinkScribblePad scribIface = new WaveLinkScribblePad();
WaveLinkWidget widgetIface;
.
.
.
    widgetIface = scribIface.getCancelButton();
    widgetIface.setDisplayText("Close");
```

ClearButton Property

This member of Class WaveLinkScribblePad is supported on: Palm, CE

The ClearButton Property stores and retrieves a pre-defined WaveLinkWidget object for the WaveLinkScribblePad object. The ClearButton Property maintains the "clear drawing" functionality of the WaveLinkScribblePad object.

Syntax

```
public WaveLinkWidget ClearButton()
```

```
public void ClearButton(WaveLinkWidget newClrButton)
```

Returns

The WaveLinkWidget object

Parameters

newClrButton The name of the WaveLinkWidget object

Remarks

Use the DisplayText Property of the WaveLinkWidget object to change the display text that appears on the Clear button. Currently, changing other widget properties for the Clear button has no effect.

OkButton Property

This member of Class WaveLinkScribblePad is supported on: Palm, CE

The OkButton Property stores and retrieves a pre-defined WaveLinkWidget object for the WaveLinkScribblePad object. The OkButton Property maintains the drawing confirmation functionality of the WaveLinkScribblePad object.

Syntax

```
public WaveLinkWidget OkButton()
```

```
public void OkButton(waveLinkWidget newOkButton)
```

Returns

The WaveLinkWidget object

Parameters

newOkButton The name of the WaveLinkWidget object

Remarks

Use the DisplayText Property of the WaveLinkWidget object to change the display text that appears on the OK button. Currently, changing other properties for the OK button has no effect.

Example

```
WaveLinkScribblePad scribIface = new WaveLinkScribblePad();
WaveLinkWidget widgetIface;
.
.
.
    widgetIface = scribIface.getOkButton();
    widgetIface.setDisplayText("Finish");
```

Title Property

This member of Class WaveLinkScribblePad is supported on: Palm, CE

The Title Property stores and retrieves a pre-defined WaveLinkWidget object for the WaveLinkScribblePad object. The Title Property maintains the title bar display characteristics of the WaveLinkScribblePad object.

Syntax

```
public WaveLinkWidget Title()

public void Title(WaveLinkWidget newTitle)
```

Returns

The WaveLinkWidget object

Parameters

newTitle The name of the WaveLinkWidget object

Remarks

Use the DisplayText Property of the WaveLinkWidget object to change the display text that appears on the title bar. Currently, changing other widget properties for the title bar has no effect.

Example

```
WaveLinkScribblePad scribIface = new WaveLinkScribblePad();
WaveLinkWidget widgetIface;
.
.
.
    widgetIface = scribIface.getTitle();
    widgetIface.setDisplayText("Customer Signature");
```

DisplayDialog Method

This member of Class WaveLinkScribblePad is supported on: Palm, CE

The DisplayDialog Method displays the scribble pad dialog box. When the user presses the OK button, the picture is saved to a .jpg or .bmp file.

Syntax

```
public void DisplayDialog(String trgtFileName)
                    throws WaveLinkError
```

Parameters

trgtFileName	The file in which the new picture is saved, with a .jpg or .bmp extension (see Remarks)

Throws

WaveLinkError

Remarks

If you do not include the full path in the file name, the ScribblePad file is stored in the directory where the application is running.

Example

```
WaveLinkScribblePad scribIface = new WaveLinkScribblePad();
.
.
.
    try {
        scribIface.DisplayDialog("Sig.jpg");
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

Class WaveLinkTerminal

`com.wavelink.clientui`

The public class **WaveLinkTerminal** is supported on all OS types.

The Class WaveLinkTerminal provides current terminal state information and alters certain terminal options.

Fields

CAPSLOCK	MAINBATGOOD
CE2740	MAINBATLOW
CE7200	NORMALKEYS
CE7540	PALM_PILOT
HARDWARECURSOR	PDT
HPC	PDT61XX
HPCPRO	PDT68XX
INTERMEC	PERCON_FALCON
LITHIUMBATDEAD	PPC
LITHIUMBATGOOD	SOFTWARECURSOR
LITHIUMBATNONE	VRC_PRC
LRT	WS10XX

Constructors

```
public WaveLinkTerminal()
                    throws WaveLinkError

public WaveLinkTerminal(IWaveLinkSession currentSession)
                    throws WaveLinkError
```

Methods

Backlight
CursorEnd
CursorMode
CursorStart
DiskSpace
KeyState
KeyTimeout
LithiumBattery
MainBattery
Memory
Ping
RawTerminalType
ReadTerminalInfo
SetDateTime
SetTerminalInfo
SystemCall
TerminalHeight
TerminalID
TerminalType
TerminalWidth
WaveLinkVersion

Throws

WaveLinkError

Remarks

A WaveLinkTerminal object contains the terminal settings. To improve network efficiency, these are only set within the WaveLinkTerminal object when the object is first created.

To return the most current terminal settings you must first use the ReadTerminalInfo Method, which updates the WaveLinkTerminal object with the latest device settings.

Changes to device settings *do not* occur until you call the SetTerminalInfo Method.

Field Detail

CAPSLOCK

```
public static final int CAPSLOCK
```

KeyState Keypad State - Represents a device keyboard state of capslock.

CE2740

```
public static final int CE2740
```

RF Terminal Type - Represents 2740 Windows CE-based mobile devices.

CE7200

```
public static final int CE7200
```

RF Terminal Type - Represents 7200 Windows CE-based mobile devices.

CE7540

```
public static final int CE7540
```

RF Terminal Type - Represents 7540 Windows CE-based mobile devices.

HARDWARECURSOR

```
public static final int HARDWARECURSOR
```

CursorMode Cursor State - Represents a hardware cursor state on the device.

HPC

```
public static final int HPC
```

RF Terminal Type - Represents HPC mobile devices.

HPCPRO

```
public static final int HPCPRO
```

RF Terminal Type - Represents HPC Pro mobile devices.

INTERMEC

```
public static final int INTERMEC
```

RF Terminal Type - Represents Intermec mobile devices.

LITHIUMBATDEAD

```
public static final int LITHIUMBATDEAD
```

LithiumBattery Battery State - RF Device lithium battery is bad.

LITHIUMBATGOOD

```
public static final int LITHIUMBATGOOD
```

LithiumBattery Battery State - RF Device lithium battery is good.

LITHIUMBATNONE

```
public static final int LITHIUMBATNONE
```

LithiumBattery Battery State - RF Device lithium battery is not present.

LRT

```
public static final int LRT
```

RF Terminal Type - Represents the LRT series terminals.

MAINBATGOOD

```
public static final int MAINBATGOOD
```

MainBattery Battery State - RF Device main battery is good.

MAINBATLOW

```
public static final int MAINBATLOW
```

MainBattery Battery State - RF Device main battery is low.

NORMALKEYS

```
public static final int NORMALKEYS
```

KeyState Keypad State - This constant represents a device keyboard state of capsoff.

PALM_PILOT

```
public static final int PALM_PILOT
```

RF Terminal Type - This constant represents Palm Pilot devices.

PDT

```
public static final int PDT
```

RF Terminal Type - This constant represents the PDT series terminals.

PDT61XX

```
public static final int PDT61XX
```

RF Terminal Type - This constant represents the PDT61XX series terminals.

PDT68XX

```
public static final int PDT68XX
```

RF Terminal Type - This constant represents the PDT68XX series mobile devices.

PERCON_FALCON

```
public static final int PERCON_FALCON
```

RF Terminal Type - This constant represents PSC Falcon devices.

PPC

```
public static final int PPC
```

RF Terminal Type - This constant represents PocketPC-based mobile devices.

SOFTWARECURSOR

```
public static final int SOFTWARECURSOR
```

CursorMode Cursor State - This constant represents a software cursor state on the device.

VRC_PRC

```
public static final int VRC_PRC
```

RF Terminal Type - This constant represents the VRC/PRC series mobile devices.

WINDOWS

```
public static final int WINDOWS
```

RF Terminal Type - This constant represents Windows-based devices.

WS10XX

```
public static final int WS10XX
```

RF Terminal Type - This constant represents the WS10XX series terminals.

Method Summary

Backlight Method

- Sets or returns the amount of time that passes before a device turns off the display backlight.

CursorEnd Method

- Sets or returns the last line of device display that is valid for printing.

CursorMode Method

- Sets or returns the current cursor type of the device.

CursorStart Method

- Sets or returns the first line of device display that is valid for printing.

DiskSpace Method

- Returns the total storage capacity of the device.

KeyState Method

- Sets or returns the state of the device's keypad.

KeyTimeout Method

- Sets or returns the amount of inactive time that passes before a device automatically powers down.

LithiumBattery Method

- Returns the state of the device's lithium battery.

MainBattery Method

- Returns the state of the device's main battery.

Memory Method

- Returns the total memory capacity of the device.

Ping Method

- Times a send and receive cycle between the RF application and the device.

RawTerminalType Method

- Returns the type of the current device.

ReadTerminalInfo Method

- Updates the RFTerminal object with the current device settings.

SetDateTime Method

- Synchronizes the date and time on the device with the host time and date.

SetTerminalInfo Method

- Applies altered device settings to the device.

SystemCall Method

- Executes common DOS commands on the device.

TerminalHeight Method

- Returns the height of the device's display.

TerminalID Method

- Returns the unique terminal ID of the device.

TerminalType Method

- Returns a string specifying the current device type.

TerminalWidth Method

- Returns the width of the device's display.

WaveLinkVersion Method

- Returns the version of the device's Wavelink Studio Client software.

Backlight Method

This member of Class WaveLinkTerminal is supported on: DOS/embedded, Palm

The Backlight Method sets or returns the inactivity time for the backlight of the device.

Syntax

```
public int Backlight()
```

```
public void Backlight(int newBacklight)
```

Parameters

newBacklight	Number of seconds of inactivity before the display backlight turns off

Returns

The backlight timeout value for the current device

Remarks

To improve network efficiency, the terminal settings within the WaveLinkTerminal object are only set when the object is first created.

To return the most current information you must first use the ReadTerminalInfo Method, which updates the WaveLinkTerminal object with the latest device settings, before making a call to any other methods.

NOTE If the backlight timeout value is more than the key timeout value, the device completely powers down before the backlight is ever deactivated.

CursorEnd Method

This member of Class WaveLinkTerminal is supported on: DOS/embedded, Palm

The CursorEnd Method sets or returns the last row that may contain the cursor on the current device.

Syntax

```
public int CursorEnd()

public void CursorEnd(int newEnd)
```

Parameters

newEnd	The device row (Row numbers that follow this number cannot be used to display text)

Returns

The last row that may contain a cursor on the current device

Remarks

To improve network efficiency, the terminal settings within the WaveLinkTerminal object are only set when the object is first created. To return the most current information you must first use the ReadTerminalInfo Method, which updates the WaveLinkTerminal object with the latest device settings, before making a call to any other methods.

CursorMode Method

This member of Class WaveLinkTerminal is supported on: DOS/embedded, Palm

The CursorMode Method sets or returns the current device's cursor mode.

Syntax

```
public int CursorMode()

public void CursorMode(int newMode)
```

Parameters

newMode	The new cursor mode (see Remarks)

Returns

The cursor mode of the current device

Remarks

The possible values are:

HARDWARECURSOR	- block cursor
SOFTWARECURSOR	- caret cursor

To improve network efficiency, the terminal settings within the WaveLinkTerminal object are only set when the object is first created. To return the most current information you must first use the ReadTerminalInfo Method, which updates the WaveLinkTerminal object with the latest device settings, before making a call to any other methods.

Example

See the WaveLinkTerminal Samples

CursorStart Method

This member of Class WaveLinkTerminal is supported on: DOS/embedded, Palm

The CursorStart Method sets or returns the first row that may contain the cursor on the current device.

Syntax

```
public int CursorStart()
```

```
public void CursorStart(int newStart)
```

Parameters

newStart	The device row. Row numbers that precede this number cannot be used to display text.

Returns

The first row that may contain a cursor on the current device

Remarks

To improve network efficiency, the terminal settings within the WaveLinkTerminal object are only set when the object is first created. To return the most current information you must first use the ReadTerminalInfo Method, which updates the WaveLinkTerminal object with the latest device settings, before making a call to any other methods.

Example

See WaveLinkTerminal Samples

DiskSpace Method

This member of Class WaveLinkTerminal is supported on: DOS/embedded, Palm

The DiskSpace Method returns the total amount of diskspace available on the current device. This method *does not* return the amount of storage space currently available.

Syntax

```
public long DiskSpace()
```

Returns

The total amount of available disk space on the current device

Remarks

To improve network efficiency, the terminal settings within the WaveLinkTerminal object are only set when the object is first created. To return the most current information you must first use the ReadTerminalInfo Method, which updates the WaveLinkTerminal object with the latest device settings, before making a call to any other methods.

KeyState Method

This member of Class WaveLinkTerminal is supported on: DOS/embedded, Palm

The KeyState Method sets or returns the current device's keyboard state.

Syntax

```
public int KeyState()
```

```
public void KeyState(int newKeyState)
```

Parameters

newKeyState The new keyboard state (see Remarks)

Remarks

The possible values for *newKeyState* are:

WLCAPSLOCK	- Caps lock on
WLNORMALKEYS	- Caps lock off

To improve network efficiency, the terminal settings within the WaveLinkTerminal object are only set when the object is first created. To return the most current information you must first use the ReadTerminalInfo Method, which updates the WaveLinkTerminal object with the latest device settings, before making a call to any other methods.

KeyTimeout Method

This member of Class WaveLinkTerminal is supported on all OS types.

The KeyTimeout Method sets or returns the current device's keyboard timeout value.

Syntax

```
public int KeyTimeout()

public void KeyTimeout(int newKeyTimeout)
```

Parameters

newKeyTimeout	The number of seconds of inactivity that may pass before a device automatically powers down

Remarks

To improve network efficiency, the terminal settings within the WaveLinkTerminal object are only set when the object is first created. To return the most current information you must first use the ReadTerminalInfo Method, which updates the WaveLinkTerminal object with the latest device settings, before making a call to any other methods.

LithiumBattery Method

This member of Class WaveLinkTerminal is supported on: DOS/embedded, Palm

The LithiumBattery Method returns the state of the current device's Lithium battery.

Syntax

```
public int LithiumBattery()
```

Returns

The state of the Lithium battery (see Remarks)

Remarks

The possible Lithium battery states are:

LITHIUMBATDEAD	- Weak charge
LITHIUMBATGOOD	- Strong charge
LITHIUMBATNONE	- No battery present

To improve network efficiency, the terminal settings within the WaveLinkTerminal object are only set when the object is first created. To return the most current information you must first use the ReadTerminalInfo Method, which updates the WaveLinkTerminal object with the latest device settings, before making a call to any other methods.

MainBattery Method

This member of Class WaveLinkTerminal is supported on: DOS/embedded, Palm

The MainBattery Method returns the state of the current device's main battery.

Syntax

```
public int MainBattery()
```

Returns

The state of the main battery (see Remarks)

Remarks

The possible main battery states are:

MAINBATGOOD	- Strong charge
MAINBATLOW	- Weak charge

To improve network efficiency, the terminal settings within the WaveLinkTerminal object are only set when the object is first created. To return the most current information you must first use the ReadTerminalInfo Method, which updates the WaveLinkTerminal object with the latest device settings, before making a call to any other methods.

Memory Method

This member of Class WaveLinkTerminal is supported on: DOS/embedded, Palm

The Memory Method returns the total amount of memory on the current device. This method returns the total *memory capacity* and *not* the total memory available.

Syntax

```
public long Memory()
```

Returns

The total memory capacity

Remarks

To improve network efficiency, the terminal settings within the WaveLinkTerminal object are only set when the object is first created. To return the most current information you must first use the ReadTerminalInfo Method, which updates the WaveLinkTerminal object with the latest device settings, before making a call to any other methods.

Ping Method

This member of Class WaveLinkTerminal is supported on all OS types.

The Ping Method sends to and receives data packets from the device the specified number of times.

Syntax

```
public long Ping(String pingPacket, long pingCount)
                    throws WaveLinkError
```

Parameters

pingPacket	The data packet to send/receive
pingCount	The number of cycles to execute

Returns

The number of milliseconds required for method execution

Throws

WaveLinkError

RawTerminalType Method

This member of Class WaveLinkTerminal is supported on all OS types.

The RawTerminalType Method returns the raw device type of the mobile device as an integer.

Syntax

```
public int RawTerminalType()
```

Returns

Device terminal type (see Remarks)

Remarks

The possible device types are:

LRT	- LRT terminals
VRC_PRC	- VRC_PRC terminals
PDT	- PDT terminals
WS10XX	- WS 10XX terminals
PDT68XX	- PDT 68XX terminals
PDT61XX	- PDT 61XX terminals
PALM_PILOT	- Palm Pilot device
CE2740	- 2740 CE-based device
CE7200	- 7200 CE-based device
CE7540	- 7540 CE-based device
PERCON_FALCON	- PSC Falcon
INTERMEC	- Intermec device
WINDOWS	- Windows device
PPC	- PPC device
HPC	- HPC device
HPCPRO	- HPC Pro device

See the Constant Values in the Appendix for the numeric values of the functions. The appendix also contains numeric constants for additional mobile devices.

ReadTerminalInfo Method

This member of Class WaveLinkTerminal is supported on all OS types (see Remarks).

The ReadTerminalInfo Method queries the current device for its terminal information. This is done automatically at object creation and may be repeated to get accurate values for the non-static fields.

Syntax

```
public void ReadTerminalInfo()
                    throws WaveLinkError
```

Throws

WaveLinkError

Remarks

To update any changes you have made to a device settings, use the SetTerminalInfo Method.

NOTE The ReadTerminalInfo Method is supported on CE *but* the terminal information returned may or may not reflect the actual parameters on the device. The values for several parameters are hard-coded values that are returned by all CE devices.

Example

```
WaveLinkTerminal termIface = new WaveLinkTerminal();
    try {
        // Request the current info from the device.
        TermIface.ReadTerminalInfo();
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

SetDateTime Method

This member of Class WaveLinkTerminal is supported on: DOS/embedded, Palm

The SetDateTime Method synchronizes the date and time on the device with that of the host by sending the current host date and time to the device.

Syntax

```
public void SetDateTime()
                    throws WaveLinkError
```

Throws

WaveLinkError

SetTerminalInfo Method

This member of Class WaveLinkTerminal is supported on: DOS/embedded, Palm

The SetTerminalInfo Method updates the device's non-static fields of terminal information with the user specified values.

Syntax

```
public void SetTerminalInfo()
                    throws WaveLinkError
```

Throws

WaveLinkError

Remarks

To improve network efficiency, the terminal settings within the WaveLinkTerminal object are only set when the object is first created. To return the most current information you must first use the ReadTerminalInfo Method, which updates the WaveLinkTerminal object with the latest device settings, before making a call to any other methods.

For these same network efficiency reasons changes to device settings *are not* actually applied until you call the SetTerminalInfo Method.

Example

See the WaveLinkTerminal Samples

SystemCall Method

This member of Class WaveLinkTerminal is supported on: DOS/embedded, Palm

The SystemCall Method executes common DOS commands and functions on a device.

Syntax

```
public void SystemCall(String sysCall)
                    throws WaveLinkError
```

Parameters

sysCall	The specific DOS command. For example, for a directory listing of the device, pass "DIR"

Throws

WaveLinkError

TerminalHeight Method

This member of Class WaveLinkTerminal is supported on all OS types.

The TerminalHeight Method returns the height, in rows, of the current device display.

Syntax

```
public int TerminalHeight()
```

Returns

The height of the display, in rows, for the current device

Remarks

To improve network efficiency, the terminal settings within the WaveLinkTerminal object are only set when the object is first created. To return the most current information you must first use the ReadTerminalInfo Method, which updates the WaveLinkTerminal object with the latest device settings, before making a call to any other methods.

Example

```
WaveLinkIO ioIface = new WaveLinkIO();
WaveLinkTerminal termIface = new WaveLinkTerminal();
int termWidth;
int termHeight;
string pszVerifyInput;
// Note: If you need to obtain current terminal
// info, use the ReadTerminalInfo Method (not shown).
termWidth = termIface.TerminalWidth();
termHeight = termIface.TerminalHeight();
    try {
        // In this example, use the termWidth and
        // termHeight variables RFPrint Method calls.
        ioIface.RFPrint(0, (termHeight -1), "Correct y/n?",
                WaveLinkIO.WLNORMAL);
        // Hide the cursor for a cleaner display using
        // termWidth or termHeight.
        ioIface.RFPrint((termWidth + 1), 0
                "", WaveLinkIO.WLNORMAL);
         // Flush the output buffer and await a
         // keystroke/scan using either RFInput or GetEvent.
        pszVerifyInput = ioIface.GetEvent();
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

TerminalID Method

This member of Class WaveLinkTerminal is supported on all OS types.

The TerminalID Method returns the current device's ID.

Syntax

```
public String TerminalID()
```

Returns

The device's ID string

Remarks

NOTE For identification purposes, each device on a Wavelink network is assigned a unique terminal ID.

To improve network efficiency, the terminal settings within the WaveLinkTerminal object are only set when the object is first created. To return the most current information you must first use the ReadTerminalInfo Method, which updates the WaveLinkTerminal object with the latest device settings, before making a call to any other methods.

NOTE On Spectrum24 networks, the terminal ID is the device's IP address.

Example

See the WaveLinkTerminal Samples

TerminalType Method

This member of Class WaveLinkTerminal is supported on all OS types.

The TerminalType Method returns a string specifying the type of device currently connected.

Syntax

```
public String TerminalType()
```

Returns

Current device type (see Remarks)

Remarks

To improve network efficiency, the terminal settings within the WaveLinkTerminal object are only set when the object is first created. To return the most current information you must first use the ReadTerminalInfo Method, which updates the WaveLinkTerminal object with the latest device settings, before making a call to any other methods.

The possible device types are:

LRT

PALM_PILOT

PDT

PDT61XX

PDT68XX

PERCON_FALCON

VRC_PRC

WS10XX

TerminalWidth Method

This member of Class WaveLinkTerminal is supported on all OS types.

The TerminalWidth Method returns the width, in characters, of the current device display.

Syntax

```
public int TerminalWidth()
```

Returns

The display height, in columns, for the current device

Remarks

To improve network efficiency, the terminal settings within the WaveLinkTerminal object are only set when the object is first created. To return the most current information you must first use the ReadTerminalInfo Method, which updates the WaveLinkTerminal object with the latest device settings, before making a call to any other methods.

Example

See the TerminalHeight Method

WaveLinkVersion Method

This member of Class WaveLinkTerminal is supported on all OS types.

The WaveLinkVersion Method returns the version number of the Wavelink Studio Client on the current device.

Syntax

```
public String WaveLinkVersion()
```

Returns

The version number of the Wavelink Studio Client on the current device

Remarks

To improve network efficiency, the terminal settings within the WaveLinkTerminal object are only set when the object is first created. To return the most current information you must first use the ReadTerminalInfo Method, which updates the WaveLinkTerminal object with the latest device settings, before making a call to any other methods.

WaveLinkTerminal Samples

The following sample code demonstrates the implementation and capabilities of the Class WaveLinkTerminal and its associated methods.

Java Sample

```
/**
* This source code is for demonstration
* purposes only and should be treated as such.
*
* This application demonstrates the use of the various
* WaveLinkTerminal methods.
*
*/
package samplecode;
import com.wavelink.clientui.*;
import java.io.*;
public class JavaTerminalDemo extends WaveLinkApplication {
    public void WaveLinkMain(String[] args)
    {
        WaveLinkIO ioIface = new WaveLinkIO();
        String strBuffer;
        // The following lines clear the display and output the
        // app title in reverse video.
        try {
            WaveLinkTerminal termIface = new WaveLinkTerminal();
            ioIface.RFPrint(0, 0, " Welcome To The ",
                            WaveLinkIO.WLCLEAR
                             | WaveLinkIO.WLREVERSE);
            ioIface.RFPrint(0, 1, " Java Terminal Demo ",
                             WaveLinkIO.WLREVERSE);
            ioIface.RFPrint(0, 3, "Hit a key to proceed",
                             WaveLinkIO.WLNORMAL);
            //Hide the cursor for a cleaner display
            ioIface.RFPrint(0, 20, "", WaveLinkIO.WLNORMAL);
            ioIface.GetEvent();
            // When designing RF applications, it is often not
            // possible to know which types of devices are
            // going to be used by the end user. The
            // WaveLinkTerminal object accesses
            // accurate information about the user's device.
            // Simply creating a WaveLinkTerminal object
            // acquires the terminal's information, but for
```

```
// demonstration purposes, we explicitly
// request the current setting from the device.
termIface.ReadTerminalInfo();
ioIface.RFPrint(0, 0, "Terminal information",
                WaveLinkIO.WLREVERSE
                 | WaveLinkIO.WLCLEAR);
//Display the terminal's screen dimensions
strBuffer = "Screen Size ("
      + Integer.toString(termIface.TerminalWidth())
      + ", "
      + String.valueOf(termIface.TerminalHeight())
      + ")";
```

```
                    ioIface.RFPrint(0, 1, strBuffer,
                                    WaveLinkIO.WLNORMAL);
                    //Display the WaveLink Client version
                    ioIface.RFPrint(0, 3, "WaveLink Version:",
                                    WaveLinkIO.WLNORMAL);
                    ioIface.RFPrint(0, 4, termIface.WaveLinkVersion(),
                                    WaveLinkIO.WLNORMAL);
                    //Display the Terminal's ID
                    ioIface.RFPrint(0, 6, "Terminal ID:",
                                    WaveLinkIO.WLNORMAL);
                    ioIface.RFPrint(0, 7, termIface.TerminalID(),
                                    WaveLinkIO.WLNORMAL);
                    ioIface.RFPrint(0, 20, "", WaveLinkIO.WLNORMAL);
                    ioIface.GetEvent();
                    ioIface.RFPrint(0, 0, " Setting Cursor To ",
                                   WaveLinkIO.WLCLEAR
                                    | WaveLinkIO.WLREVERSE);
                    ioIface.RFPrint(0, 1, " Software Mode ",
                                   WaveLinkIO.WLREVERSE);
                    ioIface.RFPrint(0, 3, "SW Curser -> ",
                                   WaveLinkIO.WLFLUSHOUTPUT);

                    termIface.CursorMode
                                     (WaveLinkTerminal.SOFTWARECURSOR);
                    termIface.SetTerminalInfo();
                    ioIface.GetEvent();
                    ioIface.RFPrint(0, 0, " Resetting Cursor To ",
                                   WaveLinkIO.WLCLEAR
                                    | WaveLinkIO.WLREVERSE);
                    ioIface.RFPrint(0, 1, " Hardware Mode ",
                                   WaveLinkIO.WLREVERSE
                                    | WaveLinkIO.WLFLUSHOUTPUT);
                    ioIface.RFPrint(0, 3, "HW Curser -> ",
                                    WaveLinkIO.WLFLUSHOUTPUT);
                    termIface.CursorMode
                                     (WaveLinkTerminal.HARDWARECURSOR);
                    termIface.SetTerminalInfo();
                    ioIface.GetEvent();
                }
                catch ( WaveLinkError wlErr ) {
                //Do error handling
                }
            }
        }
```

Class WaveLinkTone

```
com.wavelink.clientui
```

The public class **WaveLinkTone** is supported on all OS types.

The Class WaveLinkTone provides the interface for creating tone configurations for the device.

Constructors

```
public WaveLinkTone()
```

Methods

AddTone	PlayTone
ClearTones	RemoveTone
DeleteToneFile	StoreTone
Duration	ToneCount
Frequency	ToneFileCount
GetToneFile	ToneFileName
ListToneFiles	

Remarks

Individual notes are stored within a WaveLinkTone object, and each note is defined by a frequency and a duration. Once defined, a specific note remains within a WaveLinkTone object until either the object is released or all notes are cleared from the object using ClearTones Method. This allows you to use a single tone object within your application which may be modified instead of creating a new tone object for each use.

Method Summary

AddTone Method

- Adds a specific note to the WaveLinkTone object.

ClearTones Method

- Clears the notes from the WaveLinkTone object.

DeleteToneFile Method

- Deletes tone files from the device.

Duration Method

- Sets or returns the duration of a specific note in the WaveLinkTone object.

Frequency Method

- Sets or returns the frequency of a specific note in the WaveLinkTone object.

GetToneFile Method

- Returns the a tone file into the current WaveLinkTone object.

ListToneFiles Method

- Returns the number of tone files on the device and stores the list of names in the WaveLinkTone object.

PlayTone Method

- Plays a tone file.

RemoveTone Method

- Removes a specific note from the current WaveLinkTone object.

StoreTone Method

- Stores the WaveLinkTone object as a tone file on the device for future use.

ToneCount Method

- Returns the number of notes within the WaveLinkTone object.

ToneFileCount Method

- Returns the number of tone files returned by the last successful call to the ListToneFiles Method.

ToneFileName Method

- Returns the file name of a saved tone file.

AddTone Method

This member of Class WaveLinkTone is supported on all OS types.

The AddTone Method adds a new note to the current tone configuration.

Syntax

```
public void AddTone(int newFrequency, int newDuration)
                    throws IllegalArgumentException
```

Parameters

newFrequency	The frequency, in hertz, of the new note
newDuration	The duration, in milliseconds, of the new note

Throws

IllegalArgumentException

Remarks

Individual notes are stored within a WaveLinkTone object, each note defined by a frequency and a duration. Once defined, a specific note remains within a WaveLinkTone object until either the object is released or all notes are cleared from the object using ClearTones Method. This allows you to re-use a single tone object instead of creating a new tone object for each use.

Example

See the PlayTone Method

ClearTones Method

This member of Class WaveLinkTone is supported on all OS types.

The ClearTones Method removes all notes from the current tone configuration.

Syntax

```
public void ClearTones()
```

Remarks

Individual notes are stored within a WaveLinkTone object, each note defined by a frequency and a duration. Once defined, a specific note remains within a WaveLinkTone object until either the object is released or all notes are cleared from the object using ClearTones Method. This allows you to re-use a single tone object instead of creating a new tone object for each use.

Example

See the PlayTone Method

DeleteToneFile Method

This member of Class WaveLinkTone is supported on all OS types.

The DeleteToneFile Method removes tone files from a device's non-volatile memory.

Syntax

```
public void DeleteToneFile(String fileName)
                    throws WaveLinkError,
                           IllegalArgumentException
```

Parameters

fileName	The remote tone configuration file

Throws

IllegalArgumentException

WaveLinkError

Remarks

You may pass a file name up to eight characters in length. Enter a wildcard symbol ("*") to delete all tone files.

Example

See the WaveLinkTone Samples

Duration Method

This member of Class WaveLinkTone is supported on all OS types.

The Duration Method sets or returns the duration of the specified note.

Syntax

```
public int Duration(int tnIndex)

public void Duration(int tnIndex, int newDuration)
                    throws IllegalArgumentException
```

Parameters

tnIndex	The zero-based index value of the note in the current configuration (see Remarks)
newDuration	The duration, in milliseconds, of the note

Returns

The duration of the note in milliseconds

Throws

IllegalArgumentException

Remarks

To alter the duration for a note stored within a tone file on a device, first use the GetToneFile Method to set the tone file as the current tone object.

The first tone in the list is indexed as zero. For example, to return the duration for the third note in the current WaveLinkTone Object, you pass a 2.

Frequency Method

This member of Class WaveLinkTone is supported on all OS types.

The Frequency Method sets or returns the frequency of the specified note.

Syntax

```
public int Frequency(int tnIndex)

public void Frequency(int tnIndex, int newFrequency)
                    throws IllegalArgumentException
```

Parameters

tnIndex	The zero-based index value of the note in the current configuration (see Remarks)
newFrequency	The frequency, in hertz, of the note

Throws

IllegalArgumentException

Remarks

To alter a frequency for a note stored within a tone file on a device, first use the GetToneFile Method to set the tone file as the current tone object.

The first tone in the list is indexed as zero. For example, to return the frequency for the third note in the current WaveLinkTone Object, you pass a 2.

GetToneFile Method

This member of Class WaveLinkTone is supported on all OS types.

The GetToneFile Method retrieves a tone file from a device into the current WaveLinkTone object.

Syntax

```
public int GetToneFile(String fileName)
                    throws WaveLinkError,
                           IllegalArgumentException
```

Parameters

fileName	The tone configuration file (see Remarks)

Throws

IllegalArgumentException

WaveLinkError

Remarks

You may pass a file name up to eight characters in length.

ListToneFiles Method

This member of Class WaveLinkTone is supported on all OS types.

The ListToneFiles Method returns the total number of tone files saved on a device and stores a list of file names in the current object.

Syntax

```
public int ListToneFiles()
                    throws WaveLinkError
```

Returns

The number of tone files saved on the current device

Throws

WaveLinkError

Remarks

To return the value of the last successful ListToneFiles Method without making an actual call to the device, use the ToneFileCount Method. To list the total number of notes within a single tone file, use the ToneCount Method.

PlayTone Method

This member of Class WaveLinkTone is supported on all OS types.

The PlayTone Method plays a tone file saved on a device.

Syntax

```
public void PlayTone(String fileName)
                    throws WaveLinkError,
                           IllegalArgumentException
```

Parameters

fileName The tone configuration file

Throws

IllegalArgumentException

WaveLinkError

Remarks

WaveLinkTone objects must be saved as tone files on the device with the StoreTone Method *before* they may be played.

Example

```
WaveLinkTone toneIface = new WaveLinkTone();
    try {
        toneIface.ClearTones();
        toneIface.AddTone(440, 200);
        toneIface.AddTone(440, 200);
        toneIface.StoreTone("ErrorBeep");
.
.
.
        // Play the tone
        toneIface.PlayTone("ErrorBeep");
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

RemoveTone Method

This member of Class WaveLinkTone is supported on all OS types.

The RemoveTone Method removes a specific note from the current WaveLinkTone object.

Syntax

```
public void RemoveTone(int tnIndex)
```

Parameters

tnIndex	The zero-based index value of the note to be removed (see Remarks)

Remarks

To remove a note from a tone file stored on a device, you must first set the tone file as the current WaveLinkTone object using the GetToneFile Method, then use RemoveTone.

The first tone in the list is indexed as zero. For example, to remove the third note in the current WaveLinkTone Object, you pass a 2.

StoreTone Method

This member of Class WaveLinkTone is supported on all OS types.

The StoreTone Method stores the current tone configuration on the device.

Syntax

```
public void StoreTone(String fileName)
                    throws WaveLinkError,
                           IllegalArgumentException
```

Parameters

fileName	The remote tone configuration file (see Remarks)

Throws

IllegalArgumentException

WaveLinkError

Remarks

This tone file name may be up to eight characters in length.

WaveLinkTone objects must be saved as tone files on a device using the StoreTone *before* they may be played. Use the PlayTone Method to play a tone.

Remarks

See the PlayTone Method

ToneCount Method

This member of Class WaveLinkTone is supported on all OS types.

The ToneCount Method returns the number of individual notes within a WaveLinkTone object.

Syntax

```
public int ToneCount()
```

Returns

The number of notes in the WaveLinkTone object

Remarks

To count the number of notes in a tone file saved to the current device, set the tone file as the current WaveLinkTone object using GetToneFile Method, then use the ToneCount Method.

ToneFileCount Method

This member of Class WaveLinkTone is supported on all OS types.

The ToneFileCount Method returns the total number of tone files returned by the last successful call to the ListToneFiles Method.

Syntax

```
public int ToneFileCount()
                    throws WaveLinkError
```

Returns

The tone file count

Throws

WaveLinkError

Remarks

The ToneFileCount Method returns the file count from the last successful call to the ListToneFiles Method. Use the ListToneFiles Method to return the total number of tone files that are currently stored on a device.

ToneFileName Method

This member of Class WaveLinkTone is supported on all OS types.

The ToneFileName Method returns the file name of a saved tone file.

Syntax

```
public String ToneFileName(int tnIndex)
```

Parameters

tnIndex	The zero-based index value of the tone file for which file name is to be returned (see Remarks)

Returns

The name of the tone file

Remarks

For a complete listing of all tone files on a device simply loop the ToneFileName Method for the entire number of files returned by ToneFileCount Method.

The first tone file in the list is indexed as zero. For example, to return the file name for the third file, you pass a 2.

WaveLinkTone Samples

The following sample code demonstrates the implementation and capabilities of the Class WaveLinkTone and its associated methods.

Java Sample

```
/**
* This source code is for demonstration
* purposes only and should be treated as such.
*
* NOTE: For this demo, we are assuming a 20x8 display area.
*/
package samplecode;
import com.wavelink.clientui.*;
import java.io.*;
public class JavaToneDemo extends WaveLinkApplication {
    public void WaveLinkMain(String[] args)
    {
        WaveLinkIO ioIface = new WaveLinkIO();
        WaveLinkTone toneIface = new WaveLinkTone();
        // This application demonstrates the use of the
        // various WaveLinkTone methods.
        // It assumes a display area of 20x8. In practice,
        // use a WaveLinkTerminal object to format output to fit
        // the current device.
        // The following lines clear the display and output the
        // app title in reverse video.
        try {
            ioIface.RFPrint(0, 0, " Welcome To The ",
                            WaveLinkIO.WLCLEAR
                            | WaveLinkIO.WLREVERSE);
            ioIface.RFPrint(0, 1, " WaveLinkTone Demo ",
                            WaveLinkIO.WLREVERSE);
            ioIface.RFPrint(0, 3, "Hit a key to proceed",
                             WaveLinkIO.WLNORMAL);
            // Hide the cursor for a cleaner display
            ioIface.RFPrint(0, 20, "", WaveLinkIO.WLNORMAL);
            ioIface.GetEvent();
           // An RF application can respond to user input in 3
           // ways: not at all, through the display, or through
           // the speaker. Each has its benefits and drawbacks.
           // By using the device's speaker, an application
           // can quickly respond to user input with a simple
           // acknowledgement. Tones can also be used to denote
```

```
            // error conditions, status reports, and
            // application progress.
            // This application demonstrates the methods for
            // creating, storing, using, and deleting
            // tones from a device.
            // Add some notes to the WaveLinkTone object.
             for ( int lcv = 0; lcv < 9; lcv++ )
                 toneIface.AddTone(440 + (lcv * 100), 200);
             toneIface.StoreTone("ToneDemo");
             toneIface.PlayTone("ToneDemo");
            //Cleanup
             toneIface.DeleteToneFile("ToneDemo");
          }
          catch ( WaveLinkError wlErr ) {
          //Do error handling
          }
    }
}
```

Class WaveLinkWidget

`com.wavelink.clientui`

The public class **WaveLinkWidget** is supported on: Palm, CE

The Class WaveLinkWidget contains the methods necessary to build and manipulate all types of widget objects.

NOTE Deprecated.

Fields

ALLWIDGETTYPES
AUTOSIZE
BYCELL
BYPERCENTAGE
BYPIXEL
CENTERJUST
CHECKED
DOWNREPEATER
INITDISABLED
INITHIDDEN
INITSTANDARD
LEFTJUST
LEFTREPEATER
PBFIXED
PBHORIZONTAL
PBPROPORTIONAL
PBVERTICAL
RIGHTJUST
RIGHTREPEATER
UNCHECKED
UNDETERMINED
UPREPEATER
WAVELINKBITMAP
WAVELINKBUTTON
WAVELINKCHECKBOX
WAVELINKFIELD
WAVELINKHOTSPOT
WAVELINKLABEL
WAVELINKMENUBAR
WAVELINKPOPUP
WAVELINKPUSHBUTTON
WAVELINKREPEATER
WAVELINKSELECTOR

Constructors

```
public WaveLinkWidget()
```

```
public WaveLinkWidget(IWaveLinkSession currentSession)
```

Properties

CoordinateType
DisplayBackColor
DisplayFlags
DisplayFontName
DisplayFontSize
DisplayForeColor
DisplayText
Height
InitialFlags
InitialValue
PlatformFlags
SpecialString
WidgetID
WidgetType
Width
XCoord
YCoord

Methods

Enable
Focus
SetCoordinates
SetDisplayInfo
SetInitialInfo
SetLabel
SetReturnInfo
SetSpecialInfo
Show

Remarks

The WaveLinkWidget object is used primarily for manipulating widgets. Although you can also use the WaveLinkWidget object to create widgets, the WaveLinkFactory object effectively automates the widget creation process. For this reason, the WaveLinkFactory object is the recommended method for creating widgets.

NOTE Before widgets display on the device, you must use the StoreWidgets Method. To return input from a widget, use the RFInput Method or GetEvent Method. These two methods automatically return the widget ID. To process input from the widget based on the widget ID, use the LastExtendedType Method. To process input from the widget based on its general input type (scanned, keyed, widget input). use the LastInputType Method.

Field Detail

ALLWIDGETTYPES

```
public static final int ALLWIDGETTYPES
```

Includes all widget types.

AUTOSIZE

```
public static final int AUTOSIZE
```

Widget Height or Width - Determine the widget extent by size of label text.

BYCELL

```
public static final int BYCELL
```

Coordinate Type - Position the widget using cell coordinates.

BYPERCENTAGE

```
public static final int BYPERCENTAGE
```

Coordinate Type - Position the widget using percentage coordinates.

BYPIXEL

```
public static final int BYPIXEL
```

Coordinate Type - Position the widget using pixel coordinates.

CENTERJUST

```
public static final int CENTERJUST
```

Label Formatting Flag - Center justify the widget label.

CHECKED

```
public static final int CHECKED
```

Checkbox State - This constant represents a checked checkbox state.

DOWNREPEATER

```
public static final int DOWNREPEATER
```

Repeater Button Type - This constant specifies a down arrow type repeater button.

INITHIDDEN

```
public static final int INITHIDDEN
```

Initial State - Initial state of the widget is hidden.

INITDISABLED

```
public static final int INITDISABLED
```

Initial State - Initial state of the widget is disabled.

INITSTANDARD

```
public static final int INITSTANDARD
```

Initial State - Initial state of the widget is standard (shown and enabled).

LEFTJUST

```
public static final int LEFTJUST
```

Label Formatting Flag - Left justify the widget label.

LEFTREPEATER

```
public static final int LEFTREPEATER
```

Repeater Button Type - This constant specifies a left arrow type repeater button.

PBFIXED

```
Public static final int PBFIXED
```

Initial State - Size each box of a push button widget equally.

PBHORIZONTAL

```
Public static final int PBHORIZONTAL
```

Initial State - Position a push button widget horizontally.

PBPROPORTIONAL

```
Public static final int PBPROPORTIONAL
```

Initial State - Size each box of a push button widget to the size of the text that it contains.

PBVERTICAL

```
Public static final int PBVERTICAL
```

Initial State - Position a push button widget vertically.

RIGHTJUST

```
public static final int RIGHTJUST
```

Label Formatting Flag - Right justify the widget label.

RIGHTREPEATER

```
public static final int RIGHTREPEATER
```

Repeater Button Type - This constant specifies a right arrow type repeater button.

UNCHECKED

```
public static final int UNCHECKED
```

Checkbox State - This constant represents an unchecked checkbox state.

UNDETERMINED

```
public static final int UNDETERMINED
```

Checkbox State - This constant represents an undetermined checkbox state.

UPREPEATER

```
public static final int UPREPEATER
```

Repeater Button Type - This constant specifies an up arrow type repeater button.

WAVELINKBITMAP

```
public static final int WAVELINKBITMAP
```

Widget Type - A bitmap image widget. This widget is based on a bitmap file.

WAVELINKBUTTON

```
public static final int WAVELINKBUTTON
```

Widget Type - A button widget. This widget is a standard labeled button that triggers a specific action when the user "clicks" it.

WAVELINKCHECKBOX

```
public static final int WAVELINKCHECKBOX
```

Widget Type - A checkbox widget. This widget stores an on/off state. The on/off state switches when the user taps it.

WAVELINKFIELD

```
public static final int WAVELINKFIELD
```

Widget Type - A field widget. This widget is an input box.

WAVELINKHOTSPOT

```
public static final int WAVELINKHOTSPOT
```

Widget Type - A hotspot widget. This widget is an invisible area that triggers a specific action when the user taps it.

WAVELINKLABEL

```
public static final int WAVELINKLABEL
```

Widget Type - A label widget. This widget consists of positioned text displayed in a proportionally spaced font.

WAVELINKMENUBAR

```
public static final int WAVELINKMENUBAR
```

Widget Type - A menubar widget. This widget is a menubar containing a set of menus. Each menu activates a pop-up list when the user taps it. The menus are based on WaveLinkMenu configurations.

WAVELINKPOPUP

```
public static final int WAVELINKPOPUP
```

Widget Type - A popup trigger widget. This widget displays a pop-up list when the user taps it. The currently selected item appears to the right of the trigger.

WAVELINKPUSHBUTTON

```
public static final int WAVELINKPUSHBUTTON
```

Widget Type - A push button widget. This widget consists of a set of buttons that store an on/off state. When the user selects a button, the selected button stores the "on" state, and all other buttons store the "off" state. Only one button can store the "on" state at any time.

WAVELINKREPEATER

```
public static final int WAVELINKREPEATER
```

Widget Type - A repeater button widget. This widget consists of a button that implies that something can be scrolled or incremented. The four available buttons are: left arrow, right arrow, up arrow, and down arrow.

WAVELINKSELECTOR

```
public static final int WAVELINKSELECTOR
```

Widget Type - A selector trigger widget. This widget triggers a dialog box when the user "clicks" it.

Property Summary

CoordinateType Property

- Stores and retrieves the coordinate type of the widget.

DisplayBackColor Property

- Stores and retrieves the background color of the widget.

DisplayFlags Property

- Stores and retrieves the label formatting flags of the widget.

DisplayFontName Property

- Stores and retrieves the font name of the widget.

DisplayFontSize Property

- Stores and retrieves the font size of the widget.

DisplayForeColor Property

- Stores and retrieves the foreground color of the widget.

DisplayText Property

- Stores and retrieves the label display text of the widget.

Height Property

- Stores and retrieves the height of the widget.

InitialFlags Property

- Stores and retrieves the initial state of the widget.

InitialValue Property

- Stores and retrieves the initial value of the widget.

PlatformFlags Property

- Stores and retrieves the platform-specific flags of the widget.

SpecialString Property

- Stores and retrieves information specific to a widget based on its type.

WidgetID Property

- Retrieves the unique identifier of the current widget.

WidgetType Property

- Stores and retrieves the widget type.

Width Property

- Stores and retrieves the width of the widget.

XCoord Property

- Stores and retrieves the starting left coordinate of the widget.

YCoord Property

- Stores and retrieves the starting top coordinate of the widget.

Method Summary

Enable Method

- Enables or disables the widget.

Focus Method

- Sets the input focus to the current widget.

SetCoordinates Method

- Sets the type of positioning, the top left corner, the width, and the height of the widget.

SetDisplayInfo Method

- Sets the formatting and text of the widget.

SetInitialInfo Method

- Sets the initial value and state of the widget.

SetLabel Method

- Changes the display text of the current widget.

SetReturnInfo Method

- Sets the type and value of events returned by the widget.

SetSpecialInfo Method

- Takes in a WaveLinkSpecialInfo object.

Show Method

- Displays or hides the current widget.

CoordinateType Property

This member of Class WaveLinkWidget is supported on: Palm, CE

The CoordinateType Property determines how the values entered for the x and y-coordinates of the widget are interpreted.

Syntax

```
public int getCoordinateType()

public void setCoordinateType (int coordType)
                    throws IllegalArgumentException
```

Parameters

coordType The coordinate type (see Remarks)

Returns

The coordinate type of the widget (see Remarks)

Throws

IllegalArgumentException

Remarks

The possible values are:

BYCELL	- Position the widget using cell coordinates (rows and columns)
BYPERCENTAGE	- Position the widget using percentage coordinates
BYPIXEL	- Position the widget using pixel coordinates

The CoordinateType Property determines how to interpret the values entered in the XCoord, YCoord, Height, and Width properties for the widget. The default value is BYCELL.

See the Constant Values appendix for the numeric equates returned by the function.

DisplayBackColor Property

This member of Class WaveLinkWidget is supported on: CE

The DisplayBackcolor Property determines the background color of the widget.

Syntax

```
public long getDisplayBackColor()
```

```
public void setDisplayBackColor(long backColor)
```

Parameters

*backColo*r	The background color for widgets in RGB format(see Remarks)

Returns

The background color used for widgets

Remarks

The background color must be specified in six-byte RGB format (example, FFFFFF).

Pass a value of –1 to use the default system background color.

DisplayFlags Property

This member of Class WaveLinkWidget is supported on: Palm, CE

The DisplayFlags Property determines the label formatting flags of the widget.

Syntax

```
public long getDisplayFlags()

public void setDisplayFlags(long displayFlags)
```

Parameters

displayFlags	The variable that specifies the label formatting flags (see Remarks)

Returns

The label formatting flags (see Remarks)

Remarks

Valid settings for the label formatting flags include:

CENTERJUST	- Center justify the label
LEFTJUST	- Left justify the label
RIGHTJUST	- Right justify the label

The additional possible flags for a push button widget are:

PBFIXED	- Size each push button equally
PBHORIZONTAL	- Position push button horizontally
PBPROPORTIONAL	- Position to the size of the text
PBVERTICAL	- Postion push button vertically

See the Constant Values appendix for the numeric equates returned by the function.

Example

```
WaveLinkFactory factoryIface = new WaveLinkFactory();
WaveLinkWidgetCollection colIface = new
    WaveLinkWidgetCollection();
WaveLinkWidget myWidget;

    try {
.
.
```

```
.
        myWidget = factoryIface.CreateButton(4, 12, 0, 0,
                   " Start ", colIface);
        // Set the property to format the widget display.
        myWidget.setDisplayFlags(WaveLinkWidget.CENTERJUST);
        colIface.StoreWidgets();
    }
    catch (WaveLinkError wlErr) {
        // Do error handling
    }
```

DisplayFontName Property

This member of Class WaveLinkWidget is supported on: Palm, CE

The DisplayFontName Property determines the font name of the widget.

Syntax

```
public String getDisplayFontName()
public void setDisplayFontName(String fontName)
```

Parameters

fontName	The variable that specifies the font name

Returns

The font name of the widget

Remarks

Pass a *fontName* value of null to use the system default font.

For details on specifying font name for CE devices, see client documentation.

The possible values for *fontName* for Palm devices:

BOLD	- Bold font
FIXED	- Monospace font
LARGE	- Large font
LARGEBOLD	- Large bold font
STANDARD	- Standard Palm OS font

DisplayFontSize Property

This member of Class WaveLinkWidget is supported on: CE

The DisplayFontSize Property determines the font size of the widget.

Syntax

```
public int getDisplayFontSize()
```

```
public void setDisplayFontSize(int fontSize)
```

Parameters

fontSize	The variable that specifies the font size

Returns

The font size of the widget

Remarks

Pass a value of –1 to use the default system font size.

The font size must be specified in points. See documentation for the CE client for more information about specifying font size.

DisplayForeColor Property

This member of Class WaveLinkWidget is supported on: CE

The DisplayForeColor Property determines the foreground color of the widget.

Syntax

```
public long getDisplayForeColor()
```

```
public void setDisplayForeColor(long foreColor)
```

Parameters

foreColor	The variable that specifies the foreground color

Returns

The foreground color of the widget

Remarks

The color must be specified in six-byte RGB format (example, FFFFFF).

Pass a value of –1 to use the default system foreground color.

DisplayText Property

This member of Class WaveLinkWidget is supported on: Palm, CE

The DisplayText Property determines the label display text of the widget.

Syntax

```
public String getDisplayText()
```

```
public void setDisplayText(String displayText)
```

Parameters

displayText The variable that specifies the label display text

Returns

The label display text of the widget

Example

See the WaveLinkScribblePad CancelButton Property.

Height Property

This member of Class WaveLinkWidget is supported on: Palm, CE

The Height Property determines the height of the widget.

Syntax

```
public int getHeight()

public void setHeight(in widgetHeight)
                    throws IllegalArgumentException
```

Parameters

widgetHeight The variable that specifies the height (see Remarks)

Returns

The height of the widget

Throws

IllegalArgumentException

Remarks

The default setting for the Height Property is AUTOSIZE, which sizes a button or selector trigger widget according to the size of the text that it contains. We recommended that you use the AUTOSIZE setting for the height and width of all widget types except hotspot widgets.

To give a hotspot widget functionality you must set its width and height to something other than AUTOSIZE.

NOTE The CoordinateType Property determines how to interpret numeroc values entered for the height of the widget.

InitialFlags Property

This member of Class WaveLinkWidget is supported on: Palm, CE

The InitialFlags Property stores and retrieves the initial state of the widget.

Syntax.

```
public long getInitialFlags()
```

```
public void setInitialFlags(long initalFlags)
```

Parameters

initialFlags The variable that specifies the initial state (see Remarks)

Returns

The initial state of the widget (see Remarks)

Throws

WaveLinkError

Remarks

The possible values are:

INITDISABLED	- Initial state of the widget is disabled
INITHIDDEN	- Initial state of the widget is hidden
INITSTANDARD	- Initial state of the widget is shown and enabled

A disabled widget appears on the screen, but does not return when you click it. The appearance of a disabled widget is platform dependent. Typically, a disabled widget is either grayed out or its display text is crosshatched.

NOTE Unlike other widgets, a disabled hotspot widget does not appear on the RF screen.

See the Constant Values appendix for the numeric equates returned by the function.

Example

```
WaveLinkFactory factoryIface = new WaveLinkFactory();
WaveLinkWidgetCollection colIface = new
    WaveLinkWidgetCollection();
WaveLinkWidget myWidget;
```

```
    try {
.
.
.
        myWidget = factoryIface.CreateButton(4, 12, 0, 0,
                   " Start ", colIface);
        // Set the widget state.
       myWidget.setInitialFlags(WaveLinkWidget.INITHIDDEN);
        colIface.StoreWidgets();
    }
    catch (WaveLinkError wlErr) {
        // Do error handling
    }
```

InitialValue Property

This member of Class WaveLinkWidget is supported on: Palm, CE

The InitialValue Property sets and retrieves the initial value of the widget.

Syntax

```
public String getInitialValue()
```

```
public void setInitialValue(String initialValue)
```

Parameters

initialValue The variable that specifies the initial value (see Remarks)

Returns

The initial value of the widget (see Remarks)

Remarks

The possible values for a checkbox widget:

CHECKED	- Checked checkbox state
UNCHECKED	- Unchecked checkbox state
UNDETERMINED	- Undetermined checkbox state

The possible values for a repeater button widget:

DOWNREPEATER	- Down arrow button
LEFTREPEATER	- Left arrow button
RIGHTREPEATER	- Right arrow button
UPREPEATER	- Up arrow button

See the Constant Values appendix for the numeric equates returned by the function.

PlatformFlags Property

This member of Class WaveLinkWidget is *not* currently implemented on clients.

The PlatformFlags Property determines the platform-specific flags of the widget.

Syntax

```
public long getPlatformFlags()
```

```
public void setPlatformFlags(long newFlags)
```

Parameters

newFlags	The platform-specific flag variable

Returns

The platform-specific flags

SpecialString Property

This member of Class WaveLinkWidget is supported on: Palm, CE

The SpecialString Property specifies information specific to a widget, based on its type.

Syntax

```
public String getSpecialString()
```

```
public void setSpecialString(String specialString)
```

Parameters

specialString	The variables that stores the unique widget information (see Remarks)

Returns

The unique information (see Remarks)

Remarks

For a popup trigger and push button widget, the SpecialString Property specifies the name of the WaveLinkMenu configuration associated with the widget.

For a bitmap widget, the SpecialString Property specifies the bitmap file. The file name must include the full path.

For a menubar widget, the SpecialString Property contains the formatted menu list associated with the widget. This information is only available after you use the SetSpecialInfo Method.

WidgetID Property

This member of Class WaveLinkWidget is supported on: Palm, CE

The WidgetID Property retrieves the unique identifier for the current widget.

Syntax

```
public int getWidgetID()
```

Returns

The unique widget identifier

Remarks

The WidgetID Property is read-only. The unique widget ID is automatically assigned when you create the widget.

The widget ID is returned through a previous RFInput Method call or a GetEvent Method call. To process input from the widget based on the widget ID, use the LastExtendedType Method.

To process user input based on the general input type (command key, widget input, scanned input), use the LastInputType Method.

Example

```
WaveLinkIO ioIface = new WaveLinkIO();
WaveLinkFactory factoryIface = new WaveLinkFactory();
WaveLinkWidgetCollection colIface = new
    WaveLinkWidgetCollection();
WaveLinkWidget Quit;
string resultStr;

    // Create "Quit" widget (not shown).
.
.
.

    try {
        //Use RFInput or GetEvent to obtain input
        resultStr = ioIface.GetEvent();
        switch(ioIface.LastInputType())
        {
            case ioIface.WLCOMMANDTYPE:
                //Process command type
                break;
            case ioIface.WLKEYTYPE:
                //Process keyed command
                break;
            case ioIface.WLWIDGETTYPE:
                if(ioIface.LastExtendedType() ==
                   Quit.getWidgetID())
                ;   //Quit button pressed, exit process
                break;
        }
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

WidgetType Property

This member of Class WaveLinkWidget is supported on: Palm, CE

The WidgetType Property determines the type of widget.

Syntax

```
public int getWidgetType()

public void setWidgetType(int newType)
                    throws WaveLinkError
```

Parameters

newType The variable that stores the widget type (see Remarks)

Returns

The widget type (see Remarks)

Throws

WaveLinkError

Remarks

The possible widget types are:

- WAVELINKALLTYPES
- WAVELINKBITMAP
- WAVELINKBUTTON
- WAVELINKCHECKBOX
- WAVELINKHOTSPOT
- WAVELINKFIELD
- WAVELINKLABEL
- WAVELINKMENUBAR
- WAVELINKPOPUP
- WAVELINKPUSHBUTTON
- WAVELINKREPEATER
- WAVELINKSELECTOR

See the Constant Values appendix for the numeric equates returned by the function.

Width Property

This member of Class WaveLinkWidget is supported on: Palm, CE

The Width Property determines the width of the widget.

Syntax

```
public int getWidth()

public void setWidth(int widgetWidth)
                     throws IllegalArgumentException
```

Parameters

widgetWidth The variable that specifies the width (see Remarks)

Returns

The width of the widget

Throws

IllegalArgumentException

Remarks

The default setting for the Width Property is AUTOSIZE, which sizes a button or selector trigger widget according to the size of the text that it contains. We recommended that you use the AUTOSIZE setting for the height and width of all widget types except hotspots.

NOTE To give a hotspot widget functionality you must set its width and height to something other than AUTOSIZE.

The CoordinateType Property determines how to interpret numerical values entered for the height of the widget.

XCoord Property

This member of Class WaveLinkWidget is supported on: Palm, CE

The XCoord Property determines the starting left-coordinate of the widget.

Syntax

```
public int getXCoord()

public void setXCoord(int xCoord)
                    throws IllegalArgumentException
```

Parameters

xCoord	The variable that specifies the starting left-coordinate of the widget (see Remarks)

Returns

The starting left-coordinate of the widget

Throws

IllegalArgumentException

Remarks

The default setting for the XCoord Property is zero (0).

The CoordinateType Property determines how to interpret values entered for the horizontal position of the widget.

YCoord Property

This member of Class WaveLinkWidget is supported on: Palm, CE

The YCoord Property determines the starting upper-coordinate of the widget.

Syntax

```
public int getYCoord()

public void setYCoord(int yCoord)
                    throws IllegalArgumentException
```

Parameters

yCoord	The variable that specifies the starting upper-coordinate of the widget (see Remarks)

Returns

The starting upper-coordinate of the widget

Throws

IllegalArgumentException

Remarks

The default setting for the YCoord Property is zero (0).

The CoordinateType Property determines how to interpret values entered for the vertical position of the widget.

Enable Method

This member of Class WaveLinkWidget is supported on: Palm, CE

The Enable Method enables or disables the specified widget.

Syntax

```
public void Enable(boolean enableWidget)
                    throws WaveLinkError
```

Parameters

enableWidget	A true or false value that determines whether the widget is enabled or disabled (see Remarks)

Throws

WaveLinkError

Remarks

When the *enableWidget* parameter is set to true, the widget is enabled, when it is set to false, the widget is disabled.

A disabled widget appears on the screen, but does not return when you click it. The appearance of a disabled widget is platform dependent. Typically, a disabled widget is either grayed out or its display text is crosshatched.

NOTE A disabled hotspot widget does not appear on the RF screen.

Example

```
    WaveLinkFactory factoryIface = new WaveLinkFactory();
    WaveLinkWidgetCollection colIface = new
        WaveLinkWidgetCollection();
    WaveLinkWidget myWidget;

        try {
    .
    .
    .
            myWidget = factoryIface.CreateButton(1, 2, 0, 0,
                       " Go ", colIface);
            // Set the property to disable the widget.

myWidget.setInitialFlags(WaveLinkWidget.INITDISABLED);
            colIface.StoreWidgets();
        }
        catch (WaveLinkError wlErr) {
            //Do error handling
        }
    .
    .
    .
        // Now enable the disabled widget.
        myWidget.Enable(true);
```

Focus Method

This member of Class WaveLinkWidget is supported on: Palm, CE

The Focus Method sets the input focus to the current widget.

Syntax

```
public void Focus()
                    throws WaveLinkError
```

Throws

WaveLinkError

Remarks

NOTE Currently the Focus Method applies only to field widgets in the Palm Client.

SetCoordinates Method

This member of Class WaveLinkWidget is supported on: Palm, CE

The SetCoordinates Method sets the type of positioning, the top-left corner, the width, and the height of the widget.

Syntax

```
public void SetCoordinates(int coordType,
                           Point newPosition,
                           Dimension newDimension)
                    throws IllegalArgumentException
```

Parameters

coordType	The classification of the values placed in NewPoint and NewDimension (see Remarks)
newPosition	The x and y-coordinates of the widget
newDimension	The height and width of the widget (see Remarks)

Throws

IllegalArgumentException

Remarks

The possible coordinate types are:

BYCELL	- Position the widget using cell coordinates (rows and columns)
BYPIXEL	- Position the widget using pixel coordinates
BYPERCENTAGE	- Position the widget using percentage coordinates

To assign AUTOSIZE to the widget, pass a dimension that contains a value of 0 or AUTOSIZE.

AUTOSIZE sizes a button or selector trigger widget according to the size of the text that it contains. We recommended that you use the AUTOSIZE setting for the height and width of all widget types except hotspots.

NOTE To give a hotspot widget functionality you must set its width and height to something other than AUTOSIZE.

SetDisplayInfo Method

This member of Class WaveLinkWidget is supported on: CE

The SetDisplayInfo Method sets the formatting and display text of the widget.

Syntax

```
public void SetDisplayInfo(String fontName, int fontSize,
            long foreColor, long backColor,
            String displayText, long displayFlags)
```

Parameters

fontName	The type face for the label display
fontSize	The type size for the label display
foreColor	The color, in six-byte RGB format, of the label text (example, FFFFFF)
backColor	The color, in six-byte RGB format, of the widget background (example, FFFFFF)
displayText	The display text of the widget
displayFlags	The modifiers for the display text (see Remarks)

Remarks

See client documentation for more information about specifying font size and font name.

The possible values for *displayFlags* parameter are:

CENTERJUST	- Center justify the label
LEFTJUST	- Left justify the label
RIGHTJUST	- Right justify the label

The additional possible values for the *displayFlags* parameter for a push button widget are:

PBFIXED	- Size each push button equally
PBHORIZONTAL	- Position push button horizontally
PBPROPORTIONAL	- Position to the size of the text
PBVERTICAL	- Postion push button vertically

SetInitialInfo Method

This member of Class WaveLinkWidget is supported on: Palm, CE

The SetInitialInfo Method sets the initial value and state of the widget.

Syntax

```
public void SetInitialInfo(String initialVal,
                           long initialState)
```

Parameters

initialVal	The variable that specifies the initial value (see Remarks).
initialState	The variable that specifies the initial state (see Remarks)

Remarks

The values for the initial value of a checkbox widget:

CHECKED	- Checked checkbox state
UNCHECKED	- Unchecked checkbox state
UNDETERMINED	- Undetermined checkbox state

The values for the initial value of a repeater button widget:

DOWNREPEATER	- down arrow button
LEFTREPEATER	- left arrow button
RIGHTREPEATER	- right arrow button
UPREPEATER	- up arrow button

The possible values for the initial state of the widget are:

INITDISABLED	- Initial state of the widget is disabled
INITHIDDEN	- Initial state of the widget is hidden
INITSTANDARD	- Initial state of the widget is standard (shown and enabled)

A disabled widget appears on the screen, but does not return when you click it. The appearance of a disabled widget is platform dependent. Typically, a disabled widget is either grayed out or its display text is crosshatched.

NOTE Unlike other widgets, a disabled hotspot widget does not appear on the RF screen.

SetLabel Method

This member of Class WaveLinkWidget is supported on: Palm, CE

The SetLabel Method changes the display text for the current widget.

Syntax

```
public void SetLabel(String newLabel)
                    throws WaveLinkError
```

Parameters

newLabel The new text for the current widget

Throws

WaveLinkError

Remarks

The SetLabel Method applies to button, checkbox, and selector trigger widgets.

SetReturnInfo Method

This member of Class WaveLinkWidget is supported on: Palm, CE

The SetReturnInfo Method sets the type and value of events returned by the widget.

Syntax

```
public void SetReturnInfo(int infoIndex, int eventType,
                          String eventValue,
                          int eventSymbology)
                    throws IllegalArgumentException
```

Parameters

infoIndex	The variable that specifies the return type for widgets with multiple return types. Pass a value of 0 unless the widget has multiple return types (see Remarks).
eventType	The variable that specifies the event type returned by the widget (see Remarks)
eventValue	The variable that specifies the value returned by the widget (see Remarks)
eventSymbology	The variable that specifies the barcode type returned by the widget (see Remarks). Use a value of WLNOSYMBOLOGY if you do not want a barcode symbology returned by the widget.

Throws

IllegalArgumentException

Remarks

Use a value of zero (0) for the *infoIndex* parameter for all widgets unless the widget has multiple return types. Currently, only the checkbox widget has multiple return types. The possible values for the checkbox widget are:

0 - UNCHECKED

1 - CHECKED

2 – UNDETERMINED

The default event value returned by a widget is different for different widget types. Use the RFInput Method or GetEvent Method to return the value of the widget.

The default event type is WLWIDGETTYPE. To process user input based on the return type (command key, widget input, scanned input., use the LastInputType Method.

The default event value returned by a widget is different for different widget types. Use the RFInput Method or GetEvent Method to return the value of the widget.

The possible symbology types are:

B_UPC

CODABAR

CODE_11

CODE_39

CODE_93

CODE_128

CODE_D25

CODE_I25

D25_IATA

EAN_8

EAN_13

MSI

PDF_417

TO_39

UCC_128

UPC_A

UPC_E0

UPC_E1

WLNOSYMBOLOGY

NOTE Use a value of WLNOSYMBOLOGY if you do not want a barcode symbology returned by the widget.

Example

```
WaveLinkFactory factoryIface = new WaveLinkFactory();
WaveLinkWidgetCollection colIface = new
    WaveLinkWidgetCollection();
WaveLinkWidget myWidget;
```

```
    try {
        // Create one or more widgets.
.
.
.
      myWidget = factoryIface.CreateRepeaterButton(1, 2, 0,
                    0, WaveLinkWidget.RIGHTARROW, colIface);
        // Set the return info.
        myWidget.SetReturnInfo(0, WaveLinkIO.WLKEYTYPE,
                  "RIGHT", WaveLinkBarcode.WLNOSYMBOLOGY);
        colIface.StoreWidgets();
    }
    catch (WaveLinkError wlErr) {
        // Do error handling
    }
```

SetSpecialInfo Method

This member of Class WaveLinkWidget is supported on: Palm, CE

The SetSpecialInfo Method takes in a WaveLinkSpecialInfo object.

Syntax

```
public void SetSpecialInfo(WaveLinkSpecialInfo
                           specialInfo)
```

Parameters

specialInfo	The special information associated with the WaveLinkSpecialInfo object

Remarks

Currently, the only valid WaveLinkSpecialInfo object is the WaveLinkMenubarInfo object.

Although you can use the SetSpecialInfo Method to associate a WaveLinkMenubarInfo object with a menubar widget, it is recommended that you use the CreateMenubar Method of the WaveLinkFactory object to automate this process.

Show Method

This member of Class WaveLinkWidget is supported on: Palm, CE

The Show Method displays or hides the current widget based on the *showWidg* parameter.

Syntax

```
public void Show(boolean showWidg)
                    throws WaveLinkError
```

Parameters

showWidg	A true or false value that determines if a widget is displayed or hidden (see Remarks)

Throws

WaveLinkError

Remarks

When the *ShowWidg* parameter is set to true, the Show Method displays a widget that has been previously hidden. A value of false hides the widget.

Before using the Show Method, you must use the StoreWidgets Method of the Class WaveLinkWidgetCollection to store widgets on the device while simultaneously causing them to display on the RF screen.

NOTE The StoreWidgets Method does not display widgets for which initial state is set to hidden.

Example

```
WaveLinkFactory factoryIface = new WaveLinkFactory();
WaveLinkWidgetCollection colIface = new
    WaveLinkWidgetCollection();
WaveLinkWidget myWidget;

    try {
.
.
.
      myWidget = factoryIface.CreateRepeaterButton(1, 2, 0,
                  0, WaveLinkWidget.RIGHTARROW, colIface);
        // Set the property to hide the widget.
       myWidget.setInitialFlags(WaveLinkWidget.INITHIDDEN);
        colIface.StoreWidgets();
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
.
.
.
    try {
        // Now show the hidden widget.
        myWidget.Show(true);
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

Class WaveLinkWidgetCollection

`com.wavelink.clientui`

The public class **WaveLinkWidgetCollection** is supported on: Palm, CE

The WaveLinkWidgetCollection object logically groups widget objects together.

NOTE Deprecated.

Constructors

```
public WaveLinkWidgetCollection()

public WaveLinkWidgetCollection(IWaveLinkSession
                                currentSession)

public WaveLinkWidgetCollection(WaveLinkIO ioIface)
```

Methods

AddWidget
ClearWidgets
DeleteAllWidgets
DeleteWidgets
EnableAllWidgets
EnableWidgets
RemoveWidget
ShowAllWidgets
ShowWidgets
StoreWidgets
Widget

Remarks

Use the WaveLinkWidgetCollection object to manipulate a group of widgets, such as a set of widgets that appear on the same screen.

Method Summary

AddWidget Method

- Adds a widget to the current collection.

ClearWidgets Method

- Clears all widgets from the current collection.

DeleteAllWidgets Method

- Deletes all widgets present on the device.

DeleteWidgets Method

- Deletes all widgets in the collection from the device.

EnableAllWidgets Method

- Enables or disables all widgets present on the device.

EnableWidgets Method

- Enables or disables all widgets in the current collection.

RemoveWidget Method

- Removes a specific widget based on its widget ID.

ShowAllWidgets Method

- Removes a specific widget based on its index value.

ShowAllWidgets Method

- Shows or hides all widgets present on the device.

ShowWidgets Method

- Shows or hides all widgets in the current collection.

StoreWidgets Method

- Stores all widgets in the collection and displays them on the device.

Widget Method

- Returns the widget specified by the widget ID.

AddWidget Method

This member of Class WaveLinkWidgetCollection is supported on: Palm, CE

The AddWidget Method adds a widget to the current collection.

Syntax

```
public void AddWidget(WaveLinkWidget newWidget)
```

Parameters

newWidget The widget object to be added

ClearWidgets Method

This member of Class WaveLinkWidgetCollection is supported on: Palm, CE

The ClearWidgets Method clears all widgets within the current object.

Syntax

```
public void ClearWidgets()
```

DeleteAllWidgets Method

This member of Class WaveLinkWidgetCollection is supported on: Palm, CE

The DeleteAllWidgets Method deletes all widgets present on the device.

Syntax

```
public void DeleteAllWidgets()
                                throws WaveLinkError
```

Throws

WaveLinkError

DeleteWidgets Method

This member of Class WaveLinkWidgetCollection is supported on: Palm, CE

The DeleteWidgets Method deletes all widgets within the current collection on the device.

Syntax

```
public void DeleteWidgets()
                    throws WaveLinkError
```

Throws

WaveLinkError

EnableAllWidgets Method

This member of Class WaveLinkWidgetCollection is supported on: Palm, CE

The EnableAllWidgets Method enables or disables all widgets present on the device.

Syntax

```
public void EnableAllWidgets(boolean enableWidgs)
                    throws WaveLinkError
```

Parameters

enableWidgs	A true or false value that determines whether widgets on the device are enabled or disabled (see Remarks)

Throws

WaveLinkError

Remarks

For the *enableWidgs* parameter, a true value enables widgets, a false value disables widgets.

EnableWidgets Method

This member of Class WaveLinkWidgetCollection is supported on: Palm, CE

The EnableWidgets Method enables or disables all widgets within the current collection.

Syntax

```
public void EnableWidgets(boolean enableWidgs)
                    throws WaveLinkError
```

Parameters

enableWidgs	A true or false value that determines whether widgets are enabled or disabled (see Remarks)

Throws

WaveLinkError

Remarks

For the *enableWidgs* parameter, a true value enables widgets and a false value disables widgets.

Example

```
WaveLinkFactory factoryIface = new WaveLinkFactory();
WaveLinkWidgetCollection colIface = new
    WaveLinkWidgetCollection();
WaveLinkWidget myWidget;

    try {
        // Create one or more widgets.
.
.
.
        myWidget = factoryIface.CreateButton(1, 2, 0, 0,
                   " Go ", colIface);
        // Set the property to disable the widget.
     myWidget.setInitialFlags(WaveLinkWidget.INITDISABLED);
        colIface.StoreWidgets();
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
.
.
.
    try {
        // Now enable the disabled widget(s).
        colIface.EnableWidgets(true);
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

RemoveWidget Method

This member of Class WaveLinkWidgetCollection is supported on: Palm, CE

The RemoveWidget Method removes the specified widget from the collection based on the widget ID or index value of the widget.

Syntax

```
public void RemoveWidget(int widgetID)

public void RemoveWidget(WaveLinkWidget remWidget)
                    throws IllegalArgumentException
```

Parameters

widgetID	The widget ID of the widget to be removed
remWidg	The widget to be removed.

Throws

IllegalArgumentException

Remarks

The RemoveWidget Methods can remove a widget based on either its widget ID or the name of the widget object to be removed.

ShowAllWidgets Method

This member of Class WaveLinkWidgetCollection is supported on: Palm, CE

The ShowAllWidgets Method shows or hides all widgets present on the device.

Syntax

```
public void ShowAllWidgets(boolean showWidgs)
                    throws WaveLinkError
```

Parameters

showWidgs	A true or false value that determines whether widgets are shown or hidden (see Remarks)

Throws

WaveLinkError

Remarks

For the *showWidgs* parameter, a true value shows widgets and a false value hides widgets. When the *showWidgs* parameter is set to true, the ShowAllWidgets Method displays a widget that has been previously hidden.

Before using the ShowAllWidgets Method, you must use the StoreWidgets Method to store widgets on the device while simultaneously displaying them on the RF screen. The StoreWidgets Method does not display widgets for which the initial state is set to hidden.

ShowWidgets Method

This member of Class WaveLinkWidgetCollection is supported on: Palm, CE

The ShowWidgets Method shows or hides all widgets within the current collection.

Syntax

```
public void ShowWidgets(boolean showWidgs)
                    throws WaveLinkError
```

Parameters

showWidgs	A true or false value that determines whether widgets are shown or hidden (see Remarks)

Throws

WaveLinkError

Remarks

For the *showWidgs* parameter, a true value shows widgets and a false value hides widgets. When the *showWidgs* parameter is set to true, the ShowAllWidgets Method displays a widget that has been previously hidden.

Before using the ShowWidgets Method, you must use the StoreWidgets Method to store widgets on the device while simultaneously displaying them on the RF screen. The StoreWidgets Method does not display widgets for which the initial state is set to hidden.

Example

```
    WaveLinkWidgetCollection colIface = new
        WaveLinkWidgetCollection();

        try {
            // Create one or more widgets, add them to the
colIface
            // collection, store them (not shown).
    .
    .
    .
        }
        catch (WaveLinkError wlErr) {
            //Do error handling
        }
    .
    .
    .
```

```
try {
    // Hide the widget(s).
    colIface.ShowWidgets(false);
}
catch (WaveLinkError wlErr) {
    //Do error handling
}
```

StoreWidgets Method

This member of Class WaveLinkWidgetCollection is supported on: Palm, CE

The StoreWidgets Method stores all widget objects in the collection. Widgets for which initial state is set to standard display immediately when you use the StoreWidgets Method.

Syntax

```
public void StoreWidgets()
                    throws WaveLinkError
```

Throws

WaveLinkError

Remarks

NOTE Widgets display immediately when you use the StoreWidgets Method *if* their initial state is set to standard. The initial state of a widget is standard by default. Use the InitialFlags Property of the Class WaveLinkWidget to change the initial state.

Example

```
WaveLinkFactory factoryIface = new WaveLinkFactory();
WaveLinkWidgetCollection colIface = new
    WaveLinkWidgetCollection();
WaveLinkIO ioIface = new WaveLinkIO();

    try {
        ioIface.RFPrint(0, 0, "WaveLink Corporation",
                WaveLinkIO.WLCLEAR | WaveLinkIO.WLREVERSE);
        ioIface.RFPrint(0, 1, "   Auto Detailing   ",
                WaveLinkIO.WLNORMAL);
        // Create one or more widgets.
        factoryIface.CreateButton(4, 12, 0, 0, " Start ",
                    colIface);
        // Store the widget(s).
        colIface.StoreWidgets();
    }
    catch (WaveLinkError wlErr) {
        //Do error handling
    }
```

Widget Method

This member of Class WaveLinkWidgetCollection is supported on: Palm, CE

The Widget Method returns the specified widget object from the collection.

Syntax

```
public WaveLinkWidget Widget(int widgetID)
                    throws IllegalArgumentException
```

Parameters

widgetID	The widget ID of the object to be returned

Returns

A reference to the specified widget object

Class WidgetOptions

```
com.wavelink.clientui
```

The public class WidgetOptions is supported on:

The Class WidgetOptions contains options for all the Wavelink widgets. The options for a particular widget are determined by:

1. The WidgetOptions assigned to the widget.
2. For any options not assigned, the WidgetOptions inherited from the WaveLinkScreen.
3. System defaults.

Constructors

```
WidgetOptions()

WidgetOptions(java.awt.Color f, java.awt.Color b)

WidgetOptions(java.awt.Color f, java.awt.Color b,
WaveLinkFont fnt)
```

Methods

addFlags
clearFlags
clone
equals
getBackColor
getFieldMask
getFont
getForeColor
getInitialFocus
getInitialState
getJustification
getJustification
getMaxFieldLength
setBackColor
setFieldMask
setFont
setForeColor
setInitialFocus
setInitialState
setJustification
setMaxFieldLength

Method Summary

addFlags Method

- Adds additional options. See **WaveLinkConstants**.

clearFlags Method

- Clears all flags.

clone Method

- Returns a new WidgetOptions object with all the settings of the original.

equals Method

- Determines whether one WidgetOptions object has the same values as another.

getBackColor Method

- Gets the font background color.

getFieldMask Method

- Gets the field input mask.

getFont Method

- Gets the current font.

getForeColor Method

- Gets the font foreground color.

getInitialFocus Method

- Returns whether this widget will have the focus.

getInitialState Method

- Gets the initial state.

getJustification Method

- Gets the justification of the text on the widget.

getMaxFieldLength Method

- Gets the maximum field length.

setBackColor Method

- Sets the font background color.

setFieldMask Method

- Sets the field mask.

setFlags Method

- Sets options. See WaveLinkConstants.

setFont Method

- Sets the current font.

setForeColor Method

- Sets the font foreground color.

setInitialFocus Method

- Sets the initial focus state.

setInitialState Method

- Sets the initial state.

setJustification Method

- Sets the justification of the text on the widget.

setMaxFieldLength Method

- Sets the maximum field length.

Example

See WaveLinkScreen Samples

addFlags Method

This member of Class WidgetOptions is supported on: CE

The addFlags Method adds additional options. See **WaveLinkConstants** for more information about available flags.

Syntax

```
public void addFlags(long flags)
```

Parameters

flags	The flags – these can be combined using the OR operator

clearFlags Method

This member of Class WidgetOptions is supported on: CE

The clearFlags Method clears all flags

Syntax

```
public void clearFlags()
```

clone Method

This member of Class WidgetOptions is supported on: CE

The clone Method returns a new WidgetOptions object with all the settings of the original.

Syntax

```
public java.lang.Object clone()
```

Returns

A new WidgetOptions object with all the settings of the original copied

equals Method

This member of Class WidgetOptions is supported on: CE

The equals Method determines whether one WidgetOptions object has the same values as another.

Syntax

```
public boolean equals(WidgetOptions o)
```

Parameters

o	the object to compare against

Returns

True if the objects have the same configuration, false if not

getBackColor Method

This member of Class WidgetOptions is supported on: CE

The getBackColor Method gets the font background color.

Syntax

```
public java.awt.Color getBackColor()
```

Returns

A color object with the font's background color

getFieldMask Method

This member of Class WidgetOptions is supported on: CE

The getFieldMask Method gets the field input mask.

Syntax

```
public java.lang.String getFieldMask()
```

Returns

The field mask

getFont Method

This member of Class WidgetOptions is supported on: CE

The getFont Method gets the current font.

Syntax

```
public WaveLinkFont getFont()
```

Returns

A WaveLinkFont object

getForeColor Method

This member of Class WidgetOptions is supported on: CE

The getForeColor Method gets the font foreground color.

Syntax

```
public java.awt.Color getForeColor()
```

Returns

A Color object containing the foreground color

getInitialFocus Method

This member of Class WidgetOptions is supported on: CE

The getInitialFocus Method returns whether the widget will have the focus.

Syntax

```
public boolean getInitialFocus()
```

Returns

True if the object with these WidgetOptions will have initial focus, false if not

getInitialState Method

This member of Class WidgetOptions is supported on: CE

The getInitialState Method gest the initial state.

Syntax

```
public int getInitialState()
```

Returns

The state a widget will be in with these options initially (see Class WaveLink-Constants.State)

getJustification Method

This member of Class WidgetOptions is supported on: CE

The getJustification Method gets the justification of the text on the widget.

Syntax

```
public int getJustification()
```

Returns

The display justification (see Class WaveLinkConstants.Justify)

getMaxFieldLength Method

This member of Class WidgetOptions is supported on: CE

The getMaxFieldLength Method gets the maximum field length.

Syntax

```
public int getMaxFieldLength()
```

Returns

The configured maximum field length

setBackColor Method

This member of Class WidgetOptions is supported on: CE

The setBackColor Method sets the font background color.

Syntax

```
public void setBackColor(java.awt.Color b)
```

Parameters

b	the font background color

setFieldMask Method

This member of Class WidgetOptions is supported on: CE

The setFieldMask Method sets the field mask.

Syntax

```
public void setFieldMask(java.lang.String mask)
```

Parameters

mask	The field mask

setFlags Method

This member of Class WidgetOptions is supported on: CE

The setFlags Method sets options. For more information, see WaveLinkConstants.

Syntax

```
public void setFlags(long flags)
```

Parameters

flags	The flags to set. These can be combined using the OR operator.

setFont Method

This member of Class WidgetOptions is supported on: CE

The setFont Method sets the current font.

Syntax

```
public void setFont(WaveLinkFont f)
```

Parameters

f	The new font – if *null*, inherited defaults will be used.

setForeColor Method

This member of Class WidgetOptions is supported on: CE

The setForeColor Method sets the font foreground color.

Syntax

```
public void setForeColor(java.awt.Color f)
```

Parameters

f	The font foreground color

setInitialFocus Method

This member of Class WidgetOptions is supported on: CE

The setInitialFocus Method sets the initial focus state.

Syntax

```
public void setInitialFocus(boolean initFocus)
```

Parameters

initFocus	The initial focus flag

setInitialState Method

This member of Class WidgetOptions is supported on: CE

The setInitialState Method sets the initial state.

Syntax

```
public void setInitialState(int state)
```

Parameters

state	The initial state

setJustification Method

This member of Class WidgetOptions is supported on: CE

The setJustification Method sets the justification of the text on the widget.

Syntax

```
public void setJustification(int justification)
```

Parameters

justification	The display justification– see Class WaveLinkConstants.Justify

setMaxFieldLength Method

This member of Class WidgetOptions is supported on: CE

The setMaxFieldLength Method sets the maximum field length.

Syntax

```
public void setMaxFieldLength(int len,
                              int maxflag)
```

Parameters

len	The max field length
maxflag	The max field length flag

Constant Values

This appendix contains the numeric values for the WaveLink constants. The WaveLink constants have been classified based on the specific class with which they are associated:

- WaveLinkAuxPort Constants
- WaveLinkBarcode Constants
- WaveLinkConstants Constants
- WaveLinkError Constants
- WaveLinkFont Constants
- WaveLinkIO Constants
- WaveLinkMenu Constants
- WaveLinkTerminal Constants
- WaveLinkWidget Constants

WaveLinkAuxPort Constants

The WavelinkAuxPort constants configure bi-directional access to the serial port of a device.

Table 1: *WaveLinkAuxPort Constants*

Constants	Value	Description
WLCOM1	0	COM port 1 (serial port on Palm devices)
WLCOM2	1	COM port 2 (infrared port on Palm devices)
WLNOCHAR	'\0'	No start/end character
WLNOMAXLENGTH	-1	No query max length
WLWAITFOREVER	-1	No query timeout
BAUD110	10	110000 baud connection
BAUD150	0	150 baud connection

Table 1: *WaveLinkAuxPort Constants*

Constants	Value	Description
BAUD300	1	300 baud connection
BAUD600	2	600 baud connection
BAUD1200	3	1200 baud connection
BAUD1350	4	1350 baud connection
BAUD2400	5	2400 baud connection
BAUD4800	6	4800 baud connection
BAUD9600	7	9600 baud connection
BAUD19200	8	19200 baud connection
BAUD38400	9	38400 baud connection
DATABITS4	4	4 data bits
DATABITS5	0	5 data bits
DATABITS6	1	6 data bits
DATABITS7	2	7 data bits
DATABITS8	3	8 data bits
PARITYEVEN	0	Even parity
PARITYMARK	2	Mark parity
PARITYNONE	4	None parity
PARITYODD	1	Odd parity
PARITYSPACE	3	Space parity
STOPBITS1	0	One stop bit
STOPBITS2	1	Two stop bits
HARDWAREFLOWCTL	2	Hardware flow control
NOFLOWCTL	0	No flow control
SOFTWAREFLOWCTL	1	Software flow control

WaveLinkBarcode Constants

The WaveLinkBarcode constants define barcode configurations for use within your applications.

Table 2: *WavelinkBarcode Constants*

Constants	Value	Description
B_UPC	17	Barcode type B-UPC
CODABAR	6	Barcode type CodaBar
CODE_11	10	Barcode type Code-11
CODE_39	7	Barcode type Code-39
CODE_93	11	Barcode type Code-93
CODE_128	12	Barcode type Code-128
CODE_D25	8	Barcode type Code-D25
CODE_I25	9	Barcode type Code-I25
D25_IATA	14	Barcode type D24-IATA
EAN_8	4	Barcode type EAN-8
EAN_13	5	Barcode type EAN-13
MSI	3	Barcode type MSI
PDF_417	13	Barcode type PDF-417
TO_39	18	Barcode type TO-39
UCC_128	15	Barcode type UCC-128
UPC_A	2	Barcode type UPC-A
UPC_E0	0	Barcode type UPC-E0
UPC_E1	1	Barcode type UPC-E1
WLNOSYMBOLOGY	99	No barcodes acceptable for input
BCDISABLED	1	All barcodes not defined in configuration are disabled
BCENABLED	2	All barcodes not defined in configuration are enabled
NO_DEFAULT	0	All barcodes not defined remain unchanged

WaveLinkConstants Constants

The constants in the class WaveLinkConstants are used for controlling entries in widget fields.

Table 3: *WaveLinkConstants Constants*

Constants	Value	Description
AUTOSIZE	0	Widget is sized according to text size
FIELD_ALPHA	0x0000080	Accepts only alphabetic characters
FIELD_ASCII	0x0010000	Accepts only low-range ASCII characters
FIELD_DISABLEKEY	0x0040000	Normal key entry is disabled
FIELD_DISABLESCAN	0x0020000	Scanning is disabled
FIELD_FORCEENTRY	0x0000040	Field must be populated before user can complete entry
FIELD_LOWER	0x8000000	Forces lowercase characters in a field
FIELD_MULTILINE	0x2000000	Allows multiple lines in a field
FIELD_NOECHO	0x0080000	Text entered is not shown in the field
FIELD_NUMERIC	0x0000100	Accepts only numbers
FIELD_PASSWORD	0x0008000	Shows asterisks instead of characters
FIELD_UPPER	0x0400000	Forces uppercase characters in a field
FIELD_WANTRETURN	0x0004000	Field allows CRLF
RADIO_ALIGNVERT	0x0000008	Radio button list is aligned vertically

WaveLinkError Constants

The error constants are used by all the Studio Java Development Library objects. See the specific methods and properties to determine their associated error codes.

Table 4: *WaveLinkError Constants*

Constants	Value	Description
WLBARCODELIMITEXCEEDED	11	Exceeded number of allowable barcode configurations
WLBCADDERROR	9	Failure adding barcode configuration
WLBCCLEARERROR	10	Failure clearing barcode config list
WLBCPARSEFAILURE	8	Failure parsing barcode file
WLCOMMERROR	1	Communication error

Table 4: *WaveLinkError Constants*

Constants	Value	Description
WLDTOUTOFSYNC	17	Terminal Date/Time is out of sync by more that 5 seconds
WLERROR	-2	Error occurred
WLFUNCTIONFAILED	7	RF function call failed at device
WLINNVALIDRETURN	6	Invalid RF function return packet
WLINVALIDPARAMS	5	Invalid function parameters
WLIOQUEUEERROR	14	IO queue buffer error
WLMEMORYERROR	2	Memory allocation error
WLNOERROR	0	No error
WLNOINPUTTYPE	15	Unable to determine input type
WLNOTINITIALIZED	3	RF connection not initialized
WLPORTTIMEOUT	16	Aux port timeout
WLREQUESTFAIL	4	Input request error
WLTIMEOUT	18	RFInput timeout has occurred
WLTONEADDERROR	12	Failure adding tone configuration
WLTONECLEARERROR	13	Failure clearing tone config list
WLUNKNOWNERROR	-1	Unknown error occurred

WaveLinkFont Constants

The WaveLinkFont constants define font characteristics on widgets.

Table 5: *WaveLinkFont Constants*

Constants	Value	Description
STYLE_BOLD	1	Bold font style
STYLE_ITALICS	2	Italics font style
STYLE_STANDARD	0	Standard font style
STYLE_STRIKEOUT	8	Strikeout font style
STYLE_UNDERLINE	4	Underlined font style

WaveLinkIO Constants

The WaveLinkIO constants determine how basic input and output methods function over a WaveLink wireless network.

Table 6: *WaveLinkIO Constants*

Constant	Value	Description
WLCLEAR	4	Clear screen before output
WLCLREOLN	8	Clear to end of line
WLCLREOS	32	Clear to end of screen
WLFLUSHOUTPUT	16	Flush the output buffer immediately
WLIGNORE_WIDGETS	0	Do not push widgets into *.scr file when Push/Pull screen is called
WLINCLUDE_WIDGETS	1	Include widgets in the *.scr file when the Push/Pull screen is called
WLNORMAL	1	Normal video mode
WLREVERSE	2	Reverse video mode
WLCOMMANDTYPE	2	Command key entered
WLKEYTYPE	4	Normal key entered
WLMENUBARTYPE	6	Input from a menubar widget
WLPOPUPTRIGGER	7	Input from a popup trigger widget
WLSCANTYPE	5	Scanned data entered
WLTIMEDOUT	9	RFInput input prompt expired
WLWIDGETTYPE	8	Input from a widget
WLCAPSLOCK	64	Set caps lock for input
WLNORMALKEYS	0	Normal keyboard mode
WLALPHA_ONLY	0x000080	Accept alpha characters only
WLBACKLIGHT	0x000200	Enable display backlight
WLCLR_INPUT_BUFFER	0x002000	Clear the input buffer of extra data
WLDISABLE_FKEYS	0x000020	Disable function key input
WLDISABLE_KEY	0x000002	Disable keyboard input
WLDISABLE_SCAN	0x000001	Disable scanner on input

Table 6: *WaveLinkIO Constants*

Constant	Value	Description
WLECHO_ASTERISK	0x008000	Echo asterisk for each character
WLFORCE_ENTRY	0x000040	Do not return with empty input field
WLINCLUDE_DATA	0x000800	Include data with function key entry
WLIGNORE_CRLF	0x004000	Do not break packages on <CR> or <LF>
WLIGNORE_KEY	0x000002	Disable the normal keys for the current input
WLMAXLENGTH	0x000400	Do not accept more than *nLength* characters
WLNO_NONPRINTABLE	0x010000	Suppress non-printable characters
WLNO_RETURN_BKSP	0x000010	Do not return on backspace
WLNO_RETURN_FILL	0x000008	Do not return when input prompt length is reached
WLNUMERIC_ONLY	0x000100	Accept numeric keys only
WLSOFT_TRIGGER	0x001000	Soft RF device trigger
WLSUPPRESS_ECHO	0x000004	Suppress input echo

WaveLinkMenu Constants

The WaveLinkMenu constants format the WaveLinkMenu object's border.

Table 7: *WaveLinkMenu Constants*

Constant	Value	Description
WLBORDER	0	Automatically create a border around menu
WLNOBORDER	1	Does not create border around menu

WaveLinkTerminal Constants

The WaveLinkTerminal constants set and return mobile device options.

Table 8: *WaveLinkTerminal Constants*

Constant	Value	Description
CAPSLOCK	64	Keyboard mode is CAPSLOCK
NORMALKEYS	0	Keyboard mode is normal
HARDWARECURSOR	0	Use hardware input cursor
SOFTWARECURSOR	1	Use software input cursor
LITHIUMBATDEAD	0	Device's lithium battery is dead
LITHIUMBATGOOD	1	Device's lithium battery is good
LITHIUMBATNONE	2	Device's lithium battery is not present
MAINBATGOOD	0	Device's main battery is good
MAINBATLOW	1	Device's main battery is low
LRT	1	LRT series terminals
VRC_PRC	2	VRC/PRC series terminals
PDT	3	PDT series terminals
WS10XX	4	WS10XX series terminals
PDT68XX	5	PDT68XX series terminals
PDT61XX	6	PDT61XX series terminals
PALM_PILOT	11	Palm Pilot device
CE2740	27	2740 CE-based device
CE7200	72	7200 CE-based device
CE7540	75	7540 CE-based device
PERCON_FALCON	93	PSC Falcon device
N/A[a]	177	Symbol VRC5040
N/A[a]	860	Telxon 860
N/A[a]	960	Telxon 960
INTERMEC	5020	Intermec device
WINDOWS	8000	Windows device
PPC	8001	PPC device
HPC	8002	HPC device

Table 8: *WaveLinkTerminal Constants*

Constant	Value	Description
HPCPRO	8003	HPC Pro device
N/A[a]	8100	Intermec Stingray
N/A[a]	8200	Dolphin 7400
N/A[a]	8300	Unitec PT900
N/A[a]	8400	Symbol 81xx and Symbol 28xx
N/A[a]	8500	National Datacomputer
N/A[a]	8600	Fujitsu Teampad

a. Use the numeric values for these mobile devices.

WaveLinkWidget Constants

WaveLinkWidget constants describe the placement, state, and properties of widgets.

Table 9: *WaveLinkWidget Constants*

Constant	Value	Description
AUTOSIZE	0	Determine the widget extent automatically
BYCELL	2	Position using cell coordinates
BYPERCENTAGE	3	Position using percentage coordinates
BYPIXEL	1	Position using pixel coordinates
CENTERJUST	2	Center justify the label
LEFTJUST	0	Left justify the label
RIGHTJUST	1	Right justify the label
CHECKED	1	Checked checkbox state
UNCHECKED	0	Unchecked checkbox state
UNDETERMINED	2	Undetermined checkbox state
INITDISABLED	2	Initial state is disabled
INITHIDDEN	1	Initial state is hidden

Table 9: *WaveLinkWidget Constants*

Constant	Value	Description
INITSTANDARD	0	Initial state of the widget is standard (shown and enabled)
PBFIXED	0x000040	Size each box of push button equally
PBHORIZONTAL	0x000008	Position push button horizontally
PBPROPORTIONAL	0x000020	Size each box of push button to size of its text
PBVERTICAL	0x000010	Position a push button vertically
DOWNREPEATER	4	Down arrow repeater button
LEFTREPEATER	1	Left arrow repeater button
RIGHTREPEATER	2	Right arrow repeater button
UPREPEATER	3	Up arrow repeater button
ALLWIDGETTYPES	0	Include all widget types
WAVELINKBITMAP	9	Bitmap image widget
WAVELINKBUTTON	1	Button widget
WAVELINKCHECKBOX	6	Checkbox widget
WAVELINKFIELD	10	Field widget (an input box)
WAVELINKHOTSPOT	2	Hotspot widget
WAVELINKLABEL	8	Label widget
WAVELINKMENUBAR	12	Menubar widget
WAVELINKPOPUP	3	Popup trigger widget
WAVELINKPUSHBUTTON	5	Push button widget
WAVELINKREPEATER	7	Repeater button widget
WAVELINKSELECTOR	4	Selector trigger widget

Index

D

E

F

O

P

Q

R

S

T

U

V

W

X

Y