Java Development Tools – Overview and Installation

Overview

This document is intended to help with installation and initial use of the Java development tools for Cinterion Java-powered wireless modules from Gemalto.

The devices which support Java applications are the BGS5, EHS5, EHS6, EHS8, PDS5, PDS6 & PDS8 modules and the BGS5T & EHS6T Terminals with a Java-powered module inside.

If you have previously used Java with the TC65i or EGS5 modules or TC65T Terminals there is a useful “differences” paragraph at the end of the Java User Guide which helps with migration to the new family of devices.

The software installation and initial development process below references the EHS6T Terminal and EHS6 Concept Board as a target hardware platforms. Whilst these ready-to-use devices are highly convenient requiring no engineering effort, the development tools installation process is the same for all devices. Note that if you are designing your own embedded hardware platform, the development tools are most effective when the USB interface is used between the PC and your embedded device.

Getting Started

Once you have a Java ME powered Terminal or EHS6 Concept Board it’s a simple process to register the device IMEI and gain access to the Gemalto tools & support download page. The installation package is a single download containing all the required software components and development environment integrations, available at http://www.concept-board.com/
Note that the BGS5 and BGS5T have different installation packages to the other devices and the ZIP file download may need to be requested separately.

There is a choice of using either Eclipse or NetBeans development environments. The ‘walk-through’ and examples below uses NetBeans, however either can be used and both are covered in official installation notes.

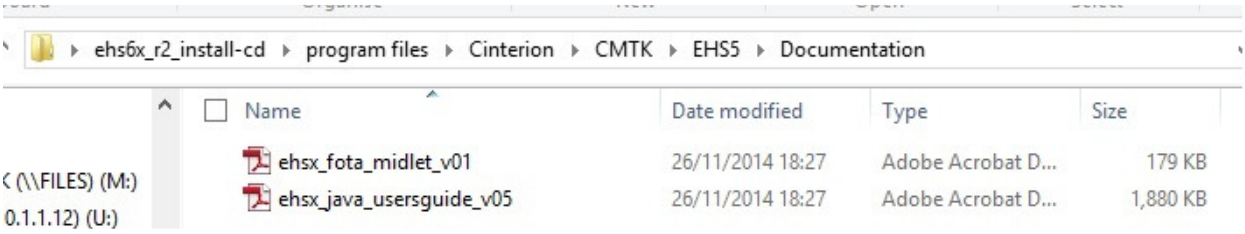
Note that installation is supported for Windows XP, Windows Vista or Windows 7.

A summary of the Java SDK installation process follows.

To start, the installation package needs to be extracted from the downloaded ZIP file. The installation process will create suitable directories and place files logically, regardless of where the package is unzipped too. Note however that the installation creates some long path names and if not extracted to a relatively high level directory, the installer may find an error during the install process.

For the walk-through below there were no existing versions of any of the applications and the package was unzipped to the Desktop to avoid extended path names and for ease of viewing during the process.

If you don't already have access to the Java User Guide then after extracting the install package it can be found in the documentation directory:

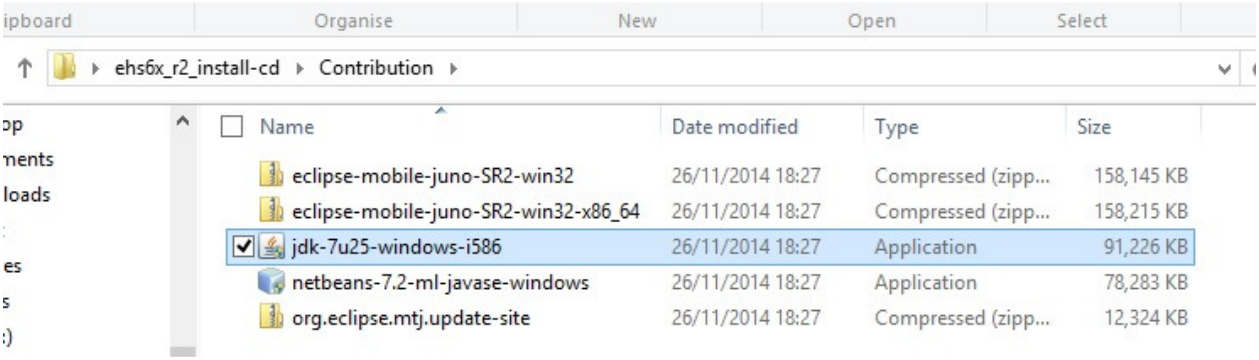


The Java User Guide contains details of specific aspects of the installation process and further information to extend your deployment and ongoing product management capabilities beyond this overview guide.

The system requirements are covered in Section 3.1.

Installation

1. Install the Java Development Kit.



The installer application provided by Oracle is for a recent Java Development Kit. The installer can be found in the /Contribution directory and will be listed as 'jdk-7u25-windows-i586' or something similar if a later version is included and is ready to run. The Wizard will guide you through all aspects automatically. At each stage it is possible to accept the default suggested option(s).

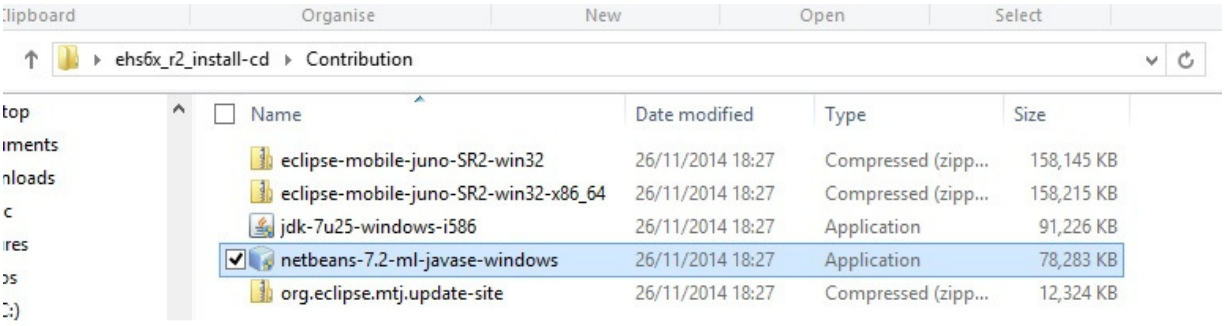
2. Install NetBeans 7.2 Integrated Development Environment

The Java Development Kit requires Integrated Development Environment (IDE) on which to function.

NetBeans can be installed from the same directory as above. Note the recommendation in the Java User Guide section 10.3 which relates to this and the next stage of the install. "It is recommended to use NetBeans 7.2. On later or earlier versions the version of the Java client implemented in the module might not be supported properly."

If you later upgrade the NetBeans IDE and find the development process fails to work correctly it may be related to this comment.

Run the NetBeans installer application 'netbeans-7.2-ml-javase-windows', accepting the licences and suggested install directories.



3. Install NetBeans Mobility plug-in

When the NetBeans IDE has completed installation, an icon will appear on the Desktop. This is used to start the IDE.

The NetBeans IDE requires a 'plug-in' to be installed so that it functions correctly in the SDK for Gemalto wireless modules.

Open the NetBeans IDE and follow the instructions in the Java User Guide section 10.3 for Installing the “Mobility” Plugin using the plugin manager. At the same time it is possible to install the Java ME SDK tools which optionally provide performance analysis of the CPU while executing applications.

Close the NetBeans IDE ready to install the rest of the tools and integrations.

4. Install the Cinterion Mobility Toolkit (CMTK)

Before installing the CMTK it is worth assembling the hardware which will be used for Java development (Concept Board / Modem Terminal). During the later stage of this section the installer is able to detect and correctly hook-in the device COM port to the IDE. Don’t worry if this is not possible at this stage though, the process can easily be re-initiated or repeated later on.

The CMTK installer is found in the route directory and called “setup”. Start the installer and follow the guidelines as per section 3.3.2 of the Java User Manual ‘Installing CMTK’, accepting the prompts as indicated.

The installer will show prompt prior to installing the Gemalto Module Exchange Suite (stage 4 of the Java User Guide). This separate utility allows a modules’ a:\ drive to be accessible within windows explorer. Once installed & with a module connected it is possible to drag and drop files straight over to the modules’ flash memory, ready for installing into the Java environment. Accept the prompt to install MES. Using MES is covered below.

Continue with the installation. A prompt will appear before asking to scan the PC COM ports for a Java module (stage 9 in the Java User Guide). If the device intended for software development is available it should be connected and powered/started now. Setup will have installed USB drivers for BGS5/EHSx devices and once powered the device should enumerate into the PC (see Windows Hardware Manager / Devices and Printers). Once confirmed continue with the installation to completion.

Creating a new application (Java Midlet)

Whenever starting a new MIDlet there are a couple of ‘housekeeping’ tasks and checks that it is wise to initially perform.

Note that if you wish to use any of the examples provided with the package, it is always better to create a new project and paste code into the MIDlet. A newly

created project should have the all the related links correctly defined for your installation.

Creating a new project and MIDlet is detailed in section 10.3.4 which should be followed through, including adding the ATCommand Class, to complete the initial configuration ready for the first and subsequent new projects.

Stage 3 allows selection of the target device for debugging the Java application. If “IMP_NG_EHS5_**REMOTE 1**” is selected debugging will be done on the connected hardware platform.

You are now ready to start writing code!

Running and De-bugging applications

Once a MIDlet is complete the project can be built using the IDE (Menu => Run => Clean and Build Project) and then run (Menu => Run => Run Project).

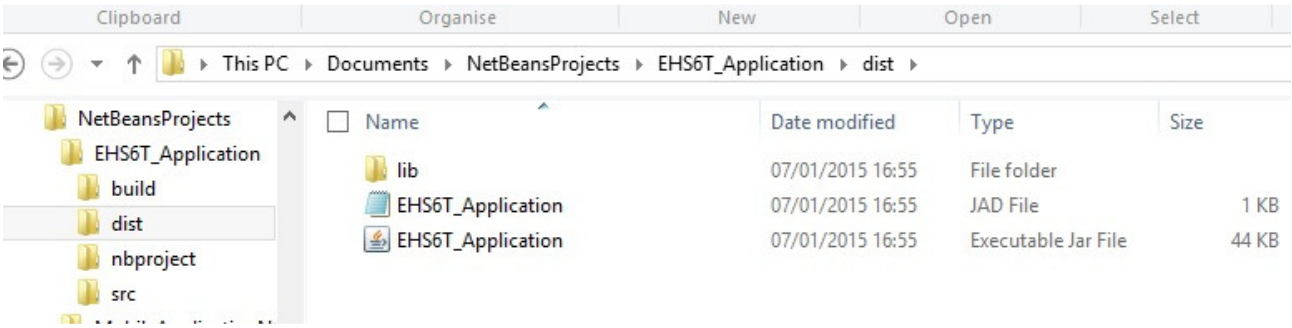
An IP connection will be created for the debug connection to a connected device. Intervention may be required to grant access for data to pass through any firewall or security present on the PC. The IDE will wait for the connection to become available.

As well as a progress bar, an output window should be visible showing the IDE process and the Stdout stream.

Running Java applications on a device without the IDE.

Once a working application is complete it can be installed and made to run on one of the Java powered devices.

The Java application consists of two files which are both required for the application to run. Both need to be copied to the module using the Module Exchange Suite which was installed as part of the CMTK installation. Further information regarding MES can be found in the Java User Guide section 7.1. The two files will be found in the ‘dist’ (distribution) sub-directory of project directory.



Running an application manually (i.e. by issuing an AT command) is covered in Section 9.3 of the Java User Manual.

Remember that after copying over the JAD & JAR files to the modules' a:\ drive they need installing into the Java environment before the application can be run using the AT command. See *Example* for the two AT commands in section 9.3

It will be necessary to connect a Terminal Emulator to one of the available AT Command interface COM ports to perform the AT command operation.
The list of installed MIDlets can be checked using the command `AT^SJAM=4`
Note there should always be an additional MIDlet installed in the device which is provided by Gemalto. This JRC "Java Remote Control" is required by the system and should never be deleted. This also relies upon the global Userware/Autostart being enabled – do not change this to disabled using the AT command.

Running an application with Autostart (i.e. without user intervention after the device has powered up) is covered in Section 9.4 of the Java User Manual.

Three additional properties need to be added to the JAD file for autostart to function correctly. These are the three entries together with a valid value, which are shown following each of the dashes/minus-signs in section 9.4

So a modified JAD file might look like this:

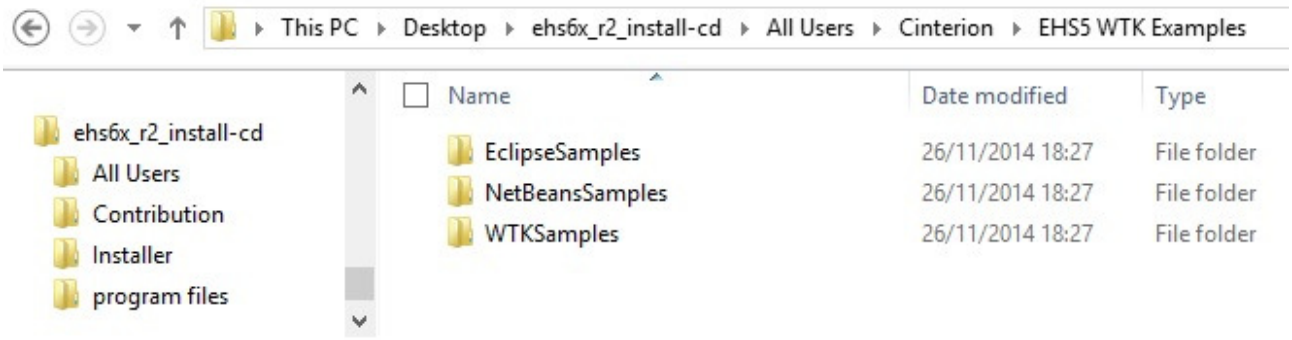
```
MIDlet-1: MobileApplication,,Fred
MIDlet-Jar-Size: 46268
MIDlet-Jar-URL: MobileApplication.jar
MIDlet-Name: MobileApplication
MIDlet-Vendor: Vendor
MIDlet-Version: 1.0
Oracle-MIDlet-Autostart: 1
Oracle-MIDlet-Restart: false
Oracle-MIDlet-Restart-Count: 1
MicroEdition-Configuration: CLDC-1.1
MicroEdition-Profile: IMP-NG
```

Once the modified JAD file is copied to a device and installed into the Java environment the MIDlet should automatically start after a device powering up.

Hints and Tips

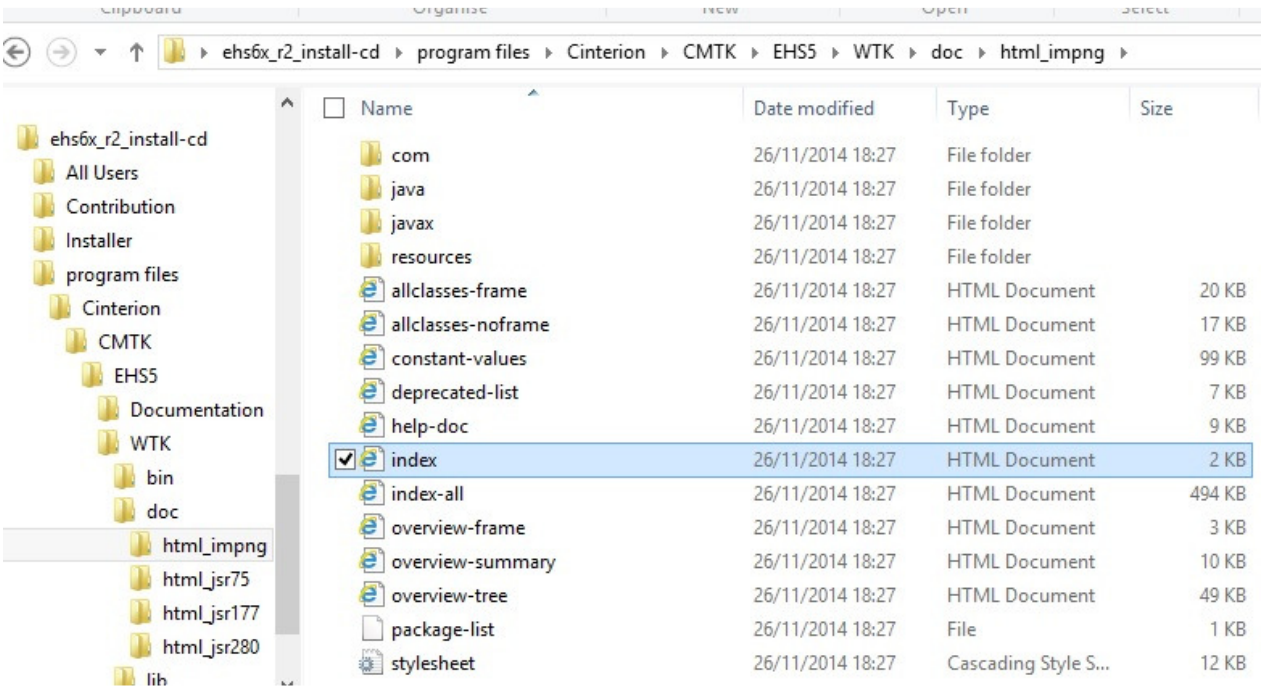
Examples

Some example Java MIDlets are included in the download package and can be found in the following directory.



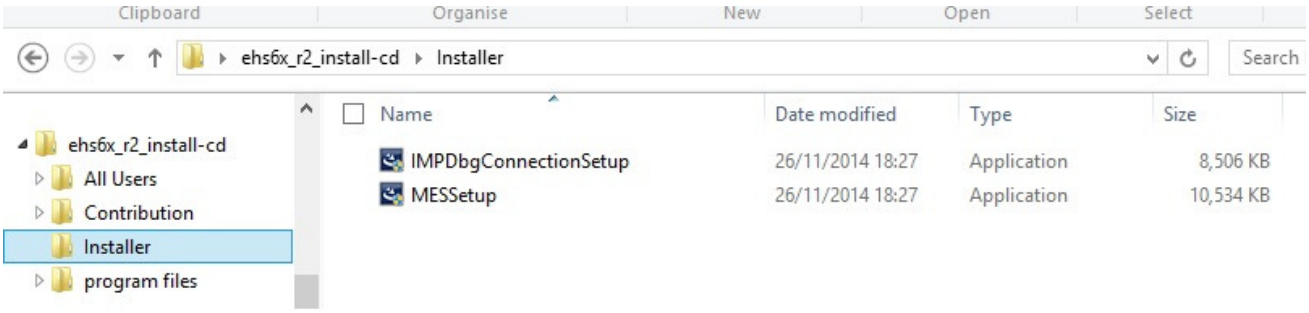
Documentation

An http based documentation resource is included in the following directory



PC COM Ports

If the USB Virtual COM port numbers (assigned by the PC when a device is connected) change this can invalidate the IDE associations. The IDE won't be able to connect with new debug and Stdout port using the out of date references. Running the "IMPDbgConnectionSetup" application which can be located in the "/Installer" directory will scan and restore the necessary associations.



If the hardware platform is changed e.g. from the Concept Board to a Terminal, it will be necessary to run this application for the new device.

Assigning Stdout

The Stdout path on the Java Powered device can be manually assigned using the command AT^SCFG="Userware/Stdout" with the relevant parameters defined as required. The new output path must be currently available when the command is issued (i.e if only the ASC0 / Serial connection is being used it is not possible to assign Stdout to a USB device port.

See the relative AT Command Guide for the device being used for development.