

This article is archived because it is not considered relevant for third-party developers creating commercial solutions today. The article is believed to be still valid for the original topic scope.



This article explains how WidSets platform differs from J2ME/MIDP, and what is needed when porting existing MIDP Java applications to WidSets Scripting Language (WSL) so that they can run on WidSets platform.

Contents

- [1 Code](#)
- [2 Variables](#)
- [3 Arrays](#)
- [4 Threads](#)
- [5 RMS](#)
- [6 UI Components](#)
- [7 Sounds](#)
- [8 Image Scaling](#)
- [9 Platform callbacks](#)
- [10 Parameters](#)
- [11 Communication](#)
- [12 Packaging](#)
- [13 Publishing](#)

Code

All widget logic must be in one compilation unit (class), so if your MIDlet has multiple classes they have to be all fitted in one .he file.

See WidSets Scripting Language for information about inner functions, closures, structs, tuples, and typedefs that make it easier to have everything in one file.

Variables

Only *boolean*, *char*, *int*, and *long* primitives are supported.

Arrays

Only byte and int arrays are supported. So if you have char, short, long, boolean, float, or double-arrays, you have to manage with `ByteArray`, `IntArray`, and Value.

Note: Java-style array constructor works only for `Value`. For `Byte` and `IntArray`s, `set()` function must be used.

```
ByteArray foo = new ByteArray(3).set(0, 'f', 'o', 'o');  
Value bar = ["bar" => "foo", "foo" => [1, 2, 3]];
```

Threads

WSL does not expose `Threads`, so only way to get timed callbacks is to use Timers.

RMS

Record Store System is replaced with more abstract Store. Each widget has it's own `Store` that can be accessed like a hashtable, `key=>value`. `Value` objects can be either Value structures or ByteArray.

UI Components

`lcdui` package is replaced by WSL UI Components. These components provide AWT/Swing type of functionality with events, action listeners, and scrolling, and yet the implementation is faster than `lcdui CustomItem` implementation on many devices.

Widgets are displayed in minimized mode on dashboard and in maximized mode (fullscreen) when they opened. Maximized mode consist of a stack of views (Shells) which contain one or more components.

Components are (`lcdui`/`SWT`/`Swing` counterpart)

- Shell (Frame)
- Flow (Container and `LayoutManager`)
- Label (single line text)
- Ticker (scrolling single line text)
- Text (multi-line text)
- HyperText (multi-line text with multiple styles and inline links)
- Input (text input using native `TextBox`)
- Picture (displays images)
- Prompt (modal dialog)
- Progress (modal waiting dialog)
- Popup (popup Frame)
- Menu (multi hierarchy menu)
- Camera (launches phone's camera to capture a photo)
- Canvas (free graphics rendering)
- Scrollable (single component container providing scrolling support to both directions when needed)

Porting_from_MIDP_to_WidSets_Scripting_Language

- View (statically planned set of components that can be defined in *widget.xml*)
- MenuBar (softkey area containing left, middle, and right key labels)

Each component can be styled (fonts, foreground/background colors, borders, frames, images) by using Styles.

Sounds

WidSets supports JSR-135 sound playing through Player.

Image Scaling

Automatic image scaling to match the device resolution is achieved by declaring images as scaled widget resource images. Scaling to 50%, 75%, and 100% are supported. See more.

Platform callbacks

Instead of MIDlet or Screen callbacks, widgets get called back to functions declared in Script. You can also declare a separate listener function for most functions mentioned here to make your code more elegant.

Parameters

Widgets can have server-stored parameters that can be changed on Web site, client-side, or by widget code. Supported parameter types are *string*, *number*, *url*, *password*, and *auth*.

Communication

Communicating with server-side is done by accessing services. Currently available services include the following:

- **http** (incoming content can be filtered at server-side using regex and xpath filters)
- **image** (supports PNG8/PNG24/JPEG, can be scaled on server-side)
- **syndication** (supports various RSS and ATOM formats)

Getting content with Services

Packaging

To release a widget, pack the *.he* code file, widget.xml, resource images, preview images (shown on the Web site and in mobile Library widget), and localization properties to a ZIP package. The package can then be uploaded with either **devkit upload** or **devkit run** commands, or by uploading the ZIP to www.widsets.com.

Publishing

You can publish your widget for other WidSets users to try and use by uploading it to www.widsets.com server, and publishing it there (select the widget on your Web Dashboard Manager and click the *Publish* button). You can then set descriptions, categories, languages, and tags that help users to find your widget in the WidSets Library. You can also define this information in your *widget.xml* so that you do not need to enter it every time you publish a new version.