# Lightweight UI Toolkit

## Resource File Specification

## Version 1.0

Part No. 06-30-2010

Please
Recycle

# Resource File Specification

This document describes the Lightweight UI Toolkit resource file format.



## Resources

The resource file is organized in chunks. Each chunk type is described in this document. The first chunk in the resource file MUST be the Header chunk type which describes the resource file.

Notice that while the resource files allow room to store meta data internally, it is important to limit meta-data storage to avoid size increases and exposure of information. The resource files are designed for shipping with an application, therefore they should remain compact, with only the meta data required by the end user remaining within the application.

Resource files are designed for reading and writing using Java-based tools and are designed around the `DataInputStream` and `DataOutputStream` architecture. This implies several things regarding the specification and the file format:

- UTF elements in the specification refer to the `readUTF` and `writeUTF` methods of `DataInputStream` and not to C-style UTF null-terminated strings.
- The file uses big endian to represent all types.

# General Format

**TABLE 1** General Format

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Number of Chunks > 1 | SHORT | 2 | |
| Chunks, first chunk must be of type header | See "Chunks" on page 3 | individual | For any resource |

## Chunks

**TABLE 2** Chunks

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Type of Resource<br>• Theme: 0xF2<br>• Image: 0xFD<br>• Font: 0xFC<br>• L10N: 0xF9<br>• Data: 0xFA<br>• Header: 0xFF<br>Values from 0xE0 to 0xFF are reserved for future use | BYTE | 1 | |
| Resource Tag (name) | UTF | individual | |
| Resource data | See respective section in this document. | individual | |

# Header

The header must be the first chunk of the file allowing tools and the application to identify details about the resource file and provide tools for future extensibility.

The current version of the specification has major version 1 and minor version 2.

**TABLE 3** Header

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Header Size | SHORT | 2 | |
| Major Version | SHORT | 2 | |
| Minor Version | SHORT | 2 | |
| Meta data count | SHORT | 2 | |
| Meta data Strings | UTF[] | Meta data count * individual | |

# Theme Resources

Themes are a set of key value pairs where the key represents a well known string using the format of `[`*ComponentUIID*`.]attribute`. The `ComponentUIID` names are use-defined. The attributes, however, are well known and determine the type of value stored in the resource file.

The attribute mapping in TABLE 4 applies to the theme:

**TABLE 4** Attribute Mappings

| Attribute | Type |
|---|---|
| `fgColor, bgColor, fgSelectionColor, bgSelectionColor` | Color |
| `font` | Font |
| `padding, margin` | Spacing |
| `transparency` | Transparency |
| `Background, selectionBackground` | Background |
| `border` | Border |

# Theme Resource Structure

**TABLE 5** Theme Resource Structure

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Number of Properties | SHORT | 2 | |
| Property Pair | See "Theme Property Pair" on page 5 | individual | For any property |

# Theme Property Pair

**TABLE 6** Theme Property Pair

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Property Key | UTF | individual | |
| Property Value | See respective table from Theme Property Values. | individual | |

# Theme Property Values



## Theme Property Value: Color

Color is expressed as an RGB value. The alpha component of the color is ignored regardless of its value.

**TABLE 7** Color

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Color value | INT | 4 | |

## Theme Property Value: Transparency

Represents an alpha transparency value between 0x00 - 0xff.

**TABLE 8** Transparency

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Transparency value | Byte | 1 | |

## Theme Property Value: Spacing

**TABLE 9** Spacing

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Top Spacing | BYTE | 1 | |
| Bottom Spacing | BYTE | 1 | |
| Left Spacing | BYTE | 1 | |
| Right Spacing | BYTE | 1 | |

## Theme Property Value: FONT

References a font chunk by name (see ).

**TABLE 10** Font

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| New Font | Boolean | 1 | |
| Font name | UTF | Individual | For new font true |
| Font face | Byte | 1 | For new font false |
| Font style | Byte | 1 | For new font false |
| Font size | Byte | 1 | For new font false |

# Background

Represents the background drawing for a component, image behavior, gradient, etcetera.

Radial gradients have a center position for the core of the radial effect. This center is determined using the relative x/relative y variables. These variables are in the range of 0.0 to 1.0 and they represent the position within the component relative to its size, where 0.5/0.5 is the exact center of the component. The radial gradient will be drawn exactly in that location.

**TABLE 11** Background

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Type<br>Scaled Image: 0xF1<br>Tiled Vertically Image: 0xF2<br>Tiled Horizontally Image: 0xF3<br>Tiled Both Image: 0xF4<br>Aligned Image 0xF5<br>Horizontal Linear Gradient: 0xF6<br>Vertical Linear Gradient: 0xF7<br>Radial Gradient: 0xF8 | BYTE | 1 | |
| Image | UTF (Image Reference id) | individual | Only for an image related type 0xf1-0xf5 |
| Alignment<br>Top: 0xf1<br>Bottom: 0xf2<br>Center: 0xf3<br>Left: 0xf4<br>Right: 0xf5 | Byte | 1 | Only for tiled vertically/horizontally & aligned images. |
| Start Color | Int | 4 | Only for gradient types |
| End Color | Int | 4 | Only for gradient types |
| Relative X | float | 4 | Only for gradient types |
| Relative Y | float | 4 | Only for gradient types |
| Relative Size | float | 4 | Only for gradient types |

# Theme Property Value: Border

The border property allows the definition of a component border, corresponding to the behavior of the LWUIT Border class. This includes some built in border types which can be augmented in future LWUIT releases.

Every border type has a different set of variables to indicate its appearance.

Some borders support user determined colors as an option, the border can receive a flag indicating whether the color be extracted from the theme or should be specified specifically for this border instance.

## *Border Structure*

**TABLE 12** Border Structure

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Border Type:<br>None: 0xff01<br>Line: 0xff02<br>Rounded: 0xff03<br>Etched Lowered: 0xff04<br>Etched Raised: 0xff05<br>Bevel Lowered: 0xff06<br>Bevel Raised: 0xff07<br>Image: 0xff08 | Short | 2 | |
| Data | Border Data | Individual (no content in the case of border "None") | |

## *Border Data: Line*

**TABLE 13** Border Data: Line

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Theme Colors | Boolean | 1 | |
| Thickness | Byte | 1 | |
| Color | int | 4 | Only if theme color is false |

## *Border Data: Rounded*

**TABLE 14** Border Data: Rounded

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Theme Colors | Boolean | 1 | |
| Arc width | Byte | 1 | |
| Arc height | Byte | 1 | |
| Color | int | 4 | Only if theme color is false |

## *Border Data: Etched Lowered or Raised*

**TABLE 15** Border Data: Etched Lowered, Bevel Raised

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Theme Colors | Boolean | 1 | |
| Highlight Color | int | 4 | Only if theme color is false |
| Shadow Color | int | 4 | Only if theme color is false |

## *Border Data: Bevel Lowered or Raised*

**TABLE 16** Bevel Lowered, Bevel Raised

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Theme Colors | Boolean | 1 | |
| Highlight Outer Color | int | 4 | Only if theme color is false |
| Highlight Inner Color | int | 4 | Only if theme color is false |
| Shadow Outer Color | int | 4 | Only if theme color is false |
| Shadow Inner Color | int | 4 | Only if theme color is false |

## *Border Data: Image*

An image border supports two patterns: nine images, or three images. The last image of a pattern may be null, therefore the border can also support patterns of eight images or two images.

**TABLE 17** Border Data: Image

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Image Count | Byte | 1 | |
| Images | UTF (Reference to Image Chunk) | Individual * Image Count | |

# Image Chunk

An image resource also represents a simple animation within the resource file. There are several different image types supported within the image chunk. See TABLE 18.

**TABLE 18** Image Types

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Image Type<br>PNG: 0xf1<br>JPEG: 0xf2<br>Indexed: 0xf3<br>Animation: 0xf4<br>SVG: 0xf5 | Byte | 1 | |
| Image | | individual | |

# Image PNG/JPEG

**TABLE 19** PNG or JPEG Image

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Length of image data | INT | 4 | |
| Image data | | individual | |

# Indexed

Indexed images are bitmaps based on a palette lookup value.

**TABLE 20** Indexed Image

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Color palette size | BYTE | 1 | |
| Color | INT[] | 4 | For any color in palette (0 indicates 256 colors!) |
| Image width | SHORT | 2 | |
| Image height | SHORT | 2 | |
| Color value per image pixel | BYTE | 1 | For any pixel of the image |

# Animation

**TABLE 21** Animation

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Color Palette Size | BYTE | 1 | |
| Color | INT[] | 4 | For any color in palette (0 indicates 256 colors!) |
| Animation Width | SHORT | 2 | |
| Animation Height | SHORT | 2 | |
| Number of Animation Frames | BYTE | 1 | |
| Total Animation Time | INT | 4 | |
| Animation Loop indicator | BOOLEAN | 1 | |
| Animation Frame Data | See Animation Frame Data | Individual | For any frame of the animation |

## Animation Frame Data

Animation Frame Data represents a frame within the animation. The number of frames depends on the complexity of the animation. Frames should be read one by one with a special case for the first frame, which is always just a bitmap.

**TABLE 22** Animation Frame Data

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| First Frame | First Frame | Individual | 1 |
| Frame | Frame | Individual | Number of Animation Frames - 1 |

## First Frame

The first frame of an animation is always a key frame and does not need a time stamp. It is a palette bitmap of the first image within the animation.

**TABLE 23** Key Frame

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Color Palette Index | BYTE[] | 1 | Animation Width * Animation height |

## Frame

The key frame paints the entire animation. It is a bitmap and is thus expensive in memory/storage.

**TABLE 24** Frame

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Time Stamp | INT | 4 | |
| Key Frame Indicator | Boolean | 1 | |
| Keyframe (palette indexes) | byte[] | Animation Width * Animation height | Only if keyframe indicator is true. |
| Previous Frame Drawing | Boolean | 1 | Only if keyframe indicator is false |
| Changed Rows | Changed Rows | individual | Only if keyframe indicator is false |

## Changed Rows

**TABLE 25** Changed Rows

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Row Offset | Short | 2 | Iterate over rows until row offset is -1 |
| Row Data (palette indexes) | Byte[] | Animation Width | |

# SVG

An SVG image can be rendered on devices that have built-in SVG image support. When no such support is available the fallback image is shown. Implementations that aren't targeted for non-SVG environments can set the fallback length to 0. A fallback has a width/height floating point ratio representing the ratio of the SVG fallback image to the screen size. This ratio allows tools to adapt the resource file for multiple screen resolutions while preserving a relative image ratio.

The base URL is used to look up resources required by the SVG parser. If the URL is an empty string the implementation can look within the resource file itself or the classpath.

Notice that the fallback image data can be 0, indicating no fallback image.

**TABLE 26** SVG

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Length of SVG File | int | 4 | |
| SVG data | byte[] | individual | |
| Base URL | UTF | individual | |
| Animated | boolean | 1 | |
| Fallback Width | float | 4 | |
| Fallback Height | float | 4 | |
| Length of fallback image data | int | 4 | |
| Fallback Image data | byte[] | individual | |

# Font Resources

A font contains a set of fallback optional values ordered by priority and always ending with a system font definition. If a platform doesn't or shouldn't support a given font type it moves to the next available font in the chain.

There are 4 font types, some of which might not be supported on a given platform. The only required font type is the system font, which must work correctly for all platforms or character encodings and serves as a fallback:

- **System Font**: Defines a font based on a few lowest common denominator attributes that would work on all devices.
- **Bitmap Font**: Represents a bitmap and charset mapping to allow drawing the font in some platforms that don't support TrueType fonts or a lookup table.
- **TrueType**: Supports embedding a TrueType font file into the resource file
- **Lookup**: A string representing a font using a platform-specific font lookup syntax. For example, Arial-Bold-16. This is a comma-separated set of strings which allows LWUIT to try various strings based on font platform availability.

Fonts are loaded as a fallback chain. Each one of the font types is stored in order of quality. The first font that is stored in the file and supported by the platform is used. The order of font storage/picking is:

1. TrueType font
2. Lookup
3. Bitmap
4. System

**TABLE 27** Font Resources

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| `systemFontFallback`<br>Bitwise value:<br>• MONOSPACE: 32<br>• PROPORATIONAL: 64<br>• SYSTEM: 0<br>• BOLD: 1<br>• ITALIC: 2<br>• PLAIN: 0<br>• LARGE: 16<br>• SMALL: 8<br>• MEDIUM: 0 | BYTE | 1 | |
| `isTrueTypeFontIncluded` | Boolean | 1 | |
| TrueType size | int | 4 | Only if the TrueType font is included |
| TrueType data | byte[] | TrueType size | Only if the TrueType font is included |
| `isLookupIncluded` | Boolean | 1 | |
| Lookup font name | UTF | | Only if the lookup font is included |
| `isBitmapIncluded` | boolean | 1 | |
| Image | Image (see theme image section) | Individual | Only if the Bitmap font is included |
| Character Count | Short | 2 | Only if the Bitmap font is included |
| Cut Offsets | Short | Character Count | Only if the Bitmap font is included |
| Char Widths | byte | Character Count | Only if the Bitmap font is included |
| Charset | UTF | Individual | Only if the Bitmap font is included |
| Rendering hint | Byte | 1 | Only if the Bitmap font is included |

## *Font Instance*

The fonts in the TABLE 28 are organized by platform, which represents a descriptive set of names to identify a specific platform. For example, MIDP, PBP, RIM etcetera. The default platform fallback has a 0 length String and is attempted if no other platform string matches. Platforms are organized by priority. For example, when running on a RIM device, the RIM platform is searched first, then the MIDP platform.

Platform names are subject to change and extensions as new LWUIT platforms are added. A platform name must only include alpha numeric characters ([A-Z][a-z][0-9]) and is treated as case insensitive. Platform names may be comma-delimited to associate a single font with multiple platforms.

**TABLE 28** Font Instance

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Default System Font | Font System | 1 | |
| Table Size | byte | 1 | |
| Platform | UTF | Individual for every table element (table size) | |
| Font | UTF | Individual for every table element (table size) | |

A sample table can look like this:

```
System Font: BOLD
Table Size: 3
Platform: "RIM,MIDP"
Font: Bitmap Font (binary data)
Platform: "PBP"
Font: "Arial-Bold-14"
Platform: "HDTV"
Font: "Arial-Bold-30"
```

Since the table should be processed by priority, a TV platform should pick the 30 pixel font rather than the 14 pixel font, which should also be appropriate for PBP platforms. MIDP and RIM platforms can just ignore that font altogether.

Notice that the system font is the default and is always required as a valid fallback for all platforms.

## *Font Instance: System*

**TABLE 29** System Font Instance

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Bitwise value:<br>• MONOSPACE: 32<br>• PROPORATIONAL: 64<br>• SYSTEM: 0<br>• BOLD: 1<br>• ITALIC: 2<br>• PLAIN: 0<br>• LARGE: 16<br>• SMALL: 8<br>• MEDIUM: 0 | BYTE | 1 | |

## *Font Instance: Bitmap*

The bitmap font is essentially a horizontal PNG image with cutoff points. It is painted as shades of red that are manipulated by the font rendering code.

The font has a generating font property that allows editor tools to regenerate the font using the given font name, size, and style information encoded within that string.

The rendering hint variable is used to generate the bitmap font based on user settings such as anti-aliasing.

**TABLE 30** Bitmap Font Instance

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Image | Image (see theme image section) | Individual | |
| Character Count | Short | 2 | |
| Cut Offsets | Short | 2 * Character Count | |
| Char Widths | byte | Character Count | |
| Charset | UTF | Individual | |
| Rendering Hint | Byte | byte | |
| Generating Font | UTF | Individual | |

### *Font Instance: TrueType*

**TABLE 31** TrueType font Instance

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Size | int | 4 | |
| File | byte[] | Individual | |

### *Font Instance: Lookup*

**TABLE 32** Lookup Font Instance

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Lookup | UTF | Individual | |

# L10N Resources

**TABLE 33** L10N Resource

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Number of L10N Property Keys | SHORT | 2 | |
| Number of L10N Languages | SHORT | 2 | |
| L10N Property Key | UTF | Individual | For any L10N property |
| L10N Property Values | See TABLE 34. | Individual | For any L10N language |

**TABLE 34** L10N Property Values

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| L10N Language | UTF | Individual | |
| L10N Property Value for this language | UTF | Individual | For any L10N Property |

# Data Resources

**TABLE 35** Data Resource

| What | Type | Size (Byte) | Iteration factor |
|---|---|---|---|
| Data Length | INT | 4 | |
| Data | | Individual | |