Oracle® Java ME Embedded
Getting Started Guide for the Reference Platform (Qualcomm IoE)
Release 3.4
E47914-01

September 2013
This book describes how to install and run the Oracle Java ME Embedded software on the Qualcomm IoE reference platform.
## Contents

**Preface** ......................................................................................................................................................... ix  
  Audience ....................................................................................................................................................... ix  
  Documentation Accessibility ......................................................................................................................... ix  
  Related Documents ....................................................................................................................................... ix  
  Operating System Commands ....................................................................................................................... ix  
  Shell Prompts ................................................................................................................................................ x  
  Conventions .................................................................................................................................................. x  

1 **Installing Oracle Java ME Embedded on the Qualcomm IoE Board**

   Setting Up the Qualcomm IoE Board ................................................................................................................ 1-1  
   Downloading and Installing the Brew MP SDK Tools .................................................................................. 1-2  
   Installing the Qualcomm IoE USB Drivers .................................................................................................. 1-2  
   Copying Files to the Qualcomm IoE Board ................................................................................................. 1-3  

2 **Tooling Over Serial and Networking**

   Tooling Overview ........................................................................................................................................... 2-1  
   Using Serial Mode ........................................................................................................................................ 2-1  
   Installing and Configuring the Java ME SDK 3.4 ....................................................................................... 2-2  
   Downloading and Installing the PuTTY Terminal Emulator Program ...................................................... 2-3  
   Using the Java Logging Interface .............................................................................................................. 2-3  
   Using the Command-Line Interface ........................................................................................................... 2-4  
   AMS and System Commands ....................................................................................................................... 2-6  
   Networking Mode ....................................................................................................................................... 2-8  
     Configuring Wi-Fi networking .................................................................................................................. 2-9  
     IP Address Periodic Logging .................................................................................................................. 2-9  
     TCP-to-Serial Fallback ........................................................................................................................... 2-10  
     Automatic Recovering for Java Logger and CLI Connections ............................................................... 2-10  

3 **Installing and Running Applications on the Qualcomm IoE Board**

   Using NetBeans with the Qualcomm IoE Board .............................................................................................. 3-1  
   Installing the Oracle Java ME SDK 3.4 Plugin for NetBeans .................................................................. 3-1  
   Adding the Qualcomm IoE Board to the Device Selector ........................................................................ 3-2  
   Assigning the Qualcomm IoE Board to Your Project .............................................................................. 3-3  
   Sample Source Code ................................................................................................................................ 3-4  
   Debugging an IMlet on the Qualcomm IoE Board ...................................................................................... 3-5
List of Examples

3–1 Sample Code to Access a GPIO Pin................................................................. 3-4
List of Figures

1–1    Device Manager with Qualcomm IoE USB Device Drivers Loaded .................................. 1-3
1–2    Brew SDK Loader Connection Manager Dialog .............................................................. 1-4
1–3    The Brew SDK Logger Application .................................................................................. 1-5
2–1    Oracle Java ME SDK Device Manager .............................................................................. 2-2
2–2    PuTTY Configuration for Java Logger Connection .......................................................... 2-3
2–3    Oracle Java ME Embedded Logger .................................................................................... 2-4
2–4    PuTTY Configuration for CLI Connection .......................................................................... 2-5
2–5    Oracle Java ME Embedded Command Line Interface ..................................................... 2-5
3–1    NetBeans Device Selector "Add a Device" Button ............................................................. 3-2
3–2    NetBeans Embedded Device Detection ............................................................................. 3-3
3–3    NetBeans Platform Properties Dialog ............................................................................. 3-4
3–4    Debugging an IMlet on the Qualcomm IoE Board Using NetBeans ................................. 3-6
3–5    Debugging an IMlet on the Board Using the Eclipse IDE .................................................. 3-9
3–6    Adding API Permissions with NetBeans .......................................................................... 3-11
3–7    The Signing Pane in the NetBeans Project Properties .................................................... 3-12
## List of Tables

<table>
<thead>
<tr>
<th>Page</th>
<th>Title</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>2–1</td>
<td>AMS CLI Commands</td>
<td>2-6</td>
</tr>
<tr>
<td>2–2</td>
<td>Additional System Commands</td>
<td>2-6</td>
</tr>
<tr>
<td>2–3</td>
<td>Additional System Commands Available when System Menu is Activated</td>
<td>2-7</td>
</tr>
<tr>
<td>4–1</td>
<td>Problems and Solutions - Starting Oracle Java ME Embedded on the Board</td>
<td>4-1</td>
</tr>
<tr>
<td>4–2</td>
<td>Problems and Solutions - Oracle Java ME SDK and the NetBeans IDE</td>
<td>4-3</td>
</tr>
<tr>
<td>C–1</td>
<td>Installer Error Codes</td>
<td>C-1</td>
</tr>
</tbody>
</table>
Preface

This book describes how to install and configure Oracle Java ME Embedded software onto a Qualcomm Internet-of-Everything (IoE) embedded device. In addition, it contains troubleshooting information and Device Access API peripheral specifications useful for Java embedded developers.

Audience

This document is intended for developers who want to run Oracle Java ME Embedded software on a Qualcomm IoE device.

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.

Related Documents

This document frequently references the Qualcomm IoE Development Platform User’s Guide, which can be downloaded at:


For a complete list of documents with the Oracle Java ME Embedded software, see the Release Notes.

Operating System Commands

This document does not contain information on basic commands and procedures such as opening a terminal window, changing directories, and setting environment
variables. See the software documentation that you received with your system for this information.

**Shell Prompts**

<table>
<thead>
<tr>
<th>Shell</th>
<th>Prompt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bourne shell and Korn shell</td>
<td>$</td>
</tr>
<tr>
<td>Windows</td>
<td>directory&gt;</td>
</tr>
</tbody>
</table>

**Conventions**

The following text conventions are used in this document:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface</strong></td>
<td>Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.</td>
</tr>
<tr>
<td>italic</td>
<td>Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.</td>
</tr>
<tr>
<td>monospace</td>
<td>Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.</td>
</tr>
</tbody>
</table>
This chapter demonstrates how to install the Oracle Java ME Embedded software on the Qualcomm Orion Internet-of-Everything (IoE) board. The following items are required for installing and developing on the Qualcomm IoE board:

- The Qualcomm IoE embedded board running the Brew MP operating system
- A desktop PC running Microsoft Windows 7
- A USB cable with a micro-B connector that can link the Qualcomm IoE board to your desktop PC
- The Qualcomm IoE USB Drivers
- The Qualcomm Netsetup Brew Application
- The Qualcomm Brew MP SDK tools
- A terminal emulation program, such as PuTTY.
- Oracle Java ME Embedded 3.4
- Oracle Java ME SDK 3.4
- The NetBeans 7.3.1+ or Eclipse Indigo or Juno IDE (optional)

Setting Up the Qualcomm IoE Board

First, download the Qualcomm IoE Development Platform User’s Guide in PDF format, which can be downloaded at:


This document contains important information about the Qualcomm IoE board and its hardware.

Next, perform the following steps, in order.

1. Assemble and connect the board’s components as listed in Chapter 3, "Hardware", and Chapter 5, "Hardware Configuration", of the Qualcomm IoE Development Platform User’s Guide.

2. Connect the power supply as shown in the Qualcomm IoE Development Platform User’s Guide and power on the board.
   - To use an AC-power source, see Section 5.3.5.5, "AC-powered operation."
■ To use a battery source, see Section 5.3.5.4, "Battery-powered operation."

**Downloading and Installing the Brew MP SDK Tools**

Download the Brew MP SDK tools from the following site:

https://developer.brewmp.com/tools/brew-mp-sdk

The version used in this documentation is 7.12.5. Double-click on the installer executable, and install the application on your Windows platform desktop as per the instructions shown. You will need the **Loader** and **Logger** applications that are installed with the Brew MP SDK Tools in later sections.

**Installing the Qualcomm IoE USB Drivers**

To develop on the Qualcomm IoE board, you must first install the Windows USB drivers for the board. If you have not already done this, use the following steps:

1. Download the Qualcomm IoE USB drivers from the following site:


2. Ensure that the Qualcomm IoE board is powered up by pressing the "PWR KEY" button on the board. Then, follow the instructions in Chapter 2, "Software Setup," of the *Qualcomm IoE Development Platform User’s Guide* to properly install the USB drivers in Windows.

3. Open the Windows Device Manager (**Start** -> Search for ‘**Device Manager’**).

4. Ensure that the drivers are successfully installed by verifying the following hardware ports, as shown in **Figure 1–1**.

   ■ HS-USB Modem (AT command port)
   ■ HS-USB Serial Port (Java tooling port)
   ■ HS-USB Diagnostics Port
   ■ HS-USB NMEA Port (GPS)
Copying Files to the Qualcomm IoE Board

Finally, you must install the Java ME Embedded Software on the Qualcomm IoE board using the **Loader** and **Logger** applications. Follow these steps to copy the appropriate files to the board:

1. Download and uncompress the Oracle Java ME Embedded Software 3.4 for the Qualcomm IoE board.
2. Obtain the latest `java.sig` and `netsetup.sig` signature files from the Qualcomm IoE website:
   
   ![Device Manager with Qualcomm IoE USB Device Drivers Loaded](image)

   ![Device Manager with Qualcomm IoE USB Device Drivers Loaded](image)

   

3. Start the **Loader** application that was installed with the Brew SDK. Click Start, and then type **Loader** in the Search box. Right-click **Loader** under Programs.
4. When the **Loader** application starts, connect to the board using the Connection type: Brew Devices (COM/DIAG), and whichever port matches the Qualcomm HS-USB Diagnostics Port, as shown in Figure 1–2.
5. If you are upgrading from the Oracle Java ME Embedded 3.2 release, you will need to delete all the files in the /sys/mod/java and /sys/mod/netsetup directories, including the .sig files. After deleting the files, reboot the board.

6. If it doesn’t already exist, create the /sys/mod/java directory and drag and drop following files from the Oracle Java ME Embedded distribution’s java folder into it.
   - appdb (folder)
   - java.mif
   - java.mod
   - jwc_properties.ini
   - watchdog.ini

7. Copy the java.sig signature file obtained from Qualcomm to the /sys/mod/java directory.

8. If it doesn’t already exist, create the /sys/mod/netsetup directory and drag and drop following files from the Oracle Java ME Embedded distribution’s netsetup folder into it.
   - netsetup.mif
   - netsetup.mod

9. Copy the netsetup.sig signature file obtained from Qualcomm to the /sys/mod/netsetup directory.

10. Reset the board by pressing the “RESET KEY” on the board, then wait approximately 40 seconds for the Java VM to startup.

11. Start the Logger application in the same way you started the Loader application. Connect to the board using the Connection type: Brew Devices (COM/DIAG), and whichever port matches the Qualcomm HS-USB Diagnostics Port. Connect to the board, press the Start Logging button, and verify that the Java VM is sending logging information to the Logger application by checking for messages that come from the [JVMStdout] file name. See Figure 1–3.
12. Once Java is successfully running on the Qualcomm IoE board, continue on to the next chapter to learn how to use the tooling features of the Oracle Java ME Embedded software on the board. If Java is not running, please see Chapter 4, "Troubleshooting" to diagnose possible problems.
This chapter demonstrates how to configure the Qualcomm IoE embedded board using the serial and networking connections of the Oracle Java ME Embedded software so that external communication is possible: a process known as tooling. However, in order to successfully communicate with the Qualcomm Orion Internet-of-Everything (IoE) board through a serial connection, the Oracle Java ME SDK 3.4 distribution must be installed, and the Device Manager must be configured to recognize the board.

---

**Note:** The term IMlet, in the context of the Oracle Java ME Embedded command-line interface and references in this chapter, is synonymous with MIDlet.

---

**Tooling Overview**

The Oracle Java ME Embedded platform offers the following tools that can be used to communicate and configure the Qualcomm IoE embedded board:

- A command-line interface (CLI) via a terminal emulator program for Application Management System (AMS) commands and for system configuration commands.
- A logging interface for obtaining JVM diagnostic information using a terminal emulator program.
- On-Device Tooling (ODT): the ability to install, run, and debug applications from an IDE on the desktop such as NetBeans or Eclipse.

It is important to note that tooling works over one physical channel. However, the CLI, logging, and ODT functions all use different ports. The ports for CLI and logging are available to the user, and will be discussed later in the chapter. However, the ports used for ODT are invisible to the user, and used by external development tools.

There are two tooling modes available: **serial** and **network**. The default tooling mode for the Oracle Java ME Embedded 3.4 release is **serial**. It's important to note that setting a particular tooling mode alters all the tooling's components. All the tooling components always work in the same mode. In other words, if the tooling mode is set to serial, all connections to the board will be made through that serial connection.

---

**Using Serial Mode**

In order to use the serial mode for tooling with the Qualcomm IoE embedded board, perform the following steps:

1. Connect the Qualcomm IoE board to the PC using a USB cable.
2. Ensure that the com.oracle.tooling.mode property in the jwc_properties.ini file on the board is set to serial, and that the odt_run_on_start property is set to true. See Appendix B, "Configuring the Java Runtime Properties" for more information.

3. Start the Oracle Java ME SDK 3.4 Device Manager on the desktop, if it is not already started, and ensure that the device is connected, as shown in the following section.

Installing and Configuring the Java ME SDK 3.4

Download and install the Java ME SDK 3.4 distribution onto your Windows desktop platform from the Oracle Technology Network website.


Once this is installed, start the Oracle Java ME SDK Device Manager (located at <SDK Installation Folder>/bin/device-manager.exe) and right click on its icon in the taskbar, then select Manage Device Addresses. Selecting this option will bring up a small dialog box that looks similar to Figure 2–1.

Figure 2–1 Oracle Java ME SDK Device Manager

Choose the COM port that corresponds to the Qualcomm IoE HT-USB Serial Port, as reported earlier by the Windows Device Manager. At this point, the Oracle Java ME SDK Device Manager should locate the device and report that an external IMP-NG device has now become available.

Note that you can freely reboot the board or restart Java on the board without rebooting the Oracle Java ME SDK Device Manager. However, if you reboot the Device
Using the Java Logging Interface

Manager, you must reboot Java or the board as well. If you have issues with the Device Manager connecting to the board, please see Chapter 4, “Troubleshooting”.

Downloading and Installing the PuTTY Terminal Emulator Program

Next, download the PuTTY Terminal Emulator Program (putty.exe) from the following site:

http://www.putty.org/

The terminal emulator executable is directly downloadable as putty.exe. The terminal emulator is used to connect to both the Java logger and the command-line interface (CLI) that issues commands to the board.

WARNING: Using the PuTTY Terminal Emulator Program is highly recommended. You’re free to use any terminal program to connect to the Java Logger or CLI. However, Oracle cannot guarantee that other terminal programs will work with the Java Logger and CLI in the same manner as PuTTY.

Using the Java Logging Interface

To connect to the Java logger, start a PuTTY executable on your desktop computer. Use this to create a Raw network connection to the board’s IP address (127.0.0.1 in the case of serial mode) with the port 65000, as shown in Figure 2–2.

Figure 2–2 PuTTY Configuration for Java Logger Connection

Once connected, you should start seeing output from the Java Logger, as shown in Figure 2–3.
Connecting to port 65000 will display the logging information from only the Oracle Java ME Embedded platform. However, you can use the Brew MP SDK Logger application, as shown in Chapter 1, to capture logging output from both the Oracle Java ME Embedded system as well as the Qualcomm IoE board itself.

Finally, if you’re using an IDE such as NetBeans or Eclipse, application output sent to `System.out` or `System.err` will appear in the Java output window of the IDE, as shown in Chapter 3, "Installing and Running Applications on the Qualcomm IoE Board".

**Note:** The NetBeans and Eclipse IDEs also connect to port 65000 of the device to capture Java logs and display them in the appropriate logging windows. If you cannot connect to port 65000 with PuTTY, check that NetBeans or Eclipse is not already using this port. The same is true if you don’t see any logs in NetBeans or Eclipse IDEs: ensure that there are no active connections to port 65000 from PuTTY.

**Using the Command-Line Interface**

The command-line interface is used to issue commands directly to Java runtime. To use the command-line interface, start a PuTTY executable on your desktop computer. Use this to create a Raw network connection to the board’s IP address (127.0.0.1 in the case of serial mode) with the port 65002, as shown in Figure 2–4.
If you are using the serial mode to connect to the board, ensure that the Java ME SDK 3.4 Device Manager has already successfully detected the device, as shown earlier in Figure 2–1.

The window from the connection provides a command-line interface (CLI), and is shown in Figure 2–5:

![PuTTY Configuration for CLI Connection](image)

![Oracle Java ME Embedded Command Line Interface](image)
AMS and System Commands

You can use the command-line interface to run numerous AMS and system commands, as shown in Table 2–1 and Table 2–2.

**Table 2–1 AMS CLI Commands**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ams-list [INDEX or NAME</td>
<td>VENDOR]</td>
</tr>
<tr>
<td>ams-install &lt;URL&gt; [username:password]</td>
<td>Install an IMlet using the specified JAR or JAD file, specified as a URL. An optional username and password can be supplied for login information as well.</td>
</tr>
<tr>
<td>ams-update &lt;INDEX or NAME</td>
<td>VENDOR&gt;</td>
</tr>
<tr>
<td>ams-remove &lt;INDEX or NAME</td>
<td>VENDOR&gt;</td>
</tr>
<tr>
<td>ams-run &lt;INDEX or NAME</td>
<td>VENDOR&gt; [IMLET_ID] [-debug]</td>
</tr>
<tr>
<td>ams-stop &lt;INDEX or NAME</td>
<td>VENDOR&gt; [IMLET_ID]</td>
</tr>
<tr>
<td>ams-suspend &lt;INDEX or NAME</td>
<td>VENDOR&gt; [IMLET_ID]</td>
</tr>
<tr>
<td>ams-resume &lt;INDEX or NAME</td>
<td>VENDOR&gt; [IMLET_ID]</td>
</tr>
<tr>
<td>ams-setup &lt;INDEX or NAME</td>
<td>VENDER&gt;</td>
</tr>
<tr>
<td>ams-info &lt;INDEX or NAME</td>
<td>VENDOR&gt;</td>
</tr>
<tr>
<td>ams-logger-list [INDEX or NAME</td>
<td>VENDOR]</td>
</tr>
<tr>
<td>ams-logger-info &lt;INDEX or NAME</td>
<td>VENDOR&gt; [LOGGER_NAME]</td>
</tr>
<tr>
<td>ams-logger-level-set &lt;INDEX or NAME</td>
<td>VENDOR&gt; [LOGGER_NAME] &lt;LOGGER_LEVEL&gt;</td>
</tr>
</tbody>
</table>

**Table 2–2 Additional System Commands**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>help [command name]</td>
<td>List the available commands or detailed usages for a single command</td>
</tr>
<tr>
<td>sysmenu &lt;on</td>
<td>off&gt;</td>
</tr>
</tbody>
</table>
When the `sysmenu` command is entered with the `on` option, additional system menu commands are available with the CLI, as shown in Table 2–3.

### Table 2–3  Additional System Commands Available when System Menu is Activated

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>setprop &lt;KEY&gt; &lt;VALUE&gt;</code></td>
<td>Sets a property identified by <code>&lt;KEY&gt;</code> with the value <code>&lt;VALUE&gt;</code></td>
</tr>
<tr>
<td><code>getprop &lt;KEY&gt;</code></td>
<td>Returns a property identified by <code>&lt;KEY&gt;</code></td>
</tr>
<tr>
<td><code>saveprops</code></td>
<td>Saves properties to an internal storage</td>
</tr>
<tr>
<td><code>net info</code></td>
<td>Shows the network information of the system</td>
</tr>
<tr>
<td><code>net set ssid &lt;SSID&gt;</code></td>
<td>Sets the SSID value for WIFI access</td>
</tr>
<tr>
<td><code>net set passwd &lt;PASSWD&gt;</code></td>
<td>Sets the password for WIFI access</td>
</tr>
<tr>
<td>`net set pref &lt;0</td>
<td>1</td>
</tr>
<tr>
<td><code>net set apn &lt;APN&gt;</code></td>
<td>Sets APN</td>
</tr>
<tr>
<td>`net set pdp_authtype &lt;0</td>
<td>1</td>
</tr>
<tr>
<td><code>net set pdp_username &lt;USERNAME&gt;</code></td>
<td>Resets the PDP username</td>
</tr>
<tr>
<td><code>net set pdp_password &lt;PASSWORD&gt;</code></td>
<td>Resets the PDP password</td>
</tr>
<tr>
<td><code>net reconnect</code></td>
<td>Reconnects the network and reboots Java</td>
</tr>
<tr>
<td>`odd [on</td>
<td>off]`</td>
</tr>
<tr>
<td><code>shutdown [-r]</code></td>
<td>Perform either a shutdown of the board, or a reboot if the <code>-r</code> parameter has been passed.</td>
</tr>
</tbody>
</table>

Here is a typical example of using the Application Management System (AMS) to install, list, run, and remove a Java ME Embedded application on the board. Note that `/Shared` is a root directory name that can be accessed from Java. Files can be placed in this directory with the help of Qualcomm's `Loader` tool. However, in the `Loader` tool, this directory is named 'shared'.

```
oracle>> ams-install file:///Shared/hello.jar
    <<ams-install,start install,file:///Shared/hello.jar
    <<ams-install,install status: stage 0, 5%
    <<ams-install,install status: stage 3, 100%
    <<ams-install,install status: stage 4, 100%
    <<ams-install,OK,Install success

oracle>> ams-install http://www.example.com/netdemo.jar
    <<ams-install,start install,http://www.example.com/netdemo.jar
    <<ams-install,install status: stage 0, 5%
    <<ams-install,install status: stage 3, 100%
    <<ams-install,install status: stage 4, 100%
```
Note that the final installation example failed with an error code and matching description. If the install process shows any error code, see Table C-1 in Appendix C, "AMS Installer Error Codes" for more information on how to resolve the error.

Once an IMlet is installed, verify it using the ams-list command. Here, we have added an additional IMlet: rs232dem. Each IMlet has been assigned a number by the AMS for convenience.

```
oracle>> ams-list
```
```
<<ams-list,0.hello|Oracle,STOPPED
<<ams-list,1.netdemo|Oracle,STOPPED
<<ams-list,2.rs232dem|Oracle,RUNNING
<<ams-list,OK,3 suites are installed
```

The ams-remove command can be used to remove any installed IMlet.

```
oracle>> ams-remove 0
<<ams-remove,OK,hello removed
```

The results can again be verified with the ams-list command.

```
oracle>> ams-list
```
```
<<ams-list,1.netdemo|Oracle,STOPPED
<<ams-list,2.rs232dem|Oracle,RUNNING
<<ams-list,OK,2 suites are installed
```

Finally, start up the IMlet using the ams-run command. The application can be terminated with the ams-stop command.

```
oracle>> ams-run 1
<<ams-run,OK,started
```
```
oracle>> ams-list
```
```
<<ams-list,1.netdemo|Oracle,RUNNING
<<ams-list,2.rs232dem|Oracle,RUNNING
<<ams-list,OK,2 suites are installed
```

In order to check the current WiFi settings of the board, activate the system menu commands and use the net info command.

```
oracle>> sysmenu on
oracle>> net info
```
```
<<net.info,Address=192.168.1.103
<<net.info,SSID=network001
<<net.info,Preference=0
<<net.info,PDP APN=wap.cingular
<<net.info,PDP Auth Type=0
<<net.info,PDP Auth Username=user
<<net.info,PDP Auth Password=password
<<net.info,OK,success getting info
```

Networking Mode

The Qualcomm Orion Internet-of-Everything (IoE) can be configured for tooling over networking as well as serial. This allows the user to bypass the USB serial connections.
when connecting to the board. However, in order to do this, the board must first be configured for networking first.

**Configuring Wi-Fi networking**

WiFi access must first be configured using the CLI. Connect the to CLI using the instructions above and perform the following steps:

1. Activate the system commands by entering the following CLI command: `sysmenu on`
2. Set the SSID of the WiFi network using the command: `net set ssid [SSID Name]`
3. Set the security password of the SSID network using the command `net set passwd [WiFi Password]`. This is only necessary if the security of WiFi network is enabled.
4. Enter the `net reconnect` command to apply the settings. Note that the Oracle Java ME SDK 3.4 Device Manager will temporarily lose its connection to the Qualcomm IoE device while it resets.

Once completed, you can verify the settings on the board by connecting to the CLI and performing the following steps.

1. Activate the system commands using the command: `sysmenu on`
2. Verify the network settings and state using the command: `net info`

If IP address is 0.0.0.0, then the connection to the WiFi network was not established successfully; check the network settings and try again. If the SSID and password of network are correct, then try to reset the board to re-initialize the WiFi. You can use “IP Address Periodic Logging” feature described below to see the IP address that has been assigned.

---

**Note:** Each time you make a connection to the Java Logger with PuTTY, you will see logs labeled with [NetSetup] channel. The logs contain information about a connection result to a WiFi access point specified earlier. If you can't connect to the Java Logger, try connecting to the board with Brew MP Loader and finding the same logs in the `/shared/netlog.txt` file.

If you see a valid IP address, then the network is configured successfully. At this point, you can reset the following option in the `jwc_properties.ini` file.

```ini
com.oracle.tooling.mode = network
```

Finally, restart Java again using the `net reconnect` command and connect to port 65000 and 65002 using the IP address that has been assigned to the board.

**IP Address Periodic Logging**

The Oracle Java ME Embedded implementation has the ability to log the IP address that has been issued to the board. The log can be seen through Java Logger and Brew MP SDK Logger application.

The behavior of this feature is controlled by `com.oracle.periodic.logging.interval` property that accepts the values counter in milliseconds. By default, the logging period is set to 10 seconds (10000 milliseconds). To disable the periodic logging, set the value of the property to 0.
TCP-to-Serial Fallback

If you are using the tooling network mode and the Wi-Fi connection does not work, you can also try the TCP-to-Serial Fallback functionality provided by the Oracle Java ME Embedded software. The feature switches the CLI to serial mode for this session in order to allow the user to identify the problem and fix it.

To leverage the feature, make sure that:

- The Device Manager is executing and trying to connect to address "127.0.0.1" (see the “Tooling Over Serial” section earlier). Note that the state of Device Manager's connection to "127.0.0.1" will not be changed to "connected" because the ODT port will be still in the network mode.
- Ensure that the com.oracle.midp.ams.headless.cli.tcp2comm.fallback property from is set to true. This value is the default.
- Ensure that the com.oracle.midp.ams.headless.cli.reconnect.timeout is set to 0. This is also the default.
- Attempt to open the CLI as normal. Note that sometimes it can take up to a couple of minutes until the runtime switches the CLI to serial mode for the session.

If the Device Manager wasn't connecting to a COM port when the problem of connecting over TCP/IP occurred, then try resetting both Device Manager and the board.

Automatic Recovering for Java Logger and CLI Connections

If you are using the tooling network mode, or you are going to deploy an application that is based on Oracle Java ME Embedded 3.4 somewhere remotely, then it is worthwhile to turn on a feature that will automatically reopen both the Java Logger and CLI socket connections on the board side in the event that a network related error occurs.

- To turn on the CLI's auto-recovering feature, make sure that the com.oracle.midp.ams.headless.cli.reconnect.timeout property is set to a value, which is measured in milliseconds.
- To turn on Java Logger's auto-recovering feature, make sure that log.tcp.reconnect.timeout is set to a value, which is measured in milliseconds.
- If both of these properties are set to 0, then the feature is turned off.

Note that there can be situations when the underlying platform will not be able to recover a network subsystem after some network-related issue occurs. In such cases your application should be designed in a way that allows it to:

- Catch network-related issues and reopen application-level network connections
- Reboot the board using Device Access API's watchdog if an app assumes that the network subsystem is unrecoverable.
Installing and Running Applications on the Qualcomm IoE Board

Developers can run and debug IMlets on the Qualcomm IoE board in serial mode using the Oracle Java ME SDK, or in networking mode without Java ME SDK usage, either using the CLI or directly from IDEs such as NetBeans or Eclipse. This chapter describes how to install and run an IMlet on the board, as well as debugging an IMLet in both the NetBeans and Eclipse IDE.

Using NetBeans with the Qualcomm IoE Board

Installing and running IMlet projects on the Qualcomm IoE board using the NetBeans IDE requires the following software:

- NetBeans IDE 7.3.1 and later with Java ME, which can be downloaded from http://www.netbeans.org/.
- Oracle Java ME SDK 3.4
- Oracle Java ME SDK 3.4 NetBeans Plugin

Installing the Oracle Java ME SDK 3.4 Plugin for NetBeans

After installing NetBeans, use these steps to install the remaining software.

1. Ensure that Java ME is enabled in NetBeans. This can be done by selection Tools -> Plugins and selecting the Installed pane. Activate the Java ME plugin if it is not already activated.

2. Install the Java ME SDK 3.4 distribution, if you have not done so already. See the Java ME SDK 3.4 documentation for details.

3. Install the Oracle Java ME SDK 3.4 NetBeans plugin. This is a downloadable ZIP file that consists of a number of NetBeans modules (.nbm files) that can be added using the Tools -> Plugins dialog and selecting the Downloaded pane. Unzip the plugin file, and add all of the .nbm files to NetBeans. The Oracle Java ME SDK 3.4 NetBeans plugins are required to interface with the Device Selector and connect to the board.

4. Ensure that the Oracle Java ME Embedded 3.4 appears in the list of Java ME platforms. In the NetBeans IDE, go to Tools -> Java Platforms. If the Oracle Java Platform Micro Edition SDK 3.4 does not appear in the list of J2ME platforms, follow these steps:
   - Click on Add Platform.
   - Select Java ME CLDC Platform Emulator and click Next.
Select the folder where the Oracle Java ME SDK 3.4 distribution resides and follow the instructions to install it. Then, click **Finish** to close the dialog.

5. Ensure that the Qualcomm IoE board has the Oracle Java ME Embedded distribution. See Chapter 1, "Installing Oracle Java ME Embedded on the Qualcomm IoE Board" for more information on how to install the runtime distribution on the Qualcomm IoE board.

### Adding the Qualcomm IoE Board to the Device Selector

Follow these steps to add the board to the Device Selector in NetBeans:

1. Ensure that the property `odt_run_on_start` has been set to `true` on the Qualcomm IoE.

2. Start the NetBeans IDE. In the NetBeans IDE, go to **Tools** -> **Java ME** -> **Device Selector**

3. If the device is not already listed in the Device Selector, click on the **Add a Device** button at the top of the Device Selector window, as shown in **Figure 3–1**.

**Figure 3–1 NetBeans Device Selector "Add a Device" Button**

4. If you are using tooling over the serial mode then the connection between the IoE board and NetBeans should appear automatically in the list in **Figure 3–1**. The IP address of the connection will be ‘127.0.0.1’. If it is not already shown, enter the IP address of the Qualcomm IoE board in the **IP Address** field, as shown in **Figure 3–2**, and click **Next**.
Using NetBeans with the Qualcomm IoE Board

Installing and Running Applications on the Qualcomm IoE Board

5. Once the device is detected, click Finish on the Device Detection screen. The list of devices in the Device Selector should now include IMPNGExternalDevice1. Alternatively, you can use the Oracle Java ME SDK 3.4 to detect the device, as shown in Chapter 2, “Tooling Over Serial and Networking”. If the device has already been detected using the Oracle Java ME SDK 3.4, it should already appear in the Device Selector.

Assigning the Qualcomm IoE Board to Your Project

If you already have an existing NetBeans project with an IMlet that you want to run or debug on the board, follow these steps:

1. Right-click on your project and choose Properties.
2. Select the Platform category on the properties window.
3. Select the entry that represents the board (IMPNGExternalDevice1) from the device list.

If you are creating a new NetBeans project from scratch, follow these steps:

1. Select File -> New Project.
2. Select the Java ME category and Embedded Application in Projects. Click Next.
3. Provide a project name and click Next. Be sure that the Create Default Package and IMlet Class option is checked.
4. Ensure the Emulator Platform is Oracle Java ME Embedded 3.4. Then, select the entry that represents the board (IMPNGExternalDevice1) from the device list and click Finish.

The configured Platform dialog is shown in Figure 3–3. After you assign the board to your project, the IMlets run on the board instead of on the emulator when you click on Run Project on the NetBeans IDE.
Sample Source Code

Once the project is created, use the source code in Example 3–1 for the default IMlet.java source file.

**Example 3–1  Sample Code to Access a GPIO Pin**

```java
package embeddedapplication1;

import com.oracle.deviceaccess.PeripheralManager;
import com.oracle.deviceaccess.PeripheralNotAvailableException;
import com.oracle.deviceaccess.PeripheralNotFoundException;
import com.oracle.deviceaccess.gpio.GPIOPin;
import java.io.IOException;
import javax.microedition.midlet.*;

public class IMlet extends MIDlet {
    public void startApp() {
        try {
            GPIOPin pin = (GPIOPin) PeripheralManager.open(14);
            for (int i = 0; i < 10; i++) {
                pin.setValue(true);
                Thread.sleep(1000);
                pin.setValue(false);
                Thread.sleep(1000);
            }
        } catch (IOException e) {
            e.printStackTrace();
        } catch (PeripheralNotAvailableException e) {
            e.printStackTrace();
        } catch (PeripheralNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```
Debugging an IMlet on the Qualcomm IoE Board

Follow these steps to debug an IMlet using NetBeans:

1. Open your IMlet class on the NetBeans editor.
2. Click once directly on the line number where you want to set a breakpoint. The line number is replaced by a red square and the line is highlighted in red.
3. Select Debug -> Debug Project or use the Debug button on the toolbar.

The debugger connects to the debug agent on the board and the program execution stops at your breakpoint, as shown in Figure 3–4.

This sample application will obtain an object representing GPIO pin 14 from the PeripheralManager, and set it from low to high at intervals of one second. This has the effect of blinking one of the LEDs on the Qualcomm IoE board. For more information on using the Device Access APIs, see the Device Access API (Version B) Guide and the associated Javadocs at:

http://docs.oracle.com/javame/embedded/embedded.html
Figure 3–4 shown an entire NetBeans debugging environment that allows the programmer to execute a program step by step as well as add and remove variables from a watch list on the bottom of the screen.

For more information on using the Device Access APIs, see the Device Access API (Version B) Guide and the associated Javadocs at:

http://docs.oracle.com/javame/embedded/embedded.html

Using Eclipse with the Qualcomm IoE Board

Running and debugging IMlet projects on the Qualcomm IoE board using the Eclipse IDE requires the following software:

- Eclipse 3.7 Indigo or Eclipse 4.2 Juno, which can be downloaded from http://www.eclipse.org/.
- Oracle Java ME SDK 3.4
- Oracle Java ME SDK 3.4 Eclipse Plugin

Installing the Oracle Java ME SDK 3.4 Plugin for Eclipse

After installing Eclipse, use these steps to install the remaining software.

1. Install the Java ME SDK 3.4 distribution, if you have not done so already. See the Java ME SDK 3.4 documentation for details.

2. Install the Oracle Java ME SDK 3.4 Eclipse plugin. This is required to use the Device Selector to connect to the board.

3. Ensure that the Qualcomm IoE board has the Oracle Java ME Embedded 3.4 runtime. See Chapter 1, "Installing Oracle Java ME Embedded on the Qualcomm IoE Board" for more information on how to install the runtime distribution on the Qualcomm IoE board.
4. Ensure that the Oracle Java ME Embedded 3.4 appears in the list of Java ME platforms. If it doesn’t appear, open "Window->Preferences" and follow these steps:

- Under the **Java ME** category, select **Device Management**. In the Device Management window, press the **Manual Install...** button.
- The Manual Device Installation window appears, without the Oracle Java ME Embedded devices. Press the **Browse** button. A browser window appears.
- Browse to the base directory of the Java ME SDK environment and press the **OK** button. After the platform is scanned and the devices are installed, close each of the respective dialogs.

**Adding the Qualcomm IoE Board to the Device Selector**

Follow these steps to add the board to the Device Selector in the Oracle Java ME SDK:

1. **Ensure that the property `odt_run_on_start` is set to true in the file `jwc_properties.ini` on the Qualcomm IoE.**

2. **Start the Eclipse IDE. In the Eclipse IDE, go to **Window** -> **Show View** -> **Other**. In the popup window that appears, expand the **Java ME** node and select **Device Selector**.**

3. **On the Device Selector, click on the **Add a Device** button at the top of the Device Selector window.**
4. If you are using tooling over the serial mode then the connection between the IoE board and NetBeans should appear automatically in the list in the figure above. If it is not listed already, enter the IP address of the Qualcomm IoE board in the IP Address field and click Next.

5. Once the device is detected, click Finish on the Add Device screen. The list of devices in the Device Selector should now include IMPNGExternalDevice1.

Assigning the Qualcomm IoE Board to Your Project

If you already have an existing Eclipse project with an IMlet that you want to run or debug on the board, follow these steps:

1. Right-click on your project and choose Properties.
2. Select the Java ME category on the properties window.
3. Select the device entry for the Qualcomm IoE board (IMPNGExternalDevice1) from the device list. If the device is not shown, add it using the Add... button, selecting Oracle Java ME Embedded 3.4 as the SDK and choosing the appropriate device that represents the Qualcomm IoE board.

If you are creating a new Eclipse project from scratch, follow these steps:

1. Select New -> Other. Then expand the Java ME tree node, and create a new MIDlet Project.
2. Expand the Java ME tree node, and create a new MIDlet Project.
3. In the Configuration pane of the creation dialog, select the appropriate entry (IMPNGExternalDevice1) from the device list.
4. Select the appropriate Profile and Configuration for your project.

After you assign the board to your project, the IMlets run on the board instead of on the emulator when you click on Project -> Run on the Eclipse IDE.
Sample Source Code

Once the project is created, use the source code given earlier in Example 3–1 for a default source file. This sample application will obtain an object representing GPIO pin 14 from the PeripheralManager, and set it from low to high at intervals of one second. This has the effect of blinking one of the LEDs on the Qualcomm IoE board.

Debugging an IMlet on the Qualcomm IoE Board

After you assign the board to your project, follow these steps to debug an IMlet:

1. Open your IMlet class on the Eclipse editor.
2. Click once directly on the line number where you want to set a breakpoint. The line number has a small circle next to it to indicate a breakpoint.
3. Select Run -> Debug or use the Debug button on the toolbar.

The debugger connects to the debug agent on the board and the program execution stops at your breakpoint, as shown in Figure 3–5.

Figure 3–5  Debugging an IMlet on the Board Using the Eclipse IDE

Figure 3–5 shown an entire Eclipse debugging environment that allows the programmer to execute a program step by step as well as add and remove variables from a watch list on the bottom of the screen.

For more information on using the Device Access APIs, please see the Device Access API (Version B) Guide and the associated Javadocs at the following site:

http://docs.oracle.com/javame/embedded/embedded.html
Installing and Running an IMlet Using the AMS CLI

If you are not using an IDE, you can still use the Oracle Java ME Embedded 3.4 CLI to install an application. Simply connect to the device at port 65002, and install and run the IMlet manually. For example:

oracle>> ams-install file:///Shared/hello.jar
<<ams-install,start install,file:///Shared/hello.jar
<<ams-install,install status: stage 0, 5%
<<ams-install,install status: stage 3, 100%
<<ams-install,install status: stage 4, 100%
<<ams-install,OK,Install success

oracle>> ams-list
<<ams-list,0.hello|Oracle,STOPPED
<<ams-list,OK,1 suites are installed

oracle>> ams-run 0
<<ams-run,OK,started

oracle>> ams-list
<<ams-list,1.netdemo|Oracle,RUNNING
<<ams-list,OK,1 suites are installed


Accessing Peripherals

Note that if an application is installed on the board using NetBeans or Eclipse during development, the application will automatically be installed in the maximum security domain as a convenience. IMlets that are not running through the NetBeans or Eclipse IDE, however, must have permission to access device peripherals using the Device Access APIs. For more information on using the device access APIs, please see the Device Access API (version B) Guide and the associated Javadocs at the following site: http://docs.oracle.com/javame/embedded/embedded.html

There are two ways to allow access to the device peripherals on the Qualcomm IoE. The first is to digitally sign the application with the appropriate API permissions requested in the JAD file; the second is to use unsigned applications and modify the security policy file.

Method #1: Signing the Application with API Permissions

The first method is more complex, but is the preferred route for applications that are widely distributed. First, the JAD file must have the proper API permissions. Here is how to sign the application in NetBeans, Eclipse, and without an IDE.

- In NetBeans, right-click the project name (EmbeddedApplication1 in this example) and choose Properties. Select Application Descriptor, then in the resulting pane, select API Permissions. Click the Add... button, and add the com.oracle.deviceaccess.gpio API, as shown in Figure 3–6. Click OK to close the project properties dialog.
In Eclipse, open the Application Descriptor for your project in the Packages window, and select the Application Descriptor pane. You will need to manually add or change the following lines in the Application Descriptor.

- MIDlet-Permissions: com.oracle.deviceaccess.gpio
- Microedition-Profile: IMP-NG

If you are not using an IDE, manually modify the application descriptor file to contain the permissions listed in the Eclipse section.

Here are the instructions on how to setup a keystore with a local certificate that can be used to sign the applications.

1. Generate a new self-signed certificate with the following command on the desktop, using the keytool that is shipped with the Java SE JDK.

   ```
   keytool -genkey -v -alias mycert -keystore mykeystore.ks -storepass spass -keypass kpass -validity 360 -keyalg rsa -keysize 2048 -dname "CN=thehost"
   ```

   This command will generate a 2048-bit RSA key pair and a self-signed certificate, placing them in a new keystore with a keystore password of `spass` and a key password of `kpass` that is valid for 360 days. Feel free to change both passwords as you see fit.

2. Copy the appdb/_main.ks keystore file from the Qualcomm IoE over to the desktop using the Loader tool and perform the following command using the mekeytool.exe command (or alternatively java -jar MEKeyTool.jar... if your distribution contains only that) that ships with the Oracle Java ME SDK 3.4 distribution.

   ```
   {mekeytool} -import -MEkeystore _main.ks -keystore mykeystore.ks -storepass spass -alias mycert -domain trusted
   ```
This will import the information in mykeystore.ks you just created to the _main.ks keystore. Once this is completed, copy the _main.ks file back to its original location on the Qualcomm IoE using the Loader tool.

Use the following steps in to sign your application before deploying to the Qualcomm IoE board, depending on whether you are using NetBeans, Eclipse, or working without an IDE.

- **In NetBeans**, perform the following steps.
  1. Right click your project and select **Properties**.
  2. Choose the **Signing** option under the **Build** category.
  3. Open the **Keystores Manager** and import the mykeystore.ks file that you created.
  4. Check the **Sign Distribution** box. If you wish, unlock the keystore and the key with the passwords that you specified earlier. This is shown in Figure 3–7.
  5. When the project is built and run, it will be digitally signed and deployed to the Qualcomm IoE.

*Figure 3–7  The Signing Pane in the NetBeans Project Properties*

- **In Eclipse**, perform the following steps:
  1. Right click your project and select **Properties**.
  2. Choose the **Signing** option under the **Java ME** category.
  3. Check the **Enable Project Specific Settings** checkbox. Import the mykeystore.ks file that you created as an External... keystore file. Provide the keystore and key passwords that you created earlier. Ensure that the mycert key alias is present.
4. Ensure that the project is being signed in the project’s Application Descriptor. When the project is built and run, it will be digitally signed when deployed to the Qualcomm IoE.

- If you are not using an IDE, enter the following command to sign a JAD:

  ```
  > jarsigner -keystore mykeystore.ks -storepass spass app.jad myalias
  ```

  If there is an issue with a non-valid certificate, be sure to check the date and time which has been setup on the board. Refer to Chapter 4, "Troubleshooting" for more details.

**Method #2: Modifying the Security Policy File**

With this method, the user will modify the Java security policy file. Typically, this is the `appdb/_policy.txt` file, but be sure to check the `security.policyfile` entry in the `jwc_properties.ini` file to verify the current file name.

Perform the following steps:

1. Use the **Loader** tool to download the `appdb/_policy.txt` file to the desktop.
2. Add the line "allow: device_access" to the "untrusted" domain (often paired with "unsecured")
3. Use the **Loader** application to copy the file back, overwriting the original file in the `appdb/` directory.

Remember that if an application is installed on the board using the NetBeans or Eclipse IDE during development, the application will automatically be installed in the **maximum** security domain as a convenience. Manual installation using the AMS, however, will place the unsigned application into the **untrusted** security domain.

After development is finished, you should **always** publish your applications with signed API permissions.
This chapter contains a list of common problems that you may encounter while installing and running the Oracle Java ME SDK and embedded software on the Qualcomm IoE board. This chapter provides information on the causes of these problems and possible solutions for them.

The common problems in this chapter are grouped in two categories:
- Starting Oracle Java ME Embedded on the Board
- Using the Board with the Oracle Java ME SDK and the NetBeans IDE

## Starting Oracle Java ME Embedded on the Board

Table 4–1 contains information about problems and solutions when starting the runtime on the board.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows does not recognize the board when</td>
<td>Ensure the USB drivers are</td>
<td>See Chapter 1, “Installing Oracle Java ME Embedded on the Qualcomm IoE</td>
</tr>
<tr>
<td>connected via USB.</td>
<td>loaded.</td>
<td>Board” for more information on installing the USB drivers for the Qualcomm IoE board.</td>
</tr>
<tr>
<td>Windows does not recognize the board when</td>
<td>Ensure the board is powered on.</td>
<td>Press the PWR KEY button on the board.</td>
</tr>
<tr>
<td>connected via USB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oracle Java ME Embedded fails to initialize</td>
<td>The network configuration is</td>
<td>Check that the network connection on the board is correct. Ensure that</td>
</tr>
<tr>
<td>the network on the board.</td>
<td>incorrect.</td>
<td>the board is using DHCP to obtain an IP address.</td>
</tr>
<tr>
<td>(continued)</td>
<td>Network configuration on the</td>
<td>Check that WiFi SSID broadcasting is enabled on the router.</td>
</tr>
<tr>
<td></td>
<td>WiFi access point is incorrect</td>
<td></td>
</tr>
<tr>
<td></td>
<td>or unsupported.</td>
<td></td>
</tr>
</tbody>
</table>
Table 4–1 (Cont.) Problems and Solutions - Starting Oracle Java ME Embedded on the Board

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>(continued)</td>
<td>WPA2-PSK authentication algorithm is not working correctly on some routers.</td>
<td>Try manually setting the authentication algorithm to WPA-PSK or disabling the security checking.</td>
</tr>
<tr>
<td>There are no Java logs in Brew MP Logger application. However, the device has been recognized successfully by the Logger and Loader application.</td>
<td>The Oracle Java ME Embedded platform did not start. The java.sig or netsetup.sig files were not updated or are invalid for Oracle Java ME Embedded 3.4</td>
<td>Check the netsetup logs. Please refer to Configuring Wi-Fi networking in Chapter 2, “Tooling Over Serial and Networking” for additional information. If there is no netlog.txt file or it is incorrect, please ensure that Netsetup application has been updated and the netsetup.sig file is valid. The signature file from previous versions of the Oracle Java ME Embedded platform will not work correctly. If the netlog.txt file is present and is correct, ensure that Java ME Embedded application in the /sys/mod/java directory has been updated and the java.sig file is valid. The Java signature file from the previous version must also be updated for 3.4.</td>
</tr>
<tr>
<td>The board is not detected by the Device Manager when connecting to the board in serial mode.</td>
<td>Varies. See Solution column.</td>
<td>If you have some issues with connecting Device Manager and the board, please check the file &lt;USER_HOME_DIR&gt;/javame-sdk/3.4/log/sos-proxy.log. 1) if the last line in the output looks similar to &quot;Open COM{Number}&quot;, then make sure that you’ve specified the correct COM port. If the port is incorrect, choose the proper one by specifying it in Device Manager and restarting the board. Note that is sometimes take a minute or more to boot Java after the board is powered on. This is because of WiFi related settings that are performed during the launch time. If you don’t use WiFi (3G), then you can disable the network setup with net related commands. In addition, there is a &quot;system.netsetup.timeout&quot; property that configures the timeout to start Java after the network initialization has been started. If the COM port is correct and more then a minute has passed after the board is powered on, try to reboot both Device Manager and the board 2) If the last line in the output is not &quot;Open COM{Number}&quot;, check that you specified the correct port numbers to connect to the CLI or the logger, then try to reboot both the Device Manager and the board.</td>
</tr>
<tr>
<td>Problem</td>
<td>Cause</td>
<td>Solution</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>The board is not detected when adding a new device to the Device Manager.</td>
<td>On-device debugging is not enabled.</td>
<td>Edit the file jwc_properties.ini and set the property odt_run_on_start to true.</td>
</tr>
<tr>
<td>The debugging session freezes, disconnects unexpectedly, or shows error messages.</td>
<td>The firewall on the computer is blocking some debugging traffic. Thunderbird is using a port that is needed for communication with the board.</td>
<td>Open TCP port 2808 on your firewall configuration settings. The exact procedure to open a port differs depending on your version of Windows or your firewall software. Close thunderbird.exe during the debugging session.</td>
</tr>
<tr>
<td>The current time and date is invalid.</td>
<td>The time and date on the Qualcomm IoE board is configured automatically only if a valid SIM card is inserted.</td>
<td>Insert a valid SIM card. If the date and time is still wrong, try another SIM card.</td>
</tr>
<tr>
<td>A signed IMlet will not install. The AMS gives a return code that the certificate or the authentication are invalid.</td>
<td>Certificate is invalid or it is not added to keystore</td>
<td>Check the certificate. Refer to Method #1: Signing the Application with API Permissions in Chapter 3, &quot;Installing and Running Applications on the Qualcomm IoE Board&quot; for details.</td>
</tr>
<tr>
<td>(continued)</td>
<td>The date and time on the board were configured incorrectly.</td>
<td>Refer to note above about setting date and time on Qualcomm IoE board.</td>
</tr>
</tbody>
</table>
This appendix provides information about the various peripheral ports and buses for the Qualcomm IoE embedded board, as well as device mappings and important notes, which are accessible using the Device Access APIs.

---

**Note:** Power Management and MMIO are not supported on the Qualcomm IoE embedded board.

---

The tables use the following legend:

- **DA API Peripheral Id** - an integer identifier that can be used to open the peripheral with a `PeripheralManager`.
- **DA API Peripheral Name** - the string name of a peripheral that can be used to open it by name with `PeripheralManager`.
- **Mapped To** - all hardware related information regarding a peripheral, such as physical location, mapping, or port. This information allows the user to find out the peripheral’s location on a target board. See the following site for more information:


- **Configuration** - properties that are passed to the specific `PeripheralConfig` constructor in order to open the peripheral by ID or name. The configuration can be used to open the peripheral using the `PeripheralManager` with the appropriate configuration.

### AT Devices

The following AT devices are pre-configured.

<table>
<thead>
<tr>
<th>DA API Peripheral ID</th>
<th>DA API Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>800</td>
<td>DEFAULT</td>
<td>Brew’s AT command interface</td>
<td>N/A</td>
</tr>
</tbody>
</table>
For a complete list of AT commands that can be used, please see the Qualcomm IoE Development Platform User’s Guide.

Please note the following when using AT commands:

- Some AT commands required a SIM card to test. (for example, "AT+CPBW" or "AT+CMUX")
- With the AT+CPBW command, the valid form is "AT+CPBW=?" or "AT+CPBW=<num>". "AT+CPBW?" is an invalid form.
- UnsolicitedResponseHandler is not supported.

**ADC**

The following Analog-to-Digital (ADC) devices are pre-configured.

<table>
<thead>
<tr>
<th>DAAPI Peripheral ID</th>
<th>DAAPI Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>converterNumber = PeripheralConfig.DEFAULT</td>
</tr>
<tr>
<td>100</td>
<td>VETHERM_N</td>
<td>On-board Temperature Sensor</td>
<td>channelNumber = 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>resolution = PeripheralConfig.DEFAULT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>samplingInterval = PeripheralConfig.DEFAULT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>samplingTime = PeripheralConfig.DEFAULT</td>
</tr>
<tr>
<td>101</td>
<td>HKAIN1</td>
<td>Any pin from J10 header.</td>
<td>converterNumber = PeripheralConfig.DEFAULT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>channelNumber = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>resolution = PeripheralConfig.DEFAULT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>samplingInterval = PeripheralConfig.DEFAULT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>samplingTime = PeripheralConfig.DEFAULT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>samplingInterval = PeripheralConfig.DEFAULT</td>
</tr>
</tbody>
</table>

Please note the following:

- The resolution and converter number are not supported. You can use PeripheralConfig.DEFAULT for those values.
- The channel number cannot be set as PeripheralConfig.DEFAULT. Use 0 or 1 instead.
- The default value for samplingInterval is 5000000 microseconds (5000 ms). The value can be changed immediately during acquisition. This is a platform-specific behavior.
- The samplingTime value of PeripheralConfig.DEFAULT is interpreted as 10 milliseconds.

**DAC**

The following Digital-to-Analog (DAC) devices are pre-configured.
Please note the following:

- Both the resolution and converter values are ignored. You can use PeripheralConfig.DEFAULT for those values. In addition, converterNumber and resolution can only be PeripheralConfig.DEFAULT.

- The default samplingInterval value is 5000000 micro seconds (5000 ms). The value can be changed immediately during generation. This is a platform-specific behavior.

- The channel parameter cannot be set as PeripheralConfig.DEFAULT only be 0 or 1.

- The DAC signal is represented as a PDM (pulse density modulation) signal, so the DAC output value affects only the frequency of output signal, not the voltage level. As such, there is no resolution of DAC in the current implementation, only the min and max value can be used for calculation of output voltage on the DAC channel.

For calculation of the output voltage, the following formula can be used: \( v_{\text{output}} = \frac{\text{value} \times v_{\text{Ref}}}{(\text{maxValue} - \text{minValue} + 1)}. \) Note that \( (\text{max} - \text{min}) = (2^n - 1) \) is not applicable.

### GPIO Pins

The following GPIO pins are pre-configured.

<table>
<thead>
<tr>
<th>DAAP Peripheral ID</th>
<th>DAAP Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>700</td>
<td>PDM0</td>
<td>Header J5 pin 18</td>
<td>converter = PeripheralConfig.DEFAULT, channel = 0, resolution = PeripheralConfig.DEFAULT, samplingInterval = PeripheralConfig.DEFAULT</td>
</tr>
<tr>
<td>701</td>
<td>PDM1</td>
<td>Reserved for future use; not available on Qualcomm IoE</td>
<td>converter = PeripheralConfig.DEFAULT, channel = 1, resolution = PeripheralConfig.DEFAULT, samplingInterval = PeripheralConfig.DEFAULT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DAAP Peripheral ID</th>
<th>DAAP Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>GPIO0</td>
<td>Header J5 pin 3</td>
<td>portNumber = PeripheralConfig.DEFAULT, pinNumber = 26, direction = GPIOPinConfig.DIR_INPUT ONLY, mode = PeripheralConfig.DEFAULT, trigger = GPIOPinConfig.TRIGGER RISING_EDGE, initValue - ignored</td>
</tr>
<tr>
<td>DAAPI Peripheral ID</td>
<td>DAAPI Peripheral Name</td>
<td>Mapped To</td>
<td>Configuration</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------</td>
<td>-----------</td>
<td>---------------</td>
</tr>
<tr>
<td>1</td>
<td>GPIO1</td>
<td>Header J5 pin 5</td>
<td>portNumber = PeripheralConfig.DEFAULT</td>
</tr>
<tr>
<td></td>
<td>JP7 ADC MUX 0</td>
<td></td>
<td>pinNumber = 25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>direction = GPIOPinConfig.DIR_INPUT_ONLY</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>mode = PeripheralConfig.DEFAULT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>trigger = GPIOPinConfig.TRIGGER_RISING_EDGE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>initValue - ignored</td>
</tr>
<tr>
<td>2</td>
<td>GPIO2</td>
<td>Header J5 pin 7</td>
<td>portNumber = PeripheralConfig.DEFAULT</td>
</tr>
<tr>
<td></td>
<td>JP8 ADC MUX 1</td>
<td></td>
<td>pinNumber = 31</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>direction = GPIOPinConfig.DIR_INPUT_ONLY</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>mode = PeripheralConfig.DEFAULT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>trigger = GPIOPinConfig.TRIGGER_FALLING_EDGE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>initValue - ignored</td>
</tr>
<tr>
<td>3</td>
<td>GPIO3</td>
<td>Header J5 pin 9</td>
<td>portNumber = PeripheralConfig.DEFAULT</td>
</tr>
<tr>
<td></td>
<td>JP9 ADC MUX 2</td>
<td></td>
<td>pinNumber = 17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>direction = GPIOPinConfig.DIR_INPUT_ONLY</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>mode = PeripheralConfig.DEFAULT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>trigger = GPIOPinConfig.TRIGGER_HIGH_LEVEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>initValue - ignored</td>
</tr>
<tr>
<td>4</td>
<td>GPIO4</td>
<td>Header J5 pin 11</td>
<td>portNumber = PeripheralConfig.DEFAULT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>pinNumber = 32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>direction = GPIOPinConfig.DIR_INPUT_ONLY</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>mode = PeripheralConfig.DEFAULT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>trigger = GPIOPinConfig.TRIGGER_HIGH_LEVEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>initValue - ignored</td>
</tr>
<tr>
<td>5</td>
<td>GPIO5</td>
<td>Header J6 pin 3</td>
<td>portNumber = PeripheralConfig.DEFAULT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>pinNumber = 28</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>direction = GPIOPinConfig.DIR_INPUT_ONLY</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>mode = PeripheralConfig.DEFAULT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>trigger = GPIOPinConfig.TRIGGER_HIGH_LEVEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>initValue - ignored</td>
</tr>
<tr>
<td>DAAPI Peripheral ID</td>
<td>DAAPI Peripheral Name</td>
<td>Mapped To</td>
<td>Configuration</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------</td>
<td>-----------</td>
<td>---------------</td>
</tr>
</tbody>
</table>
| 6                   | GPIO6                 | Header J6 pin 5 | portNumber = PeripheralConfig.DEFAULT  
|                     |                       | JP11 pin 2(to connect to G-sensor Interrupt) | pinNumber = 27  
direction = GPIOPinConfig.DIR_INPUT_ONLY  
mode = PeripheralConfig.DEFAULT  
trigger = GPIOPinConfig.TRIGGER_RISING_EDGE  
initValue - ignored |
| 7                   | GPIO7                 | Header J6 pin 7 | portNumber = PeripheralConfig.DEFAULT  
|                     |                       | JP12 pin 2 (to connect to light sensor interrupt) | pinNumber = 30  
direction = GPIOPinConfig.DIR_INPUT_ONLY  
mode = PeripheralConfig.DEFAULT  
trigger = GPIOPinConfig.TRIGGER_FALLING_EDGE  
initValue - ignored |
| 8                   | GPIO8                 | Header J6 pin 7 | portNumber = PeripheralConfig.DEFAULT  
|                     |                       |                     | pinNumber = 38  
direction = GPIOPinConfig.DIR_INPUT_ONLY  
mode = PeripheralConfig.DEFAULT  
trigger = GPIOPinConfig.TRIGGER_LOW_LEVEL  
initValue - ignored |
| 9                   | GPIO9                 | Header J6 pin 11 | portNumber = PeripheralConfig.DEFAULT  
|                     |                       | JP13 pin 1(to connect to temp sensor interrupt) | pinNumber = 33  
direction = GPIOPinConfig.DIR_INPUT_ONLY  
mode = PeripheralConfig.DEFAULT  
trigger = GPIOPinConfig.TRIGGER_FALLING_EDGE  
initValue - ignored |
| 10                  | GPIO10                | Header J7 pin 1 | portNumber = PeripheralConfig.DEFAULT  
|                     | Relay 1               |                     | pinNumber = 18  
direction = GPIOPinConfig.DIR_OUTPUT_ONLY  
mode = PeripheralConfig.DEFAULT  
trigger = GPIOPinConfig.TRIGGER_NONE  
initValue = false |
<table>
<thead>
<tr>
<th>DAAPI Peripheral ID</th>
<th>DAAPI Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
</table>
| 11                  | GPIO11                | Header J7 pin 3 | portNumber = PeripheralConfig.DEFAULT  
|                     |                       | Relay 2    | pinNumber = 24  
|                     |                       |            | direction = GPIOPinConfig.DIR_OUTPUT_ONLY  
|                     |                       |            | mode = PeripheralConfig.DEFAULT  
|                     |                       |            | trigger = GPIOPinConfig.TRIGGER_NONE  
|                     |                       |            | initValue = false  |
| 12                  | GPIO12                | Header J7 pin 5 | portNumber = PeripheralConfig.DEFAULT  
|                     |                       |            | pinNumber = 29  
|                     |                       |            | direction = GPIOPinConfig.DIR_OUTPUT_ONLY  
|                     |                       |            | mode = PeripheralConfig.DEFAULT  
|                     |                       |            | trigger = GPIOPinConfig.TRIGGER_NONE  
|                     |                       |            | initValue = false  |
| 13                  | GPIO13                | Header J7 pin 7 | portNumber = PeripheralConfig.DEFAULT  
|                     |                       |            | pinNumber = 35  
|                     |                       |            | direction = GPIOPinConfig.DIR_OUTPUT_ONLY  
|                     |                       |            | mode = PeripheralConfig.DEFAULT  
|                     |                       |            | trigger = GPIOPinConfig.TRIGGER_NONE  
|                     |                       |            | initValue = false  |
| 14                  | GPIO14                | Header J7 pin 9 | portNumber = PeripheralConfig.DEFAULT  
|                     |                       | Jumper P2 pin 2 | pinNumber = 13  
|                     |                       | (Used by LED) | direction = GPIOPinConfig.DIR_OUTPUT_ONLY  
|                     |                       |            | mode = PeripheralConfig.DEFAULT  
|                     |                       |            | trigger = GPIOPinConfig.TRIGGER_NONE  
<p>|                     |                       |            | initValue = false  |</p>
<table>
<thead>
<tr>
<th>DAAPI Peripheral ID</th>
<th>DAAPI Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
</table>
| 15                  | GPIO15                | Header J7 pin 11 | `portNumber = PeripheralConfig.DEFAULT`  
|                     |                       | Jumper P3 pin 2  | `pinNumber = 34`  
|                     |                       |                | `direction = GPIOPinConfig.DIR_OUTPUT_ONLY`  
|                     |                       |                | `mode = PeripheralConfig.DEFAULT`  
|                     |                       |                | `trigger = GPIOPinConfig.TRIGGER_NONE`  
|                     |                       |                | `initValue = false`  |
| 16                  | GPIO16                | Header J7 pin 13 | `portNumber = Peripheral.Config.DEFAULT`  
|                     |                       | Jumper P4 pin 2  | `pinNumber = 12`  
|                     |                       |                | `direction = GPIOPinConfig.DIR_OUTPUT_ONLY`  
|                     |                       |                | `mode = PeripheralConfig.DEFAULT`  
|                     |                       |                | `trigger = GPIOPinConfig.TRIGGER_NONE`  
|                     |                       |                | `initValue - ignored`  |
| 17                  | GPIO17 or LED2        | Header J7 pin 15 | `portNumber = PeripheralConfig.DEFAULT`  
|                     |                       | Jumper P5 pin 2  | `pinNumber = 16`  
|                     |                       |                | `direction = GPIOPinConfig.DIR_OUTPUT_ONLY`  
|                     |                       |                | `mode = PeripheralConfig.DEFAULT`  
|                     |                       |                | `trigger = GPIOPinConfig.TRIGGER_NONE`  
|                     |                       |                | `initValue = false`  |
| 18                  | GPIO18                | Header J7 pin 17 | `portNumber = PeripheralConfig.DEFAULT`  
|                     |                       | Jumper P6 pin 2  | `pinNumber = 36`  
|                     |                       |                | `direction = GPIOPinConfig.DIR_OUTPUT_ONLY`  
|                     |                       |                | `mode = PeripheralConfig.DEFAULT`  
|                     |                       |                | `trigger = GPIOPinConfig.TRIGGER_NONE`  
|                     |                       |                | `initValue = false`  |
| 19                  | GPIO19                | Header J7 pin 19 | `portNumber = PeripheralConfig.DEFAULT`  
|                     |                       |                | `pinNumber = 15`  
|                     |                       |                | `direction = GPIOPinConfig.DIR_OUTPUT_ONLY`  
|                     |                       |                | `mode = PeripheralConfig.DEFAULT`  
|                     |                       |                | `trigger = GPIOPinConfig.TRIGGER_NONE`  
<p>|                     |                       |                | <code>initValue = false</code>  |</p>
<table>
<thead>
<tr>
<th>DAAP Peripheral ID</th>
<th>DAAP Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
</table>
| 20                | GPIO20               | Header J7 pin 2 | portNumber = PeripheralConfig.DEFAULT  
pinNumber = 10  
direction = GPIOPinConfig.DIR_INPUT_ONLY  
mode = PeripheralConfig.DEFAULT  
trigger = GPIOPinConfig.TRIGGER_LOW_LEVEL  
initValue - ignored |
|                   |                      | DB9 J12 lower pin 3 |               |
| 21                | GPIO21               | Header J7 pin 4 | portNumber = PeripheralConfig.DEFAULT  
pinNumber = 14  
direction = GPIOPinConfig.DIR_INPUT_ONLY  
mode = PeripheralConfig.DEFAULT  
trigger = GPIOPinConfig.TRIGGER_LOW_LEVEL  
initValue - ignored |
|                   |                      | DB9 J12 lower pin 2 |               |
| 22                | GPIO22               | Header J7 pin 6 | portNumber = PeripheralConfig.DEFAULT  
pinNumber = 11  
direction = GPIOPinConfig.DIR_OUTPUT_ONLY  
mode = PeripheralConfig.DEFAULT  
trigger = GPIOPinConfig.TRIGGER_NONE  
initValue = false |
| 23                | GPIO23               | Header J7 pin 8 | portNumber = PeripheralConfig.DEFAULT  
pinNumber = 9  
direction = GPIOPinConfig.DIR_OUTPUT_ONLY  
mode = PeripheralConfig.DEFAULT  
trigger = GPIOPinConfig.TRIGGER_NONE  
initValue = false |
|                   |                      | DB9 J10 lower pin 2 |               |
| 24                | GPIO24               | Header J7 pin 10 | portNumber = PeripheralConfig.DEFAULT  
pinNumber = 37  
direction = GPIOPinConfig.DIR_OUTPUT_ONLY  
mode = PeripheralConfig.DEFAULT  
trigger = GPIOPinConfig.TRIGGER_NONE  
initValue = false |
Please note the following:

- `portNumber` can only be set to `PeripheralConfig.DEFAULT`, so `pinNumber` is the unique identifier of the GPIO pin on the Brew MP platform. Note that `pinNumber` cannot be set to `PeripheralConfig.DEFAULT`.

- Configuration of GPIO mode is not supported by the BrewMP platform, so the `mode` parameter can only be set as `PeripheralConfig.DEFAULT`.

- `TRIGGER_BOTH_EDGES` and `TRIGGER_BOTH_LEVELS` are not supported by the BrewMP platform.

- The voltage of output GPIO 1.8V pins can be 1.79 volts. This value is recognized as `false` on input GPIO pins.

- Some GPIO pins are mapped to several physical pins; this allows the programmer to use a GPIO pin in different ways. For example:
  - GPIO1, GPIO2 and GPIO3 can be used to control ADC multiplexer. Please see the Qualcomm IoE Development Platform User’s Guide at the following link for more information.
  - GPIO6, GPIO7, GPIO9 can be used as interrupt pins for the onboard sensors
  - GPIO10, GPIO11 can be used to control the state of on-board’s relays
  - GPIO14, GPIO15, GPIO16, GPIO17, GPIO18 can be used to drive a signal to the on-board’s leds. In this case leds related must be shorted. The same is true for ”LEDS” port.
  - GPIO20, GPIO21, GPIO24 drives the signal to one of DB9 connectors’ pin that is available on-board.

### GPIO Ports

The following GPIO ports are pre-configured.

<table>
<thead>
<tr>
<th>Peripheral ID</th>
<th>Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>LEDS</td>
<td>Header J7 pin 9 and Jumper P2 LED</td>
<td>direction = GPIOPortConfig.DIR_OUTPUT_ONLY initValue = false pins = 13, 34, 12 mode of pins = PeripheralConfig.DEFAULT trigger of pins = GPIOPinConfig.TRIGGER_NONE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Header J7 pin 11 and Jumper P3 LED</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Header J7 pin 13 and Jumper P4 LED</td>
<td></td>
</tr>
</tbody>
</table>

### I2C

The following configurations can be used to communicate to I2C slaves.
Pulse Counter

The protocol of the on-board battery gauge is unknown.

- The default clock frequency is 400000 Hz, and is represented by `PeripheralConfig.DEFAULT`. 100000 Hz is also supported as a `clockFrequency`.
- `addressSize` must be set either to `PeripheralConfig.DEFAULT` or to 7. 10-bit addressing mode is not supported.
- The I2C bus number can only be set to 1, which is also represented by `PeripheralConfig.DEFAULT`.

**Pulse Counter**

The pulse counter has the following configuration:

<table>
<thead>
<tr>
<th>DAAPID Peripheral ID</th>
<th>DAAPI Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>600</td>
<td>COUNTER</td>
<td>GPIO pin 35, Header J7 pin 7</td>
<td>counterNumber = <code>PeripheralConfig.DEFAULT</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>type = <code>TYPE_RISING_EDGE_ONLY</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GPIO portNumber = <code>PeripheralConfig.DEFAULT</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GPIO pinNumber = 35</td>
</tr>
</tbody>
</table>

Please note the following:

- Only the values `TYPE_RISING_EDGE_ONLY` and `TYPE_FALLING_EDGE_ONLY` are supported for the `type` parameter on BrewMP platform.
The counterNumber parameter can be set only to PeripheralConfig.DEFAULT.

SPI

The SPI has a single static configuration with the following parameters:

<table>
<thead>
<tr>
<th>DAAPID</th>
<th>Peripheral ID</th>
<th>Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>G-SENSOR</td>
<td>On-board</td>
<td>G-Sensor</td>
<td>wordLength = 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>G-Sensor</td>
<td></td>
<td>clockFrequency = 26000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>clockMode = 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>address = 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bitOrdering = PeripheralConfig.BIG_ENDIAN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>busNumber = 1</td>
</tr>
</tbody>
</table>

In order to connect an SPI device, remove JP17 jumper. Refer to section 5.2.3, "SPI" in the Qualcomm IoE Guide for more info. There are also some global SPI related options that are set for all SPI slaves:

- Chip select pin mode (0: chip select de-assert; 1: chip select keep asserted). This value is to 1 by default.
- PeripheralConfig.DEFAULT is passed as busNumber is interpreted as 1 and because of only one SPI bus with number 1 is presented on the Qualcomm IoE, only 1 or PeripheralConfig.DEFAULT value of busNumber is supported.
- clockFrequency cannot be set to PeripheralConfig.DEFAULT.
- wordLength (always 8) and bitOrdering (always BIG_ENDIAN) are ignored on the Qualcomm IoE board.
- On the IoE board, address 0 is supported because there is only one CS pin available. The address number can be passed only via first byte of address parameter of SPIDeviceConfig.
- Chip select polarity can be controlled via 9th bit of address parameter: if this bit is 1, active high polarity will be set, otherwise active low polarity will be set. So in accordance of statement above (only address 0 is supported), there are two possible addresses for IoE board:
  - 0 - address 0, polarity active low
  - 256 address 0, polarity active high.
- The minimal frequency value in Hz is set to 0.
- The deassertion time value is set to 1000 by default.

If you must change any of these properties for some SPI device, you can add the following in the jwc_properties.ini file:

deviceaccess.spi.(bus_id).(slave_address).csMode = {value}
deviceaccess.spi.(bus_id).(slave_address).csPolarity = {value}
deviceaccess.spi.(bus_id).(slave_address).minFreq = {value}
deviceaccess.spi.(bus_id).(slave_address).deassertionTime = {value}
The following UART devices are pre-configured:

<table>
<thead>
<tr>
<th>DAAPI Peripheral ID</th>
<th>DAAPI Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>COM1</td>
<td>DB9 J10 upper port</td>
<td>uartNumber - ignored</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>baudRate = 19200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dataBits = DATABITS_8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>parity = PARITY_NONE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>stopBits = STOPBITS_1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>flowcontrol = FLOWCONTROL_NONE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>inputBufferSize - ignored</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>outputBufferSize - ignored</td>
</tr>
</tbody>
</table>

Please note the following:

- There is only one UART port available from the DAAPIs with ID 40; it has the name COM1.
- uartNumber can only be set to 1, which is the value presented by PeripheralConfig.DEFAULT.
- There is only one event, INPUT_DATA_AVAILABLE, supported on Qualcomm IoE board.
- Only the dataBits values DATABITS_7 and DATABITS_8 are supported.

### Watchdog

The following watchdog devices are pre-configured:

<table>
<thead>
<tr>
<th>DAAPI Peripheral ID</th>
<th>DAAPI Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>WDG</td>
<td>Platform Watchdog</td>
<td></td>
</tr>
</tbody>
</table>
Configuring the Java Runtime Properties

There are several ways to change the value of a property that affects Java's configuration or behavior at runtime.

Direct Modification of the jwc_properties.ini File

The jwc_properties.ini file contains all the properties that affect Java configuration and behavior at runtime. In order to edit this file, do the following:

1. Open the jwc_properties.ini that is a part of the Oracle Java ME Embedded bundle (or download it from the board using the Brew MP SDK Loader tool), find the property that should be changed, and modify its value.
2. Copy the modified version of the jwc_properties.ini file to the /sys/mod/java directory on the Qualcomm IoE board using the Brew MP SDK Loader tool.
3. If there is a jwc_properties.inix file located in this directory, delete it.
4. Restart Java on the Qualcomm IoE board.

Using the CLI Setprop Command

To modify a property using the setprop command in the command-line interface (CLI), do the following.

1. Connect to the board using command-line interface (CLI)
2. Execute the "sysmenu on" command
3. Execute the "setprop <property_name> <desired_property_value>
4. Execute the "saveprops" command
5. Restart Java on the board.

Note, that by executing the "setprop" and "saveprops" commands, the jwc_properties.ini file is always updated automatically.

Using CLI Commands to Alter Network-Related Settings

To alter the network-related settings, do the following:

1. Connect to the board using command-line interface (CLI)
2. Execute the "sysmenu on" command
3. Execute a command that starts with prefix "net" to apply a network-related change.
4. Apply the network-related change and restart Java

## Restarting Java on the Qualcomm IoE Board

You can use any of the following methods to restart Java on the Qualcomm IoE board.

1. Use the CLI "shutdown -r" command. If the "vmconfig.reboot_type" property is set to "soft" (the default), then only Java will be rebooted. However, if the "vmconfig.reboot_type" property is set to "hard" then the board will be rebooted. Note that the "vmconfig.reboot_type" property also affects Device Access API watchdog’s reboot type.

2. Using the CLI "net reconnect" command. This command reconfigures the network and performs a soft Java reboot.

3. Press the "RESET KEY" located on the board, or cycle the power to the board.

For example, here are two methods to change the tooling mode from serial to network. The first method contains the following steps.

1. Open the jwc_properties.ini file and set the "com.oracle.tooling.mode" property to "network".
2. Copy the modified version of the jwc_properties.ini file to the /sys/mod/java directory using the Brew MP SDK Loader application.
3. If there is a jwc_properties.inix file located in this directory, then remove it.
4. Reset the board using the "RESET KEY".

The second method contains the following steps.

1. Connect to the board using command-line interface (CLI).
2. Execute the "sysmenu on" command.
3. Execute a "setprop com.oracle.tooling.mode network" command.
4. Execute the "saveprops" command.
5. Execute a "shutdown -r" command.
Table C–1 lists the error codes that the AMS command-line interface shows when the installation of an IMlet fails. The description of each code contains more information about the problem that caused the error.

### Table C–1  Installer Error Codes

<table>
<thead>
<tr>
<th>Constant</th>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALAA_ALIAS_NOT_FOUND</td>
<td>78</td>
<td>Application Level Access Authorization: The alias definition is missing.</td>
</tr>
<tr>
<td>ALAA_ALIAS_WRONG</td>
<td>80</td>
<td>Application Level Access Authorization: The alias definition is wrong.</td>
</tr>
<tr>
<td>ALAA_MULTIPLE_ALIAS</td>
<td>79</td>
<td>Application Level Access Authorization: An alias has multiple entries that match.</td>
</tr>
<tr>
<td>ALAA_TYPE_WRONG</td>
<td>77</td>
<td>Application Level Access Authorization: The MIDlet-Access-Auth-Type has missing parameters.</td>
</tr>
<tr>
<td>ALREADY_INSTALLED</td>
<td>39</td>
<td>The JAD matches a version of a suite already installed.</td>
</tr>
<tr>
<td>APP_INTEGRITY_FAILURE_DEPENDENCY_CONFLICT</td>
<td>69</td>
<td>Application Integrity Failure: two or more dependencies exist on the component with the same name and vendor, but have different versions or hashes.</td>
</tr>
<tr>
<td>APP_INTEGRITY_FAILURE_DEPENDENCY_MISMATCH</td>
<td>70</td>
<td>Application Integrity Failure: there is a component name or vendor mismatch between the component JAD and IMlet or component JAD that depends on it.</td>
</tr>
<tr>
<td>APP_INTEGRITY_FAILURE_HASH_MISMATCH</td>
<td>68</td>
<td>Application Integrity Failure: hash mismatch.</td>
</tr>
<tr>
<td>ATTRIBUTE_MISMATCH</td>
<td>50</td>
<td>A attribute in both the JAD and JAR manifest does not match.</td>
</tr>
<tr>
<td>AUTHORIZATION_FAILURE</td>
<td>49</td>
<td>Application authorization failure, possibly indicating that the application was not digitally signed.</td>
</tr>
<tr>
<td>CA_DISABLED</td>
<td>60</td>
<td>Indicates that the trusted certificate authority (CA) for this suite has been disabled for software authorization.</td>
</tr>
<tr>
<td>CANCELED</td>
<td>101</td>
<td>Canceled by user.</td>
</tr>
<tr>
<td>CANNOT_AUTH</td>
<td>35</td>
<td>The server does not support basic authentication.</td>
</tr>
<tr>
<td>Constant</td>
<td>Error Code</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CIRCULAR_COMPONENT_DEPENDENCY</td>
<td>64</td>
<td>Circular dynamic component dependency.</td>
</tr>
<tr>
<td>COMPONENT_DEPS_LIMIT_EXCEEDED</td>
<td>65</td>
<td>Dynamic component dependencies limit exceeded.</td>
</tr>
<tr>
<td>COMPONENT_NAMESPACE_COLLISION</td>
<td>72</td>
<td>The namespace used by a component is the same as another.</td>
</tr>
<tr>
<td>CONTENT_HANDLER_CONFLICT</td>
<td>55</td>
<td>The installation of a content handler would conflict with an already installed handler.</td>
</tr>
<tr>
<td>CORRUPT_DEPENDENCY_HASH</td>
<td>71</td>
<td>A dependency has a corrupt hash code.</td>
</tr>
<tr>
<td>CORRUPT_JAR</td>
<td>36</td>
<td>An entry could not be read from the JAR.</td>
</tr>
<tr>
<td>CORRUPT_PROVIDER_CERT</td>
<td>5</td>
<td>The content provider certificate cannot be decoded.</td>
</tr>
<tr>
<td>CORRUPT_SIGNATURE</td>
<td>8</td>
<td>The JAR signature cannot be decoded.</td>
</tr>
<tr>
<td>DEVICE_INCOMPATIBLE</td>
<td>40</td>
<td>The device does not support either the configuration or profile in the JAD.</td>
</tr>
<tr>
<td>DUPLICATED_KEY</td>
<td>88</td>
<td>Duplicated JAD/manifest key attribute</td>
</tr>
<tr>
<td>EXPIRED_CA_KEY</td>
<td>12</td>
<td>The certificate authority's public key has expired.</td>
</tr>
<tr>
<td>EXPIRED_PROVIDER_CERT</td>
<td>11</td>
<td>The content provider certificate has expired.</td>
</tr>
<tr>
<td>INCORRECT_FONT_LOADING</td>
<td>82</td>
<td>A font that is contained with the JAR cannot be loaded.</td>
</tr>
<tr>
<td>INSUFFICIENT_STORAGE</td>
<td>30</td>
<td>Not enough storage for this suite to be installed.</td>
</tr>
<tr>
<td>INVALID_CONTENT_HANDLER</td>
<td>54</td>
<td>The MicroEdition-Handler-&lt;n&gt; JAD attribute has invalid values.</td>
</tr>
<tr>
<td>INVALID_JAD_TYPE</td>
<td>37</td>
<td>The server did not have a resource with the correct type or the JAD downloaded has the wrong media type.</td>
</tr>
<tr>
<td>INVALID_JAD_URL</td>
<td>43</td>
<td>The JAD URL is invalid.</td>
</tr>
<tr>
<td>INVALID_JAR_TYPE</td>
<td>38</td>
<td>The server did not have a resource with the correct type or the JAR downloaded has the wrong media type.</td>
</tr>
<tr>
<td>INVALID_JAR_URL</td>
<td>44</td>
<td>The JAR URL is invalid.</td>
</tr>
<tr>
<td>INVALID_KEY</td>
<td>28</td>
<td>A key for an attribute is not formatted correctly.</td>
</tr>
<tr>
<td>INVALID_NATIVE_LIBRARY</td>
<td>85</td>
<td>A native library contained within the JAR cannot be loaded.</td>
</tr>
<tr>
<td>INVALID_PACKAGING</td>
<td>87</td>
<td>A dependency cannot be satisfied.</td>
</tr>
<tr>
<td>INVALID_PAYMENT_INFO</td>
<td>58</td>
<td>Indicates that the payment information provided with the IMlet suite is incomplete or incorrect.</td>
</tr>
<tr>
<td>INVALID_PROVIDER_CERT</td>
<td>7</td>
<td>The signature of the content provider certificate is invalid.</td>
</tr>
</tbody>
</table>
### Table C–1 (Cont.) Installer Error Codes

<table>
<thead>
<tr>
<th>Constant</th>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_RMS_DATA_TYPE</td>
<td>76</td>
<td>The server did not have a resource with the correct type or the JAD downloaded has the wrong media type.</td>
</tr>
<tr>
<td>INVALID_RMS_DATA_URL</td>
<td>73</td>
<td>The RMS data file URL is invalid.</td>
</tr>
<tr>
<td>INVALID_SERVICE_EXPORT</td>
<td>86</td>
<td>A LIBlet that exports a service with a LIBlet Services attribute does not contain the matching service provider configuration information.</td>
</tr>
<tr>
<td>INVALID_SIGNATURE</td>
<td>9</td>
<td>The signature of the JAR is invalid.</td>
</tr>
<tr>
<td>INVALID_VALUE</td>
<td>29</td>
<td>A value for an attribute is not formatted correctly.</td>
</tr>
<tr>
<td>INVALID_VERSION</td>
<td>16</td>
<td>The format of the version is invalid.</td>
</tr>
<tr>
<td>IO_ERROR</td>
<td>102</td>
<td>A low-level hardware error has occurred.</td>
</tr>
<tr>
<td>JAD_MOVED</td>
<td>34</td>
<td>The JAD URL for an installed suite is different than the original JAD URL.</td>
</tr>
<tr>
<td>JAD_NOT_FOUND</td>
<td>2</td>
<td>The JAD was not found.</td>
</tr>
<tr>
<td>JAD_SERVER_NOT_FOUND</td>
<td>1</td>
<td>The server for the JAD was not found.</td>
</tr>
<tr>
<td>JAR_CLASSES_VERIFICATION_FAILED</td>
<td>56</td>
<td>Not all classes within JAR package can be successfully verified with class verifer.</td>
</tr>
<tr>
<td>JAR_IS_LOCKED</td>
<td>100</td>
<td>Component or MIDlet or IMlet suite is locked by the system.</td>
</tr>
<tr>
<td>JAR_NOT_FOUND</td>
<td>20</td>
<td>The JAR was not found at the URL given in the JAD.</td>
</tr>
<tr>
<td>JAR_SERVER_NOT_FOUND</td>
<td>19</td>
<td>The server for the JAR was not found at the URL given in the JAD.</td>
</tr>
<tr>
<td>JAR_SIZE_MISMATCH</td>
<td>31</td>
<td>The JAR downloaded was not the same size as given in the JAD.</td>
</tr>
<tr>
<td>MISSING_CONFIGURATION</td>
<td>41</td>
<td>The configuration is missing from the manifest.</td>
</tr>
<tr>
<td>MISSING_DEPENDENCY_HASH</td>
<td>67</td>
<td>A dependency hash code is missing.</td>
</tr>
<tr>
<td>MISSING_DEPENDENCY_JAD_URL</td>
<td>66</td>
<td>A dependency JAD URL is missing.</td>
</tr>
<tr>
<td>MISSING_JAR_SIZE</td>
<td>21</td>
<td>The JAR size is missing.</td>
</tr>
<tr>
<td>MISSING_JAR_URL</td>
<td>18</td>
<td>The URL for the JAR is missing.</td>
</tr>
<tr>
<td>MISSING_PROFILE</td>
<td>42</td>
<td>The profile is missing from the manifest.</td>
</tr>
<tr>
<td>MISSING_PROVIDER_CERT</td>
<td>4</td>
<td>The content provider certificate is missing.</td>
</tr>
<tr>
<td>MISSING_SUITE_NAME</td>
<td>13</td>
<td>The name of MIDlet or IMlet suite is missing.</td>
</tr>
<tr>
<td>MISSING_VENDOR</td>
<td>14</td>
<td>The vendor is missing.</td>
</tr>
<tr>
<td>MISSING_VERSION</td>
<td>15</td>
<td>The version is missing.</td>
</tr>
<tr>
<td>NEW_VERSION</td>
<td>32</td>
<td>This suite is newer that the one currently installed.</td>
</tr>
</tbody>
</table>
### Table C–1 (Cont.) Installer Error Codes

<table>
<thead>
<tr>
<th>Constant</th>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO_ERROR</td>
<td>0</td>
<td>No error.</td>
</tr>
<tr>
<td>NOT_YET_VALID_PROVIDER_CERT</td>
<td>89</td>
<td>A certificate is not yet valid.</td>
</tr>
<tr>
<td>NOT_YET_VALID_CA_KEY</td>
<td>90</td>
<td>A CA’s public key is not yet valid.</td>
</tr>
<tr>
<td>OLD_VERSION</td>
<td>17</td>
<td>This suite is older than the one currently installed.</td>
</tr>
<tr>
<td>OTHER_ERROR</td>
<td>103</td>
<td>Other errors.</td>
</tr>
<tr>
<td>PROXY_AUTH</td>
<td>51</td>
<td>Indicates that the user must first authenticate with the proxy.</td>
</tr>
<tr>
<td>PUSH_CLASS_FAILURE</td>
<td>48</td>
<td>The class in a push attribute is not in MIDlet-&lt;n&gt; attribute.</td>
</tr>
<tr>
<td>PUSH_DUP_FAILURE</td>
<td>45</td>
<td>The connection in a push entry is already taken.</td>
</tr>
<tr>
<td>PUSH_FORMAT_FAILURE</td>
<td>46</td>
<td>The format of a push attribute has an invalid format.</td>
</tr>
<tr>
<td>PUSH_PROTO_FAILURE</td>
<td>47</td>
<td>The connection in a push attribute is not supported.</td>
</tr>
<tr>
<td>REVOKED_CERT</td>
<td>62</td>
<td>The certificate has been revoked.</td>
</tr>
<tr>
<td>RMS_DATA_DECRYPT_PASSWORD</td>
<td>83</td>
<td>Indicates that a password is required to decrypt RMS data.</td>
</tr>
<tr>
<td>RMS_DATA_ENCRYPT_PASSWORD</td>
<td>84</td>
<td>Indicates that a password is required to encrypt RMS data.</td>
</tr>
<tr>
<td>RMS_DATA_NOT_FOUND</td>
<td>75</td>
<td>The RMS data file was not found at the specified URL.</td>
</tr>
<tr>
<td>RMS_DATA_SERVER_NOT_FOUND</td>
<td>74</td>
<td>The server for the RMS data file was not found at the specified URL.</td>
</tr>
<tr>
<td>RMS_INITIALIZATION_FAILURE</td>
<td>81</td>
<td>Failure to import RMS data.</td>
</tr>
<tr>
<td>SUITE_NAME_MISMATCH</td>
<td>25</td>
<td>The MIDlet or IMlet suite name does not match the one in the JAR manifest.</td>
</tr>
<tr>
<td>TOO_MANY_PROPS</td>
<td>53</td>
<td>Indicates that either the JAD or manifest has too many properties to fit into memory.</td>
</tr>
<tr>
<td>TRUSTED_OVERWRITE_FAILURE</td>
<td>52</td>
<td>Indicates that the user tried to overwrite a trusted suite with an untrusted suite during an update.</td>
</tr>
<tr>
<td>UNAUTHORIZED</td>
<td>33</td>
<td>Web server authentication failed or is required.</td>
</tr>
<tr>
<td>UNKNOWN_CA</td>
<td>6</td>
<td>The certificate authority (CA) that issued the content provider certificate is unknown.</td>
</tr>
<tr>
<td>UNKNOWN_CERT_STATUS</td>
<td>63</td>
<td>The certificate is unknown to OCSP server.</td>
</tr>
<tr>
<td>UNSUPPORTED_CERT</td>
<td>10</td>
<td>The content provider certificate has an unsupported version.</td>
</tr>
<tr>
<td>UNSUPPORTED_CHAR_ENCODING</td>
<td>61</td>
<td>Indicates that the character encoding specified in the MIME type is not supported.</td>
</tr>
</tbody>
</table>
### Table C–1  (Cont.) Installer Error Codes

<table>
<thead>
<tr>
<th>Constant</th>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNSUPPORTED_PAYMENT_INFO</td>
<td>57</td>
<td>Indicates that the payment information provided with the MIDlet or IMlet suite is incompatible with the current implementation.</td>
</tr>
<tr>
<td>UNTRUSTED_PAYMENT_SUITE</td>
<td>59</td>
<td>Indicates that the MIDlet or IMlet suite has payment provisioning information but it is not trusted.</td>
</tr>
<tr>
<td>VENDOR_MISMATCH</td>
<td>27</td>
<td>The vendor does not match the one in the JAR manifest.</td>
</tr>
<tr>
<td>VERSION_MISMATCH</td>
<td>26</td>
<td>The version does not match the one in the JAR manifest.</td>
</tr>
</tbody>
</table>
**Access Point**
A network-connectivity configuration that is predefined on a device. An access point can represent different network profiles for the same bearer type, or for different bearer types that may be available on a device, such as WiFi or bluetooth.

**ADC**
Analog-to-Digital Converter. A hardware device that converts analog signals (time and amplitude) into a stream of binary numbers that can be processed by a digital device.

**AMS**
Application Management System. The system functionality that completes tasks such as installing applications, updating applications, and managing applications between foreground and background.

**APDU**
Application Protocol Data Unit. A communication mechanism used by SIM Cards and smart cards to communicate with card reader software or a card reader device.

**API**
Application Programming Interface. A set of classes used by programmers to write applications that provide standard methods and interfaces and eliminate the need for programmers to reinvent commonly used code.

**ARM**
Advanced RISC Machine. A family of computer processors using reduced instruction set (RISC) CPU technology, developed by ARM Holdings. ARM is a licensable instruction set architecture (ISA) and is used in the majority of embedded platforms.

**AT commands**
A set of commands developed to facilitate modem communications, such as dialing, hanging up, and changing the parameters of a connection. Also known as the Hayes command set, AT means attention.

**AXF**
ARM Executable Format. An ARM executable image generated by ARM tools.

**BIP**
Bearer Independent Protocol. Allows an application on a SIM Card to establish a data channel with a terminal, and through the terminal, to a remote server on the network.
CDMA
Code Division Multiple Access. A mobile telephone network standard used primarily in the United States and Canada as an alternative to GSM.

CLDC
Connected Limited Device Configuration. A Java ME platform configuration for devices with limited memory and network connectivity. It uses a low-footprint Java virtual machine such as the CLDC HotSpot Implementation, and several minimalist Java platform APIs for application services.

Configuration
Defines the minimum Java runtime environment (for example, the combination of a Java virtual machine and a core set of Java platform APIs) for a family of Java ME platform devices.

DAC
Digital-to-Analog Converter. A hardware device that converts a stream of binary numbers into an analog signal (time and amplitude), such as audio playback.

ETSI
European Telecommunications Standards Institute. An independent, non-profit group responsible for the standardization of information and communication technologies within Europe. Although based in Europe, it carries worldwide influence in the telecommunications industry.

GCF
Generic Connection Framework. A part of CLDC, it is a Java ME API consisting of a hierarchy of interfaces and classes to create connections (such as HTTP, datagram, or streams) and perform I/O.

GPIO
General Purpose Input/Output. Unassigned pins on an embedded platform that can be assigned or configured as needed by a developer.

GPIO Port
A group of GPIO pins (typically 8 pins) arranged in a group and treated as a single port.

GSM
Global System for Mobile Communications. A 3G mobile telephone network standard used widely in Europe, Asia, and other parts of the world.

HTTP
HyperText Transfer Protocol. The most commonly used Internet protocol, based on TCP/IP that is used to fetch documents and other hypertext objects from remote hosts.

HTTPS

ICCID
Integrated Circuit Card Identification. The unique serial number assigned to an individual SIM Card.
IMP-NG
Information Module Profile Next Generation. A profile for embedded “headless” devices, the IMP-NG specification (JSR 228) is a subset of MIDP 2.0 that leverages many of the APIs of MIDP 2.0, including the latest security and networking+, but does not include graphics and user interface APIs.

IMEI
International Mobile Equipment Identifier. A number unique to every mobile phone. It is used by a GSM or UMTS network to identify valid devices and can be used to stop a stolen or blocked phone from accessing the network. It is usually printed inside the battery compartment of the phone.

IMlet
An application written for IMP-NG. An IMlet does not differ from MIDP 2.0 MIDlet, except by the fact that an IMlet can not refer to MIDP classes that are not part of IMP-NG. An IMlet can only use the APIs defined by the IMP-NG and CLDC specifications.

IMlet Suite
A way of packaging one or more IMlets for easy distribution and use. Similar to a MIDlet suite, but for smaller applications running in an embedded environment.

IMSI
International Mobile Subscriber Identity. A unique number associated with all GSM and UMTS network mobile phone users. It is stored on the SIM Card inside a phone and is used to identify itself to the network.

I2C
Inter-Integrated Circuit. A multi-master, serial computer bus used to attach low-speed peripherals to an embedded platform.

ISA
Instruction Set Architecture. The part of a computer’s architecture related to programming, including data type, addressing modes, interrupt and exception handling, I/O, and memory architecture, and native commands. Reduced instruction set computing (RISC) is one kind of instruction set architecture.

JAD file
Java Application Descriptor file. A file provided in a MIDlet or IMlet suite that contains attributes used by application management software (AMS) to manage the MIDlet or IMlet life cycle, and other application-specific attributes used by the MIDlet or IMlet suite itself.

JAR file
Java Archive file. A platform-independent file format that aggregates many files into one. Multiple applications written in the Java programming language and their required components (class files, images, sounds, and other resource files) can be bundled in a JAR file and provided as part of a MIDlet or IMlet suite.

JCP
Java Community Process. The global standards body guiding the development of the Java programming language.
**JDTS**

Java Device Test Suite. A set of Java programming language tests developed specifically for the wireless marketplace, providing targeted, standardized testing for CLDC and MIDP on small and handheld devices.

**Java ME platform**

Java Platform, Micro Edition. A group of specifications and technologies that pertain to running the Java platform on small devices, such as cell phones, pagers, set-top boxes, and embedded devices. More specifically, the Java ME platform consists of a configuration (such as CLDC) and a profile (such as MIDP or IMP-NG) tailored to a specific class of device.

**JSR**

Java Specification Request. A proposal for developing new Java platform technology, which is reviewed, developed, and finalized into a formal specification by the JCP program.

**Java Virtual Machine**

A software “execution engine” that safely and compatibly executes the byte codes in Java class files on a microprocessor.

**KVM**

A Java virtual machine designed to run in a small, limited memory device. The CLDC configuration was initially designed to run in a KVM.

**LCDUI**

Liquid Crystal Display User Interface. A user interface toolkit for interacting with Liquid Crystal Display (LCD) screens in small devices. More generally, a shorthand way of referring to the MIDP user interface APIs.

**MIDlet**

An application written for MIDP.

**MIDlet suite**

A way of packaging one or more MIDlets for easy distribution and use. Each MIDlet suite contains a Java application descriptor file (.jad), which lists the class names and files names for each MIDlet, and a Java Archive file (.jar), which contains the class files and resource files for each MIDlet.

**MIDP**

Mobile Information Device Profile. A specification for a Java ME platform profile, running on top of a CLDC configuration that provides APIs for application life cycle, user interface, networking, and persistent storage in small devices.

**MSISDN**

Mobile Station Integrated Services Digital Network. A number uniquely identifying a subscription in a GSM or UMTS mobile network. It is the telephone number to the SIM Card in a mobile phone and used for voice, FAX, SMS, and data services.

**MVM**

Multiple Virtual Machines. A software mode that can run more than one MIDlet or IMlet at a time.
Obfuscation
A technique used to complicate code by making it harder to understand when it is decompiled. Obfuscation makes it harder to reverse-engineer applications and therefore, steal them.

Optional Package
A set of Java ME platform APIs that provides additional functionality by extending the runtime capabilities of an existing configuration and profile.

Preemption
Taking a resource, such as the foreground, from another application.

Preverification
Due to limited memory and processing power on small devices, the process of verifying Java technology classes is split into two parts. The first part is preverification which is done off-device using the preverify tool. The second part, which is verification, occurs on the device at runtime.

Profile
A set of APIs added to a configuration to support specific uses of an embedded or mobile device. Along with its underlying configuration, a profile defines a complete and self-contained application environment.

Provisioning
A mechanism for providing services, data, or both to an embedded or mobile device over a network.

Pulse Counter
A hardware or software component that counts electronic pulses, or events, on a digital input line, for example, a GPIO pin.

Push Registry
The list of inbound connections, across which entities can push data. Each item in the list contains the URL (protocol, host, and port) for the connection, the entity permitted to push data through the connection, and the application that receives the connection.

RISC
Reduced Instruction Set Computing. A CPU design based on simplified instruction sets that provide higher performance and faster execution of individual instructions. The ARM architecture is based on RISC design principles.

RL-ARM
Real-Time Library. A group of tightly coupled libraries designed to solve the real-time and communication challenges of embedded systems based on ARM processor-based microcontroller devices.

RMI
Remote Method Invocation. A feature of Java SE technology that enables Java technology objects running in one virtual machine to seamlessly invoke objects running in another virtual machine.
RMS
Record Management System. A simple record-oriented database that enables an IMlet or MIDlet to persistently store information and retrieve it later. MIDlets can also use the RMS to share data.

RTOS
Real-Time Operating System. An operating system designed to serve real-time application requests. It uses multi-tasking, an advanced scheduling algorithm, and minimal latency to prioritize and process data.

RTSP
Real Time Streaming Protocol. A network control protocol designed to control streaming media servers and media sessions.

SCWS
Smart Card Web Server. A web server embedded in a smart card (such as a SIM Card) that allows HTTP transactions with the card.

SD card
Secure Digital cards. A non-volatile memory card format for use in portable devices, such as mobile phones and digital cameras, and embedded systems. SD cards come in three different sizes, with several storage capacities and speeds.

SIM
Subscriber Identity Module. An integrated circuit embedded into a removable SIM card that securely stores the International Mobile Subscriber Identity (IMSI) and the related key used to identify and authenticate subscribers on mobile and embedded devices.

Slave Mode
Describes the relationship between a master and one or more devices in a Serial Peripheral Interface (SPI) bus arrangement. Data transmission in an SPI bus is initiated by the master device and received by one or more slave devices, which cannot initiate data transmissions on their own.

Smart Card
A card that stores and processes information through the electronic circuits embedded in silicon in the substrate of its body. Smart cards carry both processing power and information. A SIM Card is a special kind of smart card for use in a mobile device.

SMS
Short Message Service. A protocol allowing transmission of short text-based messages over a wireless network. SMS messaging is the most widely-used data application in the world.

SMSC
Short Message Service Center. The SMSC routes messages and regulates SMS traffic. When an SMS message is sent, it goes to an SMS center first, then gets forwarded to the destination. If the destination is unavailable (for example, the recipient embedded board is powered down), the message is stored in the SMSC until the recipient becomes available.
SOAP
Simple Object Access Protocol. An XML-based protocol that enables objects of any type to communicate in a distributed environment. It is most commonly used to develop web services.

SPI
Serial Peripheral Interface. A synchronous bus commonly used in embedded systems that allows full-duplex communication between a master device and one or more slave devices.

SSL
Secure Sockets Layer. A protocol for transmitting data over the Internet using encryption and authentication, including the use of digital certificates and both public and private keys.

SVM
Single Virtual Machine. A software mode that can run only one MIDlet or IMlet at a time.

Task
At the platform level, each separate application that runs within a single Java virtual machine is called a task. The API used to instantiate each task is a stripped-down version of the Isolate API defined in JSR 121.

TCP/IP
Transmission Control Protocol/Internet Protocol. A fundamental Internet protocol that provides for reliable delivery of streams of data from one host to another.

Terminal Profile
Device characteristics of a terminal (mobile or embedded device) passed to the SIM Card along with the IMEI at SIM Card initialization. The terminal profile tells the SIM Card what values are supported by the device.

UART
Universal Asynchronous Receiver/Transmitter. A piece of computer hardware that translates data between serial and parallel formats. It is used to facilitate communication between different kinds of peripheral devices, input/output streams, and embedded systems, to ensure universal communication between devices.

UICC
Universal Integrated Circuit Card. The smart card used in mobile terminals in GSM and UMTS networks. The UICC ensures the integrity and security of personal data on the card.

UMTS
Universal Mobile Telecommunications System. A third-generation (3G) mobile communications technology. It utilizes the radio spectrum in a fundamentally different way than GSM.

URI
Uniform Resource Identifier. A compact string of characters used to identify or name an abstract or physical resource. A URI can be further classified as a uniform resource locator (URL), a uniform resource name (URN), or both.
USAT
Universal SIM Application Toolkit. A software development kit intended for 3G networks. It enables USIM to initiate actions that can be used for various value-added services, such as those required for banking and other privacy related applications.

USB
Universal Serial Bus. An industry standard that defines the cables, connectors, and protocols used in a bus for connection, communication, and power supply between computers and electronic devices, such as embedded platforms and mobile phones.

USIM
Universal Subscriber Identity Module. An updated version of a SIM designed for use over 3G networks. USIM is able to process small applications securely using better cryptographic authentication and stronger keys. Larger memory on USIM enables the addition of thousands of contact details including subscriber information, contact details, and other custom settings.

WAE
Wireless Application Environment. An application framework for small devices, which leverages other technologies, such as Wireless Application Protocol (WAP).

WAP
Wireless Application Protocol. A protocol for transmitting data between a server and a client (such as a cell phone or embedded device) over a wireless network. WAP in the wireless world is analogous to HTTP in the World Wide Web.

Watchdog Timer
A dedicated piece of hardware or software that "watches" an embedded system for a fault condition by continually polling for a response. If the system goes offline and no response is received, the watchdog timer initiates a reboot procedure or takes other steps to return the system to a running state.

WCDMA
Wideband Code Division Multiple Access. A detailed protocol that defines how a mobile phone communicates with the tower, how its signals are modulated, how datagrams are structured, and how system interfaces are specified.

WMA
Wireless Messaging API. A set of classes for sending and receiving Short Message Service (SMS) messages.

XML Schema
A set of rules to which an XML document must conform to be considered valid.
## Index

### A
- Application Management System (AMS)
  - Commands, 2-6

### B
- Brew MP SDK
  - Installation, 1-2
  - Logger, 2-4

### C
- Command Line Interface, 2-1, 2-4
- com.oracle.tooling.mode, 2-2

### E
- Eclipse, 3-6
  - Assigning the Qualcomm IoE Board to a project, 3-8
  - Debugging, 3-9
  - Device Selector, 3-7
  - Oracle Java ME SDK 3.4 Plugin, 3-6
  - Signing an Application with API Permissions, 3-11

### J
- Java ME SDK 3.4, 2-2
  - Device Manager, 2-2
  - Manage Device Addresses, 2-2

### L
- Loader Application, 1-2, 1-3
- Logger Application, 1-2, 1-4
- Logging Interface, 2-1, 2-3

### N
- NetBeans, 3-1
  - Accessing Peripherals, 3-10
  - Assigning a Board to the Project, 3-3
  - Debugging, 3-5
  - Device Selector, 3-2
  - Java ME SDK 3.4 Plugin, 3-1

### O
- Signing an Application with API Permissions, 3-10
- On-Device Tooling, 2-1
- Oracle Java ME SDK 3.4
  - Device Manager, 2-2

### P
- PuTTY
  - Installation, 2-3

### Q
- Qualcomm IoE
  - Copying Files to the Board, 1-3
  - /sys/mod/java directory, 1-4
  - USB Drivers, 1-2
- Qualcomm IoE Board
  - Setting Up, 1-1
- Qualcomm IoE User's Guide, 1-1

### S
- Security Policy File, 3-13

### T
- TCP-to-Serial Fallback, 2-10
- Tooling
  - Over Network, 2-1
  - Over Networking, 2-8
  - Over Serial, 2-1
tooling, 2-1

### W
- WiFi Networking, 2-9
- Windows Device Manager, 1-2