

Sun Java™ Wireless Toolkit for CLDC

Create, Optimize, and Tune MIDP Applications — Quickly and Easily



Highlights

- Build applications with the push of a button
- Build MIDP applications with many combinations of optional APIs
- Choose customization options, emulator skins, and enhanced screen features
- Profile methods, monitor network and memory usage, and tune performance
- Integrate with popular Java IDEs

Awards

- Developer.com Wireless Development Product of the Year for 2004, 2005, and 2006
- Software Development Jolt Product Excellence Award for 2004 and 2005
- Java Pro Readers Choice for 2004
- JavaWorld™ Editor's Choice for 2003



The Sun Java™ Wireless Toolkit for Connected Limited Device Configuration (CLDC) (formerly the J2ME™ Wireless Toolkit) is a state-of-the-art, award-winning toolkit for developing wireless applications using the Java programming language. The toolkit includes build tools, a device emulator, and monitoring tools for developing applications targeted at Mobile Information Device Profile (MIDP) mobile devices. Hundreds of thousands of developers worldwide use the toolkit to create, optimize, and tune MIDP applications quickly and easily.

The Java Wireless Toolkit for CLDC functions as a standalone development environment, or it can be used with an integrated development environment (IDE), such as the NetBeans™ Mobility Pack for CLDC. In standalone mode, you can use KToolbar to set preferences, create deployable MIDlet descriptor and archive files, compile, test applications, and more. (A command-line interface is also available.) When using the Java Wireless Toolkit with an IDE, developers benefit from the IDE's capabilities, such as source-level debugging and GUI-building tools, including drag-and-drop MIDlets (available with NetBeans tools).

The Java Wireless Toolkit emulator, based on Java ME CLDC and MIDP reference implementations, is fully compliant with the Technology Compatibility Kit (TCK) for the supported Java Specification Requests (JSRs). This ensures that all Java APIs in the specifications are present and correct. The toolkit provides a selection of emulator skins for application testing, and adding your own emulator skin is easy.

Figure 1 shows the toolkit's flexibility. You can combine a target platform (MSA, JTWI, or custom) with a profile and a configuration. You can also create a project that's backwards compatible with MIDP 1.0 and CLDC 1.0.

Emulator and build environment support

Visit jcp.org

- JSR 248 — Mobile Service Architecture
- JSR 185 — Java Technology for the Wireless Industry
- JSRs 30 and 139 — Connected Limited Device Configuration 1.0 and 1.1
- JSRs 37 and 118 — Mobile Information Device Profile 1.0, and 2.0

Optional Packages

- JSR 75 — PDA Optional Packages
- JSR 82 — Java APIs for Bluetooth
- JSR 135 — Mobile Media API
- JSR 172 — Java Platform, Micro Edition Web Services
- JSR 177 — Security and Trust Services API
- JSR 179 — Location API for J2ME
- JSR 180 — SIP API for J2ME
- JSR 184 — Mobile 3D Graphics API
- JSRs 120 and 205 — Wireless Messaging API
- JSR 211 — Content Handler API
- JSR 226 — Scalable 2D Vector Graphics API for J2ME
- JSR 229 — Payment API
- JSR 234 — Advanced Multimedia Supplements
- JSR 238 — Mobile Internationalization API
- JSR 239 — Java Binding for the OpenGL® ES API

	MSA	MSA Subset	JTWI	JTWI	Custom
MIDP	2.1	2.1	2.0	2.0	2.1
CLDC	1.1	1.1	1.0	1.1	1.1
75	✓	✓	◆	◆	◆
82	✓	✓	◆	◆	◆
135	✓	✓	✓	✓	◆
172	✓	✗	✗	✗	***
177*	◆	◆	◆	◆	◆
179	✓	◆	✗	◆	◆
180	✓	◆	◆	◆	◆
184	✓	✓	✗	◆	◆
205	✓	✓	◆	◆	**
211	✓	◆	◆	◆	◆
226	✓	✓	✗	✗	***
229	✓	◆	✗	◆	◆
234	✓	◆	◆	◆	◆
238	✓	◆	✗	◆	◆
239	◆	◆	✗	◆	◆

✓ Required ◆ Optional ✗ Not Available

* 177 has four options: APDU, Crypto, PKI, and JCRMI. Any combination can be used.

** The custom target allows you to exclude WMA or to select 120 or 205 support. 135 is included by default, but not required.

*** Web Services and JSR 226 require JAXP XML Parser

Figure 1. API selection options

MIDP support

The Java Wireless Toolkit includes tools that support these and other MIDP features:

- An enhanced security architecture, including permission-based security and secure HTTPS networking
- Ability of MIDlets to receive incoming network connections using a push registry
- Over-the-air installation of applications

Security support

MIDP 2.0 applications must have permission to perform network access or other sensitive operations. The toolkit's Permissions dialog makes it easy to select and add permission attributes to a MIDlet suite.

Mobile device applications execute in specific MIDP 2.0 runtime protection domains. The Preferences dialog provides a selection of predefined protection domains that are specific to the chosen platform (for testing purposes only). The toolkit also provides tools for signing MIDlet suites, including a test root certificate accepted by the emulator to verify signed MIDlet suites. You can easily create your own test keys and certificates from the signing tool.

Push registry support

MIDP offers a push registry to launch applications in response to incoming network connections or timers. With the Java Wireless Toolkit's implementation of the push registry, both an inbound network connection or a timer-based alarm can activate a MIDlet. This

gives MIDlets a broad range of new uses, such as responding to newly received email messages or scheduled appointments or performing regularly scheduled tasks.

Content handlers

The Content Handler API provides a way to launch MIDlets in response to incoming content. For example, a device might launch a video player MIDlet to handle incoming video content. The Java Wireless Toolkit makes it easy to map content types to MIDlets, and you can also use the emulator to test the behavior of your content handlers.

Over-the-air application management

Users typically download applications over a wireless network, a process that is commonly referred to as over-the-air (OTA), user-initiated provisioning. The Java Wireless Toolkit includes a small server that simulates a production OTA environment. Using this feature, you can easily test provisioning of your applications without setting up and configuring an OTA server or moving MIDlet suite files. This utility is critical if you want your applications to test MIDP 2.0 features, such as the push registry or signed MIDlets.

Built-in support for optional APIs

In addition to the extensive optional API support built into the emulator, the Java Wireless Toolkit provides utilities or preferences for configuring a variety of APIs:

- **Java APIs for Bluetooth (JSR 82)**

Bluetooth is an important standard for local wireless networking. The specification standardizes a set of Java technology APIs to allow these Java technology-enabled devices to integrate into a Bluetooth environment and includes basic support for the OBEX protocol.

- **J2ME Web Services Specification (JSR 172)**

The Java Wireless Toolkit includes a stub generator to simplify the process of accessing Web services from MIDP applications.

- **Location API (JSR 179)**

The Location API allows applications equipped with a Global Positioning System (GPS) or other hardware to determine where the device is located. You can set the simulated location of the emulator at runtime or run a script that changes the location automatically over time.

- **Session Initiation Protocol (SIP) API for J2ME (JSR 180)**

The (SIP) provides a standard way for applications to communicate. It can be used to manage instant messaging, text chat, voice chat, videoconferencing, and other types of communication. The toolkit simplifies your development because it provides a basic SIP proxy server and registrar for you. SIP messages and responses can be viewed in the network monitor.

- **Wireless Messaging API (WMA) (JSR 120 and JSR 205)**

Sending short text messages with mobile devices is increasingly popular. Using the Wireless Messaging API (WMA), devices have the ability to exchange Short Message Service (SMS) and other messages from MIDP and other Java platforms. The toolkit includes a WMA console, a utility that enables message exchange with the emulator. Messages can be sent between emulators or from the console itself to an emulator, and they can even be broadcast to all running emulators.

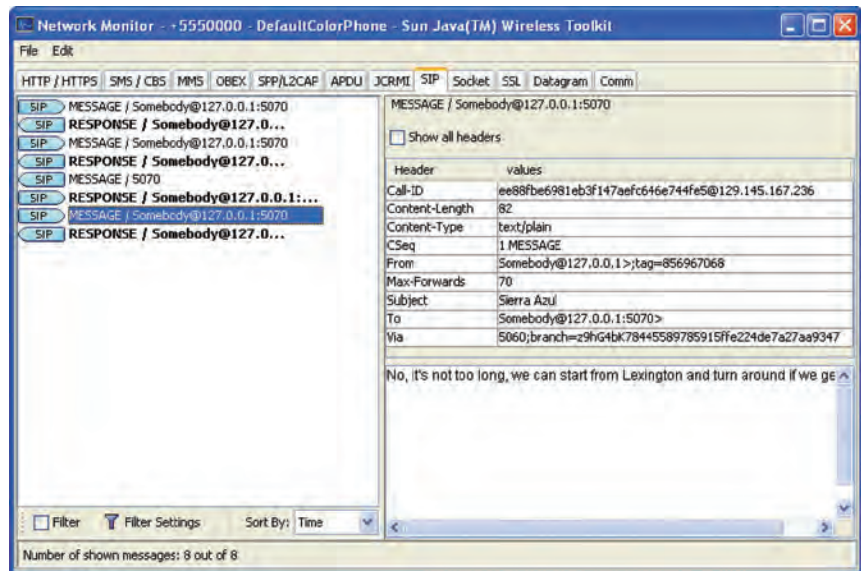


Figure 2. The network monitor tracks SIP messages and responses.

- **Scalable 2D Vector Graphics API for J2ME (JSR 226)**

The API for Scalable Vector Graphics (SVG) supports interactive 2D graphics written in XML. Useful features include animation and the ability to create graphics that adapt to the display resolution and form factor of the current device.

- **Payment API (JSR 229)**

JSR 229, the Payment API, enables applications to make payments on behalf of their users. It supports different payment mechanisms through payment adapters. The emulator implements the Payment API with a sample payment adapter that simulates both Premium Priced SMS (PPSMS) and credit card payments. KToolbar provides forms for setting up payment attributes and preferences. A payment console enables you to easily track payment attempts and record success or failure.

- **Mobile Internationalization API (JSR 238)**

This API provides a way for applications to be displayed in multiple languages and used in multiple countries. Resources are stored in files in a format defined in JSR 238. The resource files are bundled as part of the MIDlet suite JAR file. The toolkit provides a resource manager that simplifies the job of creating and maintaining resource files.

- **Java Binding for the OpenGL® ES API (JSR 239)**

The OpenGL® for Embedded Systems (OpenGL® ES) API is a low-level graphics library suitable for accessing hardware-accelerated 3D graphics. JSR 239 defines the Java programming language bindings for the OpenGL® ES API.

Performance monitoring and tuning

The Java Wireless Toolkit enables you to monitor and performance-tune your applications. The tools available to developers include:

- Method profiler — highlights application bottlenecks by showing the methods an application called, how often they were called, and the time spent in each method
- Memory monitor — shows the memory a MIDlet uses as it runs and the current memory in use for each object.
- Network monitor — shows a MIDlet's network connections, including the values of all HTTP requests and responses between the client and the server. It also monitors various other communication protocols, as shown in Figure 2.

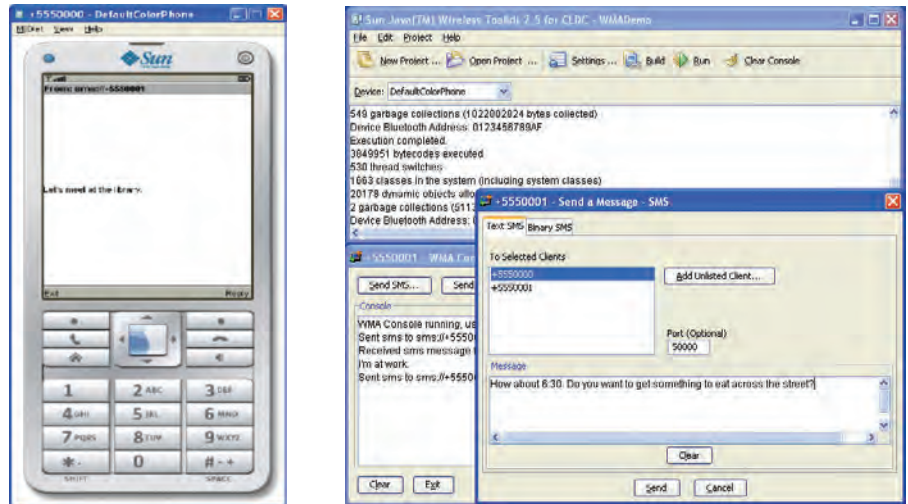


Figure 3. WMA messaging in KToolbar, the WMA console, and the default emulator. The WMA console and the emulator send and receive messages. KToolbar displays project information.

Customization

You can easily create new emulator skins for the toolkit's default emulator, changing the screen size, modifying display colors, repositioning buttons, and more.

Java Wireless Device Custom Toolkit

For further customization, you can purchase a technology license for the source code from Sun. The license allows you to create custom versions of the Java Wireless Toolkit. Or Sun can customize the source code for you to build an emulation environment for your specific devices or networks. You can also use the package to build a custom development environment based on open standards.

About Sun

A singular vision, The Network is the Computer™, drives Sun in delivering industry-leading technologies that focus on the whole system — where computers, software, storage, and services combine. With a proven history of sharing, building communities, and innovation, Sun helps create opportunities, both social and economic, around the world. You can learn more about Sun at sun.com.

Learn more

Get the inside story on the trends and technologies shaping the future of computing by signing up for the Sun Inner Circle Program.

You'll receive a monthly newsletter packed with information on the latest innovations, plus access to a wealth of resources. Register today to join the Sun Inner Circle Program at sun.com/joinic.

For more information on the Java Wireless Device Custom Toolkit, see www.sun.com/software/jpe/es. To learn more about the Java Platform, Micro Edition, visit java.sun.com/javame. To download the Java Wireless Toolkit, visit java.sun.com/products/sjwtoolkit.