LWUIT Resource File Specification

Version 1.0

Content

1 Resources

The resource file is comprised of chunks, each chunk type is described bellow. 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 for meta-data to be stored within, it is advised to make very limited use of such facilities to avoid size increases and exposure of information. The resource files are designed for shipping with an application and thus should remain compact with only the meta-data required by the end user remaining within the application.

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

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

1.1 General Format

What	Type	Size (Byte)	Iteration factor
Number of Chunks > 1	SHORT	2	
Chunks, first chunk must be of type header	See 1.2	individual	For any resource

1.2 Chunks

What	Type	Size (Byte)	Iteration factor
Type of Resource - Theme: 0xF2 - Image: 0xFD - Font: 0xFC - L10N: 0xF9 - Data: 0xFA - Header: 0xFF Values from 0xE0 to 0xFF are reserved for future use	BYTE	1	
Resource Tag (name)	UTF	individual	
Resource data	See Respective Chapter	individual	

2 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.

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	

3 Theme Resources

Themes are a set of key value pairs where the key represents a well known string using the format format of [ComponentUIID.]attribute. The ComponentUIID names are user defined the attributes however are well known and determine the type of value stored in the resource file.
The following attribute mapping applies to the theme:

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

3.1 Theme Resource Structure

What	**Type**	**Size (Byte)**	**Iteration factor**
Number of Properties	SHORT	2	
Property Pair	See 3.2	individual	For any property

3.2 Theme Property Pair

What	**Type**	**Size (Byte)**	**Iteration factor**
Property Key	UTF	individual	
Property Value	See Respective Sub-Chapter of 3.3	individual	

3.3 Theme Property Values

3.3.1 Theme Property Value: Color

Color is RGB color the alpha component of the color is ignored regardless of its value

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

3.3.2 Theme Property Value: Transparency

Represents alpha transparency value between 0x00 - 0xff

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

3.3.3 Theme Property Value: Spacing

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

3.3.4 Theme Property Value: FONT

References a font chunk by name

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

3.3.5 Background

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

Radial gradients have a center position for the core of the radial effect, this center is determined relatively using the relaitve x/relative y variables. These variables are in the range of 0 .0 to 1.0 and they represent the position within the component relatively to its size, where 0.5/0.5 is the exact center of the component. The radial gradient will be drawn exactly in that

What	Type	Size (Byte)	Iteration factor
Type Scaled Image: 0xF1 Tiled Vertically Image: 0xF2 Tiled Horizontally Image: 0xF3 Tiled Both Image: 0xF4 Aligned Image 0xF5 Horizontal Linear Gradient: 0xF6 Vertical Linear Gradient: 0xF7 Radial Gradient: 0xF8	BYTE	1	
Image	UTF (Image Reference id)	individual	Only for an image related type 0xf1-0xf5
Alignment Top: 0xf1	Byte	1	Only for tiled vertically/horizontally

Bottom: 0xf2 Center: 0xf3 Left: 0xf4 Right: 0xf5			& 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

3.3.6 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.

3.3.6.1 Border Structure

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

3.3.6.2 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

3.3.6.3 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

3.3.6.4 Border Data: Etched Lowered/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

3.3.6.5 Border Data: Bevel Lowered/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

3.3.6.6 Border Data: Image

An image border supports two patterns 9 images or 3 images, the last of those images may be null hence the border can also support 2 or 8 images.

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

4 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:

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

4.1 Image PNG/JPEG

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

4.2 Indexed

Indexed images are bitmaps based on a palette lookup value

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

4.3 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 4.3.1	Individual	For any frame of the animation

4.3.1 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

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

4.3.2 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.

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

4.3.3 Frame

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

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

4.3.4 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	

4.4 SVG

An SVG image can be rendered on devices that have builtin 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 lookup 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.

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	

5 Font Resources

A font contains a set of fallback optional values ordered by priority 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/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 true type font file into the resource file
- Lookup – A string representing a font using a platform specific font lookup syntax e.g. 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 where 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 will be used. The order of font storage/picking is:

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

What	Type	Size (Byte)	Iteration factor
systemFontFallback Bitwise value: - MONOSPACE : 32 - PROPORATIONAL : 64 - SYSTEM : 0 - BOLD: 1 - ITALIC: 2 - PLAIN: 0 - LARGE: 16 - SMALL: 8 - 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	Individual	Only if the Bitmap font is included

	image section)		
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

5.1.1.1 Font Instance

The fonts in the table are organized by platform which represents a descriptive set of names to identify a specific platform, e.g. MIDP, PBP, RIM etc. The default platform fallback will have a 0 length String and would be attempted if no other platform string matches. Platforms would organize by priority, e.g. when running on a RIM device the RIM platform would be 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.

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/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.

5.1.1.2 Font Instance: System

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

5.1.1.3 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/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

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

5.1.1.4 Font instance: TrueType

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

5.1.1.5 Font instance: Lookup

What	Type	Size (Byte)	Iteration factor
Lookup	UTF	Individual	

6 L10N Resources

6.1 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 6.2	Individual	For any L10N language

6.2 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

7 Data Resources

7.1 Data Resource

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