Oracle® Java ME Embedded

Reference Platform Release Notes (Qualcomm IoE)

Release 3.4

E47940-01

September 2013

Table of Contents

Introduction

The Oracle Java ME Embedded software for the Qualcomm IoE platform is a ready-to-run software image for use with a Qualcomm IoE embedded board. The Oracle Java ME Embedded software is an optimized platform stack for small embedded devices, which includes the Connected Limited Device Configuration (CLDC) HotSpot Implementation (Java Virtual Machine), Information Module Profile - Next Generation (IMP-NG) application environment, and enhanced support for Java Specification Requests (JSRs).

What's Supported in this Release

The following features are supported in the Oracle Java ME Embedded 3.4 software:

- Support for the Qualcomm IoE embedded platform using a multitasking virtual machine (MVM). The Qualcomm IoE hardware platform is also available in emulation on the Windows desktop. See the *Oracle Java ME SDK 3.4* documentation at the following address for more information:

  `http://docs.oracle.com/javame/developer/developer.html`

- Tooling over USB that employs a serial connection to the Qualcomm IoE embedded platform for logging, command-line inteface, and debugging.
- Support for the Application Management System (AMS) API, the Logging API, and the AccessPoint API.

ORACLE

- Support for Device Access APIs, Version B, which provide enhanced device controls and improved input/output (IO) for small embedded devices, including the following new features:
  - Peripheral Manager
  - GPIO pins and ports
  - I2C
  - SPI
  - Pulse Counter
  - Watchdog Timer
  - Analog-to-Digital Converter (DAC)
  - Digital-to-Analog Converter (ADC)
  - Universal Asynchronous Receiver/Transmitter (UART)
  - Modem Control (AT) command set
- Ongoing support for the following optional packages:
  - JSR 139 - CLDC 1.1
  - JSR 228 - IMP-NG
  - JSR 75 - (FileConnection API only)
  - JSR 172 - Web Services
  - JSR 177 - Security and Trust Services API (SATSA-CRYPTO package only)
  - JSR 280 - XML API for Java ME
  - JSR 120 - WMA
  - JSR 179 - Location

Usage Notes

The Oracle Java ME Embedded software for the Qualcomm IoE embedded platform consists of a binary files and a directory structure that needs to be downloaded to a Qualcomm IoE board running the Brew MP operating system.

The *Getting Started Guide for the Reference Platform (Qualcomm IoE)* describes how to download the binary file to the board, how to connect to the board from the development host computer and how to install, run, and debug IMlets on the board.

Observe the following important notes before running the Oracle Java ME Embedded software on the board:

- During any modification of the MAX_ISOLATES property in the `jwc_properties.ini` file, keep in mind that up to 3 isolates can be reserved by the Java runtime.
- The CPU and Memory Profiler are not supported in this release.
- Memory Monitoring is not supported on the board in this release. However, the Memory Status feature is supported and can be used to get basic memory status information.
- Only the default Access Point is supported in this release.

- Oracle recommends using the serial mode for tooling, especially for debugging, as it is faster and more reliable than the network mode, which works over WiFi or 3G.
- Be aware that a network can be lost and appear again at any time (such as WiFi and 3G specific networks), IP address can be changed during time due to routers, DHCP, and networks specifics. IMlets should be written correspondingly to react to these situations.
- Some mobile-network related functionality (for example, WMA or AT-commands) depend on a particular network carrier and may not work in certain environments. For example, depending on the carrier, it may be not possible to send large SMS (about 400 characters) messages every 100 milliseconds.

Installation and Runtime Security Guidelines

The Oracle Java ME Embedded Release 3.4 software installation requires an execution model that makes certain networked resources available for device emulator execution. These required resources might include, but are not limited to, a variety of communication capabilities between the product's installed components.

It is extremely important to note that the product's installation and runtime system is fundamentally a developer system not specifically designed to guard against any malicious attacks from outside intruders. Given this, the product's architecture can present an insecure operating environment to the installation file system itself, as well as its runtime environment, during execution. For this reason, it is critically important to observe the precautions outlined in the following security guidelines when installing and running the software.

> **Note:** The security-related functionality of a final developed application for release into the field is supported by the available components of the Oracle Java ME Embedded software stack incorporated by the developer into the application. The security precautions required by applications in the field are beyond the scope of these recommendations, but must nonetheless be observed by the application developer.

To maintain optimum network security, the software package can be installed and run in a “closed” network operating environment, meaning the software system is not connected directly to the Internet, or to a company Intranet environment that could introduce unwanted exposure to malicious intrusion. This is the ideal secure operating environment whenever the application under development does not require it.

An example of a requirement for an Internet connection is when the system must communicate with a wireless network over the Internet to fully execute the application under development. Whether or not an Internet connection is required depends on the particular Java ME application running in the development environment. For example, some Java ME applications can use an HTTP connection. If the environment is open to any network access you must always observe the following precautions to protect valuable resources from malicious intrusion:

- Locate the development environment behind a secure firewall that strictly limits unauthorized network access to its file system and services. Limit access privileges to those that are required for development while allowing all the bi-directional local network communications that are necessary for the application's

functionality. The firewall configuration must support these requirements to run the software while also addressing them from a security standpoint.

- Follow the principle of “least privilege” by assigning the minimum set of system access permissions required for installation and execution of the software.
- Do not store any data sensitive information on the same file system that is hosting the installation.
- To maintain the maximum level of security, make sure the operating system patches are up-to-date on any host machines in the development environment.

Security Certificate Precautions

The Oracle Java ME Embedded software distribution bundle contains security certificates that are needed for testing purposes during development of products for final release to customers. Some of these certificates are self-signed security certificates generated by Oracle that are mapped to privileged security domains. IMlets signed by these certificates get high privileges to access restricted APIs, and so these certificates present a security vulnerability if they are released to end users on a customer's device. Care should be taken to remove these certificates after final testing of the product is completed when the product is being prepared for release to end users. This does not apply to certificates issued by universally recognized certificate authorities (CAs), because these are used only for signature verification and do not present a vulnerability.

Command-Line Interface Precautions

The command-line interface (CLI) feature in this Oracle Java ME Embedded software release is provided only as a concept for your reference. It uses insecure connections with no encryption, authentication, or authorization. If you decide to implement this feature in any product deployment, it is your responsibility to incorporate adequate security measures around the CLI.

Known Bugs

Table 1, " Known Bugs", shows the known bugs in the 3.4 release of the Oracle Java ME Embedded software.

Table 1 Known Bugs

Bug Number	Bug Description
MERT-2257	**FileConnection.lastModified returns a value that depends on TimeZone setting** `FileConnection.lastModified` provides a long value representing the time the file was last modified, measured in milliseconds since the epoch (00:00:00 GMT, January 1, 1970), or 0L if an I/O error occurs. However, the value returned depends on `TimeZone` setting.

Table 1 (Cont.) Known Bugs

Bug Number	Bug Description
MERT-2425	**Public AMS API: running a task with the wrong suite name doesn't cause an error.** Running a task using the AMS API with the wrong suite name doesn't throw an exception, nor does it return `null` for the `TaskInfo` instance if the task can't start. **Recommendation**: Don't rely on an exception or the returned value from actually running the task; instead, use the infomation from the `TaskInfo` instance received from `TaskListener.notifyStatusUpdate()` and `TaskListener.notifyStatusStopped()`.
MERT-2935	**Debugging: Expressions tab does not show fields.** In Eclipse, using static variables in the Expressions tab in the Debug perspective might cause the debug session to freeze with the following symptoms: Only "pending" values instead of expression results; multiple errors in the device console. **Solution:** Avoid the use of static variables in expressions; if necessary, monitor these values in the Variables tab.
MERT-3041	**The JMEE VM is not fully compatible with JDWP.** Only a JDWP subset is supported. This subset is enough to work with the current versions of Eclipse/NetBeans.
MERT-3291	**PeripheralNotFoundException when opening newly registered UART** The return value of the `register()` function (the new ID) cannot be used as a parameter to the `PeripheralManager.open()` function, or a `PeripheralNotFoundException` will be thrown. **Solution**: You can create your own `UARTConfig` object and pass it into the `PeripheralManager.open()` function without error.
MERT-3437	**There are a small number of unsupported JDWP features.** The most noteworthy missing feaures are: 1) Missing method entry/exit breakpoint support, although it is visible for NetBeans users 2) Missing ClassObject req support; Eclipse breaks on it when showing static variables.
MERT-3577	**PeripheralTypeNotSupportedException** is thrown during call to `PeripheralManager.open(config)` The Oracle Java ME Embedded platform may throw a unexpected exception when using the `PeripheralManager.open(config)` method with an invalid configuration.
MERT-3638	**Device Access API: isOpen() returns different values** Please note that `Peripheral` instances returned by `PeripheralManager.list()` are in a closed state, and calling `isOpen()` for them always returns 'false' even if such `Peripheral` was already opened by other means.
MERT-3651	**SPI: Can open devices using negative addresses** According to the specification, an `IllegalArgumentException` should be thrown if address is not in the defined range (i.e. not a positive or null integer). However, the implementation now contains workarounds to support additional devices that have negative addresses.

Table 1 (Cont.) Known Bugs

Bug Number	Bug Description
MERT-3676	**Infinite log output when JC_SOCKET=0** When the TCP logger is used and the `JC_SOCKET` parameter is set to 0 in the `jwc_properties.ini` file, there can be infinite output from the logger. **Workaround**: Never set `JC_SOCKET` to 0. The best solution is manually set `JC_SOCKET` to 4 in the `jwc_properties.ini` file.
MERT-3798	**Device Access API Spec: Description on unregistering an application-registered peripheral ID while this particular peripheral is still in an open state** The Device Access API Spec does not outline what behavior should occur when unregistering an application-registered peripheral ID while this peripheral is still in the open state. The curent behavior is to close the peripheral.
MERT-3799	**ADCChannel.startAcquisition behavior with provided count = 0 is not specified in DA API specification.** The DA API specification does not outline the behavior when the `ADCChannel.startAcquisition(values, offset, count, doubleBuffering, listener)` method is called and `count` is zero. Currently, an IllegalArgumentException is thrown. **Workaround**: Do not set count = 0 while starting acquisition.
MERT-3800	**DACChannel.startGeneration behavior with provided count = 0 is not specified in DA API specification.**The DA API specification does not outline the behavior when the `DACChannel.startGeneration(values, offset, count, doubleBuffering, listener)` method is called and `count` is zero. Currently, an IllegalArgumentException is thrown. **Workaround**: Do not set count = 0 while start generation.
MERT-3806	**AcquisitionEvent and Monitoring constructors do not throw any exceptions** The `AcquisitionEvent` and `MonitoringEvent` constructors do not throw any exceptions (except `NullPointerException`) when invalid parameters are supplied.
MERT-3808	**PinEvent and PortEvent constructors do not throw any exceptions** The `PinEvent` and `PortEvent` constructors do not throw exceptions when their parameters are out of the defined range.
MERT-3887	**An unexpected PeripheralConfigInvalidException is thrown, instead of PeripheralNotFoundException, when opening a peripheral using incorrect configuration parameters.** Instead of a `PeripheralNotFoundException`, an unexpected `PeripheralConfigInvalidException` is thrown when calling `PeripheralManager.open(java.lang.Class intf, PeripheralConfig config)` when the `config` parameter contains incorrect peripheral hardware addressing. **Workaround**: An application should catch both exceptions when opening a peripheral.

Table 1 (Cont.) Known Bugs

Bug Number	Bug Description
MERT-3932	**UART port fails to open with a valid or invalid configuration right after it was opened with an unsupported dataBits configuration.** Right after a failed attempt to open UART port with an invalid `dataBits` value, and a `PeripheralConfigInvalidException` is thrown, any further attempt to open the same UART port with a valid or invalid configuration will throw a `PeripheralNotAvailableException`. **Workaround**: 1. Use only valid configuration, that is supported by underlying platform (refer to the Getting Started Guide for supported configuration information); 2. If a `PeripheralNotAvailableException` is thrown, wait some time or reboot the board and open UART port with valid configuration again.
MERT-4306	**GPIO output voltage problem.** As was observed on some instances of IoE boards, an output voltage of a GPIO pin may be 1.79 V, not 1.8 V.
MERT-4351	**External I2C slave devices with pull-up resistors do not work.** An external I2C slave device that has its own pull-up resistors on both the SDA and SCL lines doesn't work with the board. If the external device does not have pull-up resistors, there are no problems. **Workaround**: Remove the R16/R17 (SCL/SDA pulls) resistors from the main IoE board (Gobi) to disable the pull up, or do the same on I2C slave's side.
MERT-4357	**Impossible to open one more SPI slave device with a different configuration without rebooting the board.** Only one SPI slave device can be opened at a time. When the first SPI device is closed, any attempt to open the same or another SPI device with a different configuration will fail. A new SPI configuration is applied only after a board reboot. **Workaround**: Before working with a new SPI slave device, reboot the board.
MERT-4384	**Breakpoint condition "Multiple of" does not work with remote JMEE VM.** Debugging using NetBeans of the code with several threads is unpredictable. This is an issue with NetBeans, and is documented at `https://netbeans.org/bugzilla/show_bug.cgi?id=227746`.
MERT-4386	**An unnecessary pin (pin 36) is mentioned in the jwc_properties.ini file for a pre-configured GPIO port (id=200, name="LEDS")** The port "LEDS" is configured to include three pins; this configuration is described in GSG. However, the list of pins in the `jwc_properties.ini` file for this port includes four pins. The last pin in the list (pinNumber=36) is ignored by the implementation since the `pinCount` for the port is 3. The implementation behaves according to the documentation (only three pins are opened when the port is opened; the fourth pin is ignored). The confusing fourth pin may be deleted from the `jwc_properties.ini` file.
MERT-4569	**Garbage appears in the serial channel which leads to channel closing** In rare cases a PC can lost a connection with the board when the tooling over serial is used for running, debugging, or profiling applications. This is caused by "garbage" that appears in the serial communication channel. **Workaround**: Reboot the board

Table 1 (Cont.) Known Bugs

Bug Number	Bug Description
MERT-4570	**ATDevice.sendCommand method sometimes throws an IOException when a supported command is being sent** In some cases, the `ATDevice.sendCommand()` function can throw an `IOException` at the time that a supported AT command is being sent. **Workaround**: Try to send the command again if it is in the list of supported commands.
MESDK-1939	**Occasionally during the debugging or profiling of an IMlet, the Device Manager looses a connection to the board.** Occasionally during the debugging or profiling of an IMlet, the Device Manager looses a connection to the board. The situation is reproduced only if an IMlet tries to open many network related connections to URLs that are not available/reachable. **Workaround:** If tooling's network mode is leveraged then change JC_SOCKET property's value to 4 (DISABLE), it will decrease the number of faults significantly. Otherwise, use tooling's serial mode for debugging/profiling. This mode is more preferable then the network mode. (Refer to the Getting Started Guide in order to figure out what is meant under a tooling mode and how to change system properties.)

Product Documentation

The following documentation is included with this release of the Oracle Java ME Embedded software. See `http://docs.oracle.com/javame/embedded/embedded.html`.

Application	Title	Format
All (This document)	*Release Notes*	HTML
Introduction to running Oracle Java ME Embedded on the Qualcomm IoE reference platform.	*Getting Started Guide for the Reference Platform (Qualcomm IoE)*	HTML PDF
Application Management System classes (unchanged from version 3.3)	*AMS API Javadocs*	HTML
Device Access API classes (unchanged from version 3.3)	*Device Access API Javadocs*	HTML
Access Point API classes (unchanged from version 3.3)	*Access Point API Javadocs*	HTML
Logging API classes (unchanged from version 3.3)	*Logging API Javadocs*	HTML

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at:
`http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or

visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Oracle Java ME Embedded Reference Platform Release Notes (Qualcomm IoE), Release 3.4
E47940-01