Wireless Studio Application Files Specification

This document presents the xml tags that will be used to give format to the resources and forms displayed on an application developed with WireLess Studio Library.

Revision:
12/06/2009 First version
20/07/2009 Tags and properties were updated.

Table of contents

Introduction

For most programmers using an xml to describe the interface of an application is easier than write a lot of lines of code just to create a few buttons in a form.

This document defines the structure of an xml file needed to define the resources and forms used on an application written using the WireLess Studio Widgets Library (From now on WSWL).

Definition of the XML tags

The following example shows the basic composition of an xml file used to describe an application in the WireLess Studio Framework:

```
<?xml version="1.0" encoding="UTF-8" ?>
<wst_application version=”1.0.0”>
	<resources>
		<icons>
			<icon name=”exit” id=”1” dll_index=”WS_EXIT”
				dll_name=”IconLib.dll”/>
			<icon name=”write” id=”2” dll_index=”WS_WRITE”
				dll_name=”IconLib.dll”/>
		<icons/>
		<files>
			<file name=”CarBitmap” local_path=”c:\car.bmp”
				remote_path=”\Files\car.bmp”/>
			<file name=”datafile” local_path=”c:\data.csv”
				remote_path=”\Files\ data.csv”/>
		<files/>
		<fonts>
			<font name=”LargeBoldTahoma” id=”1”
				face_name=”Tahoma”  bold=”true”/>
		<fonts/>
	<resources/>

	<form name="MainForm">
		<coordinates  type=" WS_FORM_CT_BYCELL "/>
		<widgets>
			<button name="ExitButton" id="200">
				<text value="Exit"/>
				<position x=”30” y=”50”/>
				<size width=”30” height=”20”/>
				< icon name="exit"/>
			</button>
			<menu name="MainMenu" id="202">
				<menu type="WS_BUTTON_MENU_TYPE"/>
				<position x=”10” y=”10”/>
				<size width=”50” height=”70”/>

				<options>
					<option text=”Load”/>
					<option text=”Write” icon=”write”/>
					<option text=”Connect”/>
				</options>
			</menu>
		</widgets>
	</form>
</wst_application>
```

As you can see the main node (<wst_application>) contains two basic nodes:

The resources node (<resources>): This node contains the icons, fonts and files that will be used and should be transferred to the mobile unit.

A form node: This section describes a form and contains the widgets that will be displayed.

Each basic node can be used more than once to define several forms and resource sections.

Both nodes will be detailed in the following sections.

Resources node

Any application written using the WireLess Studio Widgets Library has access to icons, bitmap, data and scanner configuration files that are usually used by the widgets created for each application.

In order to describe and detail each of these resources three containers nodes are used.

Icon's node (<icons>)

This node contains the icons that can be used by the WS_BUTTON, WS_MENU and WS_LIST_CTRL widgets.

Each icon must be specified using the following tag

<icon name="exit" id="1" dll_index="WS_EXIT" dll_name="IconLib.dll"/>

Attributes of the <icon> tag

- name: Used to reference a specific icon by a unique identifier. This parameter is mandatory.
- id: The unique identifier for an icon used by the WSWL (See LoadExternalIcon() for more details).
- dll_index: A raw index of the icon in the dll. This attribute can be one of the strings listed here if the dll used is IconLib.dll.
- dll_name: The dll where the icon will be extracted from.

All the properties of this tag are mandatory, except for attribute "name".

File's node (<files>)

The WS_BITMAP and WS_LIST_CTRL use or can use external files such as bitmap files or data files. Each of these files can be specified using the following tag

<file name="CarBitmap" local_path="c:\car.bmp" remote_path="\Files\car.bmp"/>

Attributes of the <file> tag

- name: Used to reference a specific file by a unique identifier. This parameter is mandatory.
- local_path: The location of the file in the local file system. If it's not specified, the WSWL assumes the file is currently stored in the mobile unit.
- remote_path: The location of the file in the file system of the mobile unit. If the local_path attribute was specified, the file will be transferred to this location. This parameter is mandatory.
- overwrite: A Boolean value that indicates if the file should be overwritten.

Font's node (<fonts>)

This node contains the custom fonts created by the user to change the appearance of many widgets. Each font must be specified using the following tag.

<font name="LargeBoldTahoma" id="1" height="20" italic="false"
bold="true" underlined="false" face_name="Tahoma" alignment="WS_CENTER"/>

Attributes of the <font> tag

Almost all properties of the font tag corresponds to a parameter of the CreateCustomFont() method of the WStFactory class.

- name: Name of font, used to identify a defined font in an easy way. This parameter is mandatory.
- id: Corresponds to the usId parameter of the CreateCustomFont() method. This attribute is mandatory.
- height: Corresponds to the usHeight parameter of the CreateCustomFont() method. This parameter is mandatory.
- italic: Corresponds to the bItalic parameter of the CreateCustomFont() method. Default value: "false".

- bold: Corresponds to the bBold parameter of the CreateCustomFont() method. Default value: “false”.
- underlined: Corresponds to the bUnderlined parameter of the CreateCustomFont() method. Default value: “false”.
- face_name. Corresponds to the sFaceName parameter of the CreateCustomFont() method. This attribute is mandatory.
- Alignment: Corresponds to the usAlignment parameter of the CreateCustomFont() method. Default value: WS_LEFT.

Form node

This node describes a form and the widgets attached to it. Each form node is composed by several property nodes and one <widgets> node.

The following is an example of a <form> node

```
<form name="MainForm">

        < coordinates type="WS_FORM_CT_BYCELL"/>
        < function_keys enabled="false" map_arrow_keys=”false”/>
        < focus widget="ExitButton"/>
        <background color="#FF0000"/>
        <wait_cursor visible="true"/>

        <property name="map_arrow_keys” value="false"/>
        <background color="#FF0000"/>

        <widgets>
                <button name="ExitButton" id="200">
                        <text value="Exit"/>
                        <position x=”30” y=”50"/>
                        <size width=”30”  height=”20"/>
                </button>
        </widgets>
</form>
```

Attributes of the <form> tag

The supported attributes for a form node are:

- name: Identifies a form using a string.
- Id: Identifies a form using a number.

At least one of these attributes must be specified. For more information see CreateForm().

Properties of the <form> tag

The list of supported properties is:

- coordinates: The “type” attribute sets the coordinates type used by this form. Default Value: “WS_FORM_CT_BYCELL”. See also SetCoordType().
- function_keys:
    - Attributes:
        - enabled: enables or disabled the use of function keys. Default Value: “true”. See also EnableFunctionKeys().
        - map_arrow_keys: Specifies if the arrow keys must be mapped to function keys. Default Value: “false”. See also MapArrowKeysToFunctionKeys().
- focus: The attribute “widget_id” / “widget_name” changes the focus on a specific widget once the form is loaded. This property is optional. See also SetFocus().
- background: the attribute “color” sets the background color used by a specific form. The color must follow the format #XXXXXX, where “X” is a hexadecimal digit. This parameter is optional and its default value is specific of each platform where the client is running.
- wait_cursor: If the attribute “visible” is set to true, an hourglass cursor is displayed when the form is loaded. It can be hidden by calling the ShowWaitCursor() method. Default Value: “false”.

Widgets container node (<widgets>)

The widgets attached to a form must be defined here. The definition of each supported widget is explained below.

Widget node

Every visible widget supported by the WSWL can be defined and attached to a form using a widget node.

Example:

```
<button name="ExitButton" id="200">
        <text value="Exit"/>
        <position x=”30” y=”50"/>
```

```
        <size  width=”30”  height=”20"/>
</ button>
```

Widget node attributes

The supported widget nodes are: <button>, <bitmap>, <checkbox>, <field>, <html_ctrl>, <label>, <line>, <menu>, <list_ctrl>, <progress_bar> and <radio_button>

The supported attributes for a widget node are:

- name: Identifies a widget using a string.
- Id: Identifies a widget using a number.

At least one of these attributes must be specified. For more information see CreateWidget().

Widget properties

Each widget has common, shared and unique properties, the common properties are related with size, location, font and colors among others. The unique properties are related with the specific behavior of each widget. When some widgets have similar behaviors these can be configured using shared properties.

Widget common properties

This is an example of the common properties shared by all widgets in the WSWL:

```
<button name="ExitButton" id="201">
        <border visible=”false”/>
        <background color="#00FF00"/>
        <enabled value="true"/>
        <font name=”LargeBoldTahoma”/>
        <position x="30" y="40"/>
        <size width="30" height="40"/>
        <text value="Exit" color="#550011"/>
        <tab_stop enabled="true"/>
        <visible value="false"/>
        <submit_mode enabled="false"/>
        <icon name="write"/>
</ button>
```

Each common property is detailed below:

- border: The attribute “visible” indicates if the widget’s border should be displayed. This parameter is optional. See also ShowBorder().
- background: The attribute “color” is the background color used by a specific widget. The color must follow the format #XXXXXX, where “X” is a hexadecimal digit. This

parameter is optional and its default value is specific of each platform where the client is running.

- enabled: Specifies if a widget is enabled. This property is optional. Default Value: “true”. See also Enable().
- font: The name of a font previously defined on the resources sections. This property is optional. See also SetFont().
- position: A point defined by the attributes “x” and “y”. This property is mandatory. See also SetPosition().
- size: This property is defined by its two attributes “width” and “height”. This property is mandatory. See also SetSize().
- tab_stop: The attribute “enabled” specified if a widget can receive the keyboard focus. The tab order of a form is specified by the order in which its widgets are defined in the file. For more information see EnableTabStop() and GetNextWidget().
- text:
    - Attributes:
        - text: The text displayed by the widget. This property is optional. See also SetText().
        - color: The color of the text displayed by the widget. The color must follow the format #XXXXXX, where “X” is a hexadecimal digit. E.g.: #FF0000 is Red, #0000FF is blue. This parameter is optional and its default value depends on the platform in which the client is running. See also SetTextColor().
- visible: Indicates if a widget is visible or not. This property is optional. Default Value: “true”. See also Show().

Widget unique properties

Each widget has unique properties related with its specific behavior and features. The list of properties for each widget is detail below.

WS_BITMAP unique properties

The image displayed by a WS_BITMAP can be set using the “image” property. Its attribute “file” must reference one of the files defined in the resources section. This property is optional, for more information see SetImageFile().

Example of a WS_BITMAP widget:

```
<bitmap name="MainImage" id="200">
        <position x="0" y="50"/>
        <size width="10" y="20"/>
        <image file="CarBitmap"/>
</ bitmap>
```

WS_BUTTON unique properties

Example of a WS_BUTTON widget:

```
<button  name="ExitButton" id="201">
        <position x="30" y="40"/>
        <size width="30" height="40"/>
        <text value="Exit" color="#550011"/>
        <tab_stop enabled="true"/>
        <submit_mode enabled="false"/>
        <icon name="write"/>
        <flat_style enabled="true"/>
</ button>
```

Properties:

- submit_mode: Enables or disables the submit mode. Default value: “false”, for more information see EnableSubmitMode().
- icon: The property “name” references an icon previously defined in the resources sections. This parameter is optional, for more information see SetIcon().

WS_FIELD unique properties

Example:

```
<field name="ExitButton" id="202">
        <position x="30" y="40"/>
        <size width="30" height="40"/>
        <display_mode flags="WS_FIELD_DM_ECHO_ASTERISK
                WS_FIELD_DM_MULTILINE"/>
        <input_devices flags="WS_FIELD_ID_DISABLE_SCAN
                WS_FIELD_ID_DISABLE_KEY"/>
        <input_mode flags="WS_FIELD_IM_ALPHA_ONLY
                 WS_FIELD_IM_NUMERIC_ONLY"/>
        <len_ctrl_mode flags="WS_FIELD_LC_FORCE_ENTRY
                 WS_FIELD_LC_NO_CR_IF_EMPTY"/>
        <keyboard_mode flag="WS_FIELD_KM_ALPHABETIC"/>
        <max_length value="3"/>
        <scanner_config file="scannerConfiguration"/>

</ field >
```

Properties:

- display_mode: This property is composed by a space separated list of strings E.g.: “WS_FIELD_DM_ECHO_ASTERISK WS_FIELD_DM_MULTILINE”. This parameter is optional, for more information see ModifyDisplayMode().

- input_devices: This property is composed by a space separated list of strings E.g.: “WS_FIELD_ID_DISABLE_SCAN WS_FIELD_ID_DISABLE_KEY**”.** This parameter is optional, for more information see ModifyInputDevices().

- input_mode: This property is composed by a space separated list of strings E.g: “WS_FIELD_IM_ALPHA_ONLY WS_FIELD_IM_NUMERIC_ONLY**”.** This parameter is optional, for more information see ModifyInputMode().

- len_ctrl_mode: This property is composed by a space separated list of strings E.g: “WS_FIELD_LC_FORCE_ENTRY WS_FIELD_LC_NO_CR_IF_EMPTY**”.** This parameter is optional, for more information see ModifyLenCtrlMode().

- keyboard_mode: The current keyboard mode, E.g.: “WS_FIELD_KM_ALPHABETIC”. This parameter is optional, for more information see SetKeyboardMode().

- max_length: Sets maximum number of characters that can be typed. This parameter is optional, for more information see SetMaxLength().

- scanner_config: The attribute “file” sets the scanner configuration file. This parameter is optional, for more information see SetScannerConfigFile().

WS_HTML_CTRL unique properties

Example:

```
<html_ctrl name="htmlCtrl" id="203">
        <position x="70" y="40"/>
        <size width="30" height="40"/>
        <url address="http://www.softogo.com"/>
</ html_ctrl>
```

Properties:

- url: The attribute “address” sets the displayed page. This parameter is optional, for more information see SetUrlAddress().

WS_LABEL unique properties

Example:

```
<label name="label" id="204">
```

```
        <position x="70" y="40"/>
        <size width="30" height="40"/>
        <battery_life_percent visible="false"/>
        <current_time format="WS_TIME_FORMAT_NONE"/>
</ label>
```

Properties:

- battery_life_percent: The attribute “visible” indicates if the current battery life percent should be displayed. Default value: “false”, for more information see DisplayBatteryLifePercent().
- current_time: The attribute “format” specifies if the current time should be displayed on a label and in which format. Default value: “WS_TIME_FORMAT_NONE”, for more information see DisplayCurrentTime().

WS_LINE unique properties

Example:

```
<line id="205">
        <position x="70" y="40"/>
        <size width="30" height="40"/>
        <line width="1" orientation="WS_ORIENTATION_HORIZ"/>
</ line>
```

Properties:

- line
    - Attributes:
        - width: Specifies the line width in pixels. Default value: “1”. See also SetLineWidth().
        - orientation: The orientation of the line. Default value: “WS_ORIENTATION_HORIZ”, for more information see SetOrientation().

WS_LIST_CTRL unique properties

Example:

```
<list_ctrl name="listCtrl" id="206">
        <position x="10" y="20"/>
        <size width="50" height="50"/>
        <columns>
                <column text="Index" width="30"/>
                <column text="Book" width="35"/>
```

```
        </columns>
        <rows>
                <row icon="exit">
                        <item text="331X32"/>
                        <item text="The Moon"/>
                </row>
                <row icon="write">
                        <item text="23KJF"/>
                        <item text="The Sun"/>
                </row>
        </rows>
</ list_ctrl>
```

Properties:

- columns: This node contains the definition of each column on this control. Each column is described by a <column> tag as you can se below.

  ```
  <column text="Index" width="30"/>
  ```

  The “text” attribute is displayed on the column header of the list ctrl. The “width” attribute specifies the size of the column.

  See also: AddColumn()

- rows: The rows displayed by a list control are specified here. Each row is composed by a list of items. E.g.:

  ```
  <rows>
          <row icon=”House” >
                  <item text=”27/03/2209”/>
                  <item text=”1000”/>
          <row/>
          <row icon=”Box” >
                  <item text=”15/03/2209”/>
                  <item text=”2456”/>
          <row/>
  <rows/>
  ```

  The “icon” attribute of the row tag specifies the name of an icon defined in the resources section. This attribute is optional.

  The “text” attribute indicates the text that will be displayed by the item. This attribute is mandatory.

  This node container is optional. See also AddRow().

- data: Instead of defining on the form file the rows that will be displayed on a list ctrl, they can be loaded from a data file previously defined in the resources section.

  Example:

```
<files>
        <file name=" Data1" localPath="c:\data.csv" remotePath="\Files\ data.csv"/>
<files/>
…
< data file="Data1">
```

WS_MENU unique properties

Example:

```
<menu name="list Box" id="207">
        <position x="0" y="10"/>
        <size width="40" height="10"/>
        <menu type="WS_BUTTON_MENU_TYPE"/>

        <options>
                <option icon="write" text="jperez@softogo.com"/>
                <option icon="exit" text=" jdoe@softogo.com"/>
        </options>
</ menu>
```

Properties:

- menu: The attribute "type" sets the menu type. Default value: "WS_BUTTON_MENU_TYPE". See Also: ChangeMenuType().

- options: This node contains the list of options that will be displayed. Each option is defined by <option> tag.

  Each option has two properties: The "icon" property specifies the name of an icon defined in the resources section and it's optional. The "text" attribute specifies the text displayed in the screen. See Also AddOption().

WS_PROGRESS_BAR unique properties

Example:

```
<progress_bar name="progBar" id="208">
        <position x="0" y="10"/>
        <size width="40" height="10"/>
        <bar color="#550011" position="0" min_value="0" max_value="100" step="1"/>
</progress_bar>
```

Properties:

- bar
  - Attributes:

- color: Specifies the color of the bar. This parameter is optional. See also SetBarColor().
- position: Sets the position of the bar when is displayed. Default value: “0”. See also SetBarPosition().
- min_value and max_value: Set the limits of the bar. See also SetRange().
- step: Sets the step increment. Default value: “1”. See also SetStep().

WS_RADIO_BUTTON unique properties

Example:

```
< radio_button name="radioButton" id="209">
        <position x="0" y="10"/>
        <size width="40" height="10"/>
        <group id="1"/>
</ radio_button>
```

Properties:

- group: The “id” attribute specifies the group id for a radio button. Default value: “0”, See also SetGroupId().

Widget shared properties

Some properties are shared by several widgets. The list of shared properties is explained below.

WS_MENU and WS_ BUTTON shared properties

Example:

```
<button name="ExitButton" id="201">
        <background color="#00FF00"/>
        <enabled value="true"/>
        <position x="30" y="40"/>
        <size width="30" height="40"/>
        <text value="Exit" color="#550011"/>
        <flat_style enabled="true"/>
</ button>
```

Properties:

- flat_style: The attribute “enabled” of this property enables or disables the flat style. Default value: “false”, for more information see EnableFlatStyle().

WS_CHECK_BOX and WS_RADIO_BUTTON shared properties

Example:

```
< checkbox name="checkBox" id="210">
        <position x="0" y="10"/>
        <size width="40" height="10"/>
        <style value="WS_STYLE_NORMAL"/>
        <state value="WS_STATE_CHECKED"/>
</ checkbox>
```

Properties:

- style: The style used by a widget. Default value: WS_STYLE_NORMAL, for more information see ModifyStyle().
- state: The initial state of the widget. Default value: WS_STATE_UNCHECKED, for more information see SetState().

WS_LIST_CTRL and WS_MENU shared properties

Example:

```
<menu name="menu2" id="211">
        <position x="0" y="10"/>
        <size width="40" height="10"/>
        <options>
                <option text="Load"/>
        </options>
        <selected_item index="0"/>
</ menu>
```

Properties:

- selected_item: The index property sets the selected item. This property is optional.

  See also SetSelectedItem().