How to conform with Unified Testing Criteria

How to conform with Unified Testing Criteria for Java(TM) Technology-based Applications for Mobile Devices

Overview

The purpose of this wiki page is helping you to develop applications that conform Java Verified testing criteria. Most of the tests are to ensure that applications are mature enough to be published for wider audience. Applications must be easy to use and robust.

Please read the original UTC document before sending your application for signing. If there is a conflict between this page and unified testing criteria trust the latter one.

Application Characteristics (AC)

Information about the application is provided to help the test houses in the testing work.

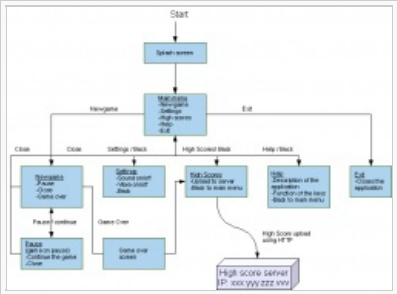
AC1 Application flow

The information in the flow of the application must be provided according to the application specification.

Application flow chart should cover whole application and it must be easy to read.

At least following information must be shown:

- Name of each section
- Description of each section
- How to get to each section
- Where to go from each section
- Where passwords are used



AC1 Flow diagram

AC2 Application information

The application characteristics document must be properly filled.

Fill in the "Application specific questions" form on Java Verified submission portal with care.

Stability (ST)

Focusing on the application being stable on the device.

ST1 Application stability

Must not crash during testing

Application stability is one of the key factors in user experience and it is observed during the testing process. Be sure to test all the functionalities before sending app for testing. In case it is a game play it trough, and ensure that it will not crash on end credits.

If your application uses connectivity it is worthwhile to pay extra attention on writing robust code, because error situations are also tested (CO1-CO13)

ST2 Power Consumption (Observation Only)

This is not a requirement but an observation, i.e. the result of this test does not have an effect on the overall pass/fail verdict, but will be documented.

Refer to the optimization links on FN13. When application is optimized, it usually consumes less power.

On S60 devices, you can use Nokia Energy Profiler to monitor power consumption

Application Launch (AL)

(C) Copyright Nokia 2014. All rights reserved.

Once an application is loaded it must start (launch) and stop correctly in relation to the device and other applications on the device.

AL1 Application Installation

The application must install via OTA (over-the-air).

Java over-the-air installation is very common way to install MIDlet suite. OTA installation requires that JAD and JAR attributes are set correctly.

Some Nokia devices, especially Series 40 devices, have limit for maximum OTA file size. When transferring files over the air it is always good to keep file sizes small.

- How to reduce JAR size
- Series 40: Considerations regarding MIDlet startup and runtime execution

If you want to test OTA install with your own web server, appropriate MIME types must be configured for jar and jad:

- JAR: application/java-archive
- JAD: text/vnd.sun.j2me.app-descriptor

Tip: Netbeans comes with built in obfuscator, which can be used to make the JAR files smaller. You can switch it on from project properties.

AL2 Application start up

Application must start properly in 25s

Branded splash screens are omitted. What is measured is time from launch to main menu (or interactive screen).

Typically, end-user wants fast response times, and because of that application start-up time should be minimized. It is always a good idea to provide some kind of progress dialog and splash screen if application loading takes much time. Otherwise end-user is likely to think that application has halted.

- Series 40: Considerations regarding MIDlet startup and runtime execution

In some cases, lazy initialization might help you to get rid of the crucial extra second.

User Interface (UI)

The intent is to not specify exactly how to design a user interface but rather to give general guidelines. It is expected that publishers and network operators will further define the look and feel of an application's user interface to make it more in conformance with their overall look and feel.

Apart from the look and feel, these tests are about usability of the application. You can find comprehensive UI and graphics section for creating compelling and cool Java applications from online Nokia Developer Java TM ME Developer's Library.

In addition, Nokia Developer Design and User Experience Library provides useful generic information on application design usability and user experience.

If you want to have a good insight on usability it is newer too late to familiarize yourself with one of the Jakob Nielsen's books.

UI1 Graphic clarity

All graphics and animations displayed must be readable and clear to the user

Devices can have several display resolutions, it is good to note that in some devices screen orientation can be changed during application execution, and application should be able to handle it.

High-level UI components are automatically scaled to screen resolution. In addition in S60 starting from S60 3rd Edition, Feature Pack 2, you can define scaling for Canvas graphics. For more information on setting scaling, please see Nokia Developer Java TM ME Developer's Library.

For setting fixed landscape starting from S60 5th Edition, note this Known Issue with 5800 XpressMusic (first release) in Nokia Developer Wiki.

UI2 UI consistency

The user interface of the application must be consistent throughout the application

(C) Copyright Nokia 2014. All rights reserved.

Check that command labels are consistent with actions. If you use sounds, do not use the same sound for positive and negative feedback.

UI3 Browsing through the application

The browsing through the application and inputting information must be clear and without unnecessary steps.

Make sure that there are no intermediate screens between commands and expected actions. E.g. selecting help will show help screen and not some another screen where you must select help again.

Localization (LO)

Applications that are to be deployed to localities other than their point of origin must account for changes in language, alphabets, date and money formats, etc.

LO1 Localisation boot test

Text present in the localised version of the application must be translated in the targeted language.

This test only covers the main menu command labels / texts. For other areas of the application, developers are responsible to ensure that all localisation criteria are respected throughout the application. Java Verified reserves the right to revoke approvals granted to any application that does not meet these criteria.

Three ways to localize application:

- Resource bundles Resource bundles
- Text files
- Application attributes

LO2 Translation accuracy

In every language of the application, all text must be translated with respect to the application and the targeted language. In case you provide translation to different languages, the whole application must be translated. Single untranslated word will not cause an error, but if the whole sentence is left translated, it is considered as an error.

LO3 Spelling errors

The application must be free of spelling errors. A spelling error is defined as a strict miss-spelling of a word (no grammar or punctuation rules will be applied). Missing diacrits and accents (e.g. acutes, cedillas, umlauts etc.) will not be reported as spelling errors.

LO4 Technical text errors

The text in the application must be clear and readable. The application must be free of technical text display issues such as: Text cut off / Text overlapping.

Translation of single word to different languages may result big variations in character count. Because of this, developers should check that translated words will fit in their places and they are not cut off in any direction.

MIDP 2.0 allows developer to control word wrapping by using `setFitPolicy(int fitPolicy)`. Where the fit policy is one of the following `TEXT_WRAP_DEFAULT`, `TEXT_WRAP_ON` or `TEXT_WRAP_OFF`. Read more from Sun Developer Network: http://developers.sun.com/mobility/midp/ttips/wordwrap/index.html

Functionality (FN)

Documented features are implemented in the application and work as expected. Sources for the information are user manuals, formatted application specification documents and online documentation.

Tests FN2 - FN8, FN16 and FN18 are all about pausing the application in case of external interruption. Application must not crash in any of these tests. Pausing is not required function if application does not require immediate user intervention. Application pausing is described in detail in section FN8 Pause.

FN1 Application hidden features

The application does not introduce any hidden features, its functionality set is consistent with the help and it does not harm the data on the device.All the features are introduced in the Help, the application has no hidden features

(C) Copyright Nokia 2014. All rights reserved.

FN2 voice call and FN16 video call

Interruption by an incoming video/voice call causes the application to enter the paused state. After the video call ends, the application either continues from the point where it was interrupted or present the user an option to continue.

FN3 SMS, FN4 MMS, FN5 Bluetooth FN6 infrared and FN18 Alarm Clock

- *Go into pause state, after the user exits the communication/alarm clock, the application presents the user with a continue option or is continued automatically from the point it was suspended at*

OR

- *Give a visual or audible notification*

FN7 charging

Application can either continue uninterrupted or to pause and ask if user wants to continue.

FN8 Pause

The application must support a pause feature in areas of the application where immediate user interaction is needed (for example in game). The pause feature must support an option to resume the application, and an option to go back to the main menu of the application. Pause screen must indicate clearly that the application is at pause state e.g. present a "paused" text. In addition pause screen must have a clear indication how the user can continue using the application.

Canvas and GameCanvas have two methods that are useful when implementing pause (in case of external interruption etc.) `showNotify()` and `hideNotify()`. They are called prior to a Canvas object is made visible or removed from the display. Conditional code block in paint method could be used to paint paused text on to the screen.

- http://developers.sun.com/mobility/midp/ttips/gamecanvas/index.html

High-level UI components can call `isShown()` to check if they are on the top.

- How to handle MIDlet sent to the background

Test case FN19 states that audio playback must also be paused when there is incoming call.

Tip.

- Series 40: you can use HyperTerminal for debugging call and message events. Use `System.out.println( "hideNotify called" );` to print messages. Right COM port can be seen from device manager under Ports section. For more details on debugging in Series 40, please refer to this blog article on MIDlets, System.out, and Series 40 phones by Hartti Suomela.
- S60: Install EcmtAgent_MIDP.SIS and RedMIDlet.jar (can be found form app manager after EcmtAgent installation). Start EcmtAgent and std.out redir. You can monitor System.out messages with <sdk dir>\Epoc32\tools\ecmt\ecmtgw.exe. Complete guide for setting up on device debugging Java OOD over WLAN with eclipse and NetBeans IDEs

FN9 Sound settings and FN10 Vibra settings

If application plays sounds or uses vibra there must be easy to use settings for both of them. Application settings must be stored and restored at start-up.

Following code snippets shows how to store application settings:

- How to store application settings

Controlling vibra settings:

- How to control vibra settings

FN11 Main menu requirements

Application main menu must contain following commands: Exit, About and Help

About screen must provide developer and application name and version of the application. This data must match to following jar/jad manifest fields: MIDlet-Vendor, MIDlet-Name and MIDlet-Version. About screen information can be included in to the Help screen.

FN12 Application response

(C) Copyright Nokia 2014. All rights reserved.

The application should never leave the user in a position where the state of the application is unknown or appears to be unresponsive (i.e. may have locked up). The application notifies the user when the user needs to wait for something longer than 5 seconds.

One way to fulfil this requirement is to use progress indicators. Using progress indicator with alert dialog

FN13 The speed of application in use

The application works in the device it was targeted for. It is usable on the device. The speed of the application is acceptable to the purpose of the application and must not alter the user experience by being uncontrollable.

Java ME applications must run in a wide variety of devices. One must ensure that application runs smoothly on the device it is targeted on. For games, in addition to generic optimization guidelines, one should also consider the situation when the game is played on more powerful phone. More CPU power means more FPS and hence the game may become uncontrollable, unless you slow it down by design.

- Introducing a delay to keep frame rate constant
- Keeping frame rate constant by using `System.currentTimeMillis()`

```
time = System.currentTimeMillis();
... repaint & update
time = delay – (System.currentTimeMillis() –time);
Thread.sleep( (time<0 ? 0 : time) );
```

Optimization articles:

- How to optimize in Java ME
- Java Optimazation Articles

FN14 Data deletion

The application must indicate whether data will be permanently deleted or offer easy reversal of the deletion. Check if there is a reversal (undo) available for the user or that the user is notified before deletion is permanent

One of the key elements in user experience is interaction. Before deleting data, user should be prompted for approval.

- How to implement a simple confirmation dialog

FN15 Unexpected user behaviour

The application must be able to handle unexpected user behaviour, for example erroneous actions and multiple key presses.

On S60 devices menu key or end key are not covered, because menu key moves the application to background and end (red) key quits the application. On Series 40, end (red) key quits the application.

Used can save the game state automatically if closed by pressing the red key (the save logic can be implemented within destroyApp() which is called after each red key press). Most likely, this is unintended key press and saving the game state will prevent player for losing his/her nerves.

FN16 External incoming communication – video call

Check FN2

FN17 System Shutdown

Application must correctly handle situations where it is closed due to system shutdown (terminal switched off).

FN18 External Interruption – Alarm Clock

Check FN8

FN19 Influence on Terminal System Features

Application must correctly handle situations where following user input, or some external event (e.g. a phone call), it is switched to the background by the terminal. Upon restart the application must resume its execution correctly. While in the background the application must not emit any audio and all handset functions should remain intact. While being in the background, the application

(C) Copyright Nokia 2014. All rights reserved.

must either not affect the use of the system features or other applications or, if the application does so, such behaviour must be described in the help file.

- If application execution makes some system features unavailable or affects them otherwise in a noticeable manner, and application help file doesn't describe such situations, the application must fail this test.
- If the application under test by its design is expected to produce audio output whilst in background mode (such as an MP3 player, IM client etc.) then this is acceptable so long as the audio is paused during external events as described in FN2, FN3, FN4, FN5, FN6, FN7, FN16 and is able to resume audio correctly.

Connectivity (CO)

If an application has communication capabilities then it must demonstrate its ability to communicate over a network correctly. It must be capable of dealing with both network problems and server-side problems.

You can find a lot of Java networking resources from Nokia Developer Java Developer's Library.

CO1 Network connectivity not allowed

When the application uses network capabilities, it must be able to handle situations where the network connection is not allowed.

More about Java Security Domains in Nokia Developer Wiki.

CO2 Network delays and the loss of connection

When the application uses network capabilities, it must be able to handle network delays and any loss of connection.

For example Connector class has method `open(String name, int mode, boolean timeouts)`. Passing `true` for timeouts enables connector class to throw `IOException` when timeout occurs.

CO3 Closing the network connection IP connections

When the application uses network capabilities, it must be able to use the connection correctly and correctly close it after using it.

SE1 test states that encryption must be used when communicating sensitive data. This example utilizes SSL when commincating with a web server.

CO4 Messaging

When the application uses the messaging capability of the device (e.g. SMS, MMS), it must be able to send messages correctly.

Code snippets in Nokia Developer Wiki:

- Sending SMS
- Sending MMS

CO5 Messaging errors

When the application uses the messaging capability of the device (e.g. SMS, MMS), it must be able to take into account error situations, such as device settings and display informative error messages.

The examples provided in the CO4 omitted exception handling. This test is all about implementing all "todo" sections. You must be sure that user gets meaningful and informational error messages and that application can recover from erroneous situations.

CO6 Bluetooth connections

When the application uses Bluetooth connections, it must be able to communicate correctly over Bluetooth and close the connection when exiting.

Guides on Bluetooth API (JSR-82) in Nokia Developer Java TM Developer's Library:

- Bluetooth connections: API overview and step by step examples
- Bluetooth examples

CO7 Bluetooth errors

When the application uses Bluetooth connections, it must be able to handle Bluetooth connection errors.

As the unified testing criteria states the application must clearly state that the connection to the other party is lost in the error message and it must not crash.

(C) Copyright Nokia 2014. All rights reserved.

Please read CO6

CO8 Push registration

Applications using Push Registry must be able to handle this correctly and must be able to Register Push Events (Auto launch events).

All push registry features used by an application should be entered in the connections section in the Application Characteristics. The Application gracefully handles situations where the user denies permission for registration.

- Push registry in Java TM ME Developer's Library
- CS001422 - Using PushRegistry in Java ME

CO9 Push activation

The application must be able to start via the Push Registry on a receipt of a registered event

Please read CO8

CO10 IrDA connections

When the application uses IrDA connections, it must be able to communicate correctly over IrDA and close the connection when exiting.

- Article on checking Java support for IrDA in Nokia Developer Wiki
- IrDA example on Using Java IrCOMM in the Nokia 5140 5140i

CO11 IrDA errors

When the application uses IrDA connections, it must be able to handle IrDA connection errors.

Please read CO10

CO12 Contactless Communication

When the application uses Contactless Communication (JSR-257), it must be able to communicate correctly.

Resources on contactless communication:

- An Introduction to Near-Field Communication and the Contactless Communication API by Sun
- Category:Near Field Communication (NFC)
- JSR 257 Contactless Communication API specification in Nokia Developer website. The zip-package includes jsr-257-spec-1.0.pdf and code examples.

CO13 Contactless Communication errors

When the application uses Contactless Communication (JSR-257), it must be able to handle communication errors correctly.

JSR-257 defines exceptions which must be handled in case of the errors. In addition, some Nokia devices may include extensions to JSR-257 and therefore include related additional exceptions (more on these extensions, please refer to the Javadocs on the SDKs with JSR-257 support).

Personal Information Management (PIM)

The application accessing user information needs to be able to do it in an appropriate manner and not to destroy the information.

PI1 Accessing personal information

The application must be able to handle the cases where the connection to the PIM applications is not allowed.

You can simulate this situation by navigating to the device settings and set the read / write user data to "Not allowed"

PIM security model conforms that handling personal data has privacy and security implications. In addition to this generic guideline for PIM API, network operators may have more strict policy regarding accessing this data with PIM API. Due to these differences, it is good to ensure that application can handle situations when access to PIM is limited or declined altogether.

Trying to read or write without proper permission throws java.lang.SecurityException.

Requesting read and write permissions for PIM API database types ContactList, EventList and ToDoList must be expressed with the attribute 'MIDlet-Permissions' in JAD and JAR Manifest. For example for EventList, the permission must be set in JAD as

(C) Copyright Nokia 2014. All rights reserved.

follows:

MIDlet-Permissions: javax.microedition.pim.EventList.read, javax.microedition.pim.EventList.write

If reading or writing fails, end user should be notified on the failure by using display a nice informative (not cryptic) message.

Read more about PIM:

- PIM API Developers Guide
- Using the PIM API for Java ME, Part 3 - Security Considerations by Sun

Info on operator and manufacturer specific support for related API protection groups (read user data, write/edit user data) in Nokia Developer Wiki:

- Java Security Domains

PI2 Using personal information

The application must be able to connect to the PIM applications correctly and not destroy any content without the explicit permission of the user.

Please read FN14 Data deletion

Security (SE)

Listing different security related issues tested from the applications.

SE1 Application Declaration

Check the application declarations in the "Application Characteristics". It has been declared that the application uses encryption when communicating sensitive data

- CS001314 - Encrypting and decrypting in Java ME

SE2 Passwords

Passwords or other sensitive data are not stored in the device and the passwords are not echoed when inputted to the application.

Passwords, credit card details, or other sensitive data is not stored at the fields where they were previously entered.

The easiest way to implement password field is to use built in high-level ui component.

```
TextField password =  new TextField("Password", "", 8, TextField.PASSWORD);
```

SE3 JAD/JAR manifest information accuracy

The JAD file and JAR manifest file MIDlet-Permissions attributes must contain exactly the same information or the application will not install after it has being signed.

Compare the MIDlet-Permission and MIDlet-Permission-Opt attributes in the files with each other and with the Declarations section of the application document

Tip: As the manifest information inaccuracy results into problems when installing signed applications. Here are some other items which can be checked if the signed application does not install.

MIDlet attributes The MIDlet fields must be written in an identical manner including the use of spaces. There are only two exceptions: `MIDlet-Jar-Size` & `MIDlet-Jar-URL` attributes which are not available in the JAR file.

Warning: MIDlet attributes must be the same in both JAR manifest and the JAD! Otherwise the application will not work after being signed.
For more information on signing and Java verfied see this FAQ. In particular topics "Signing Java ME applications" and "Why doesn't my signed midlet work?"

Permissions Permissions are used to protect Application Programming Interfaces (APIs) that are sensitive and require authorization. If permissions are not declared, then the application will not be able to implement functions that require permissions. Thus, permissions need to be defined to make the application work properly. Permissions are defined in the `MIDlet-Permissions` attribute. For example:

(C) Copyright Nokia 2014. All rights reserved.

- `MIDlet-Permissions: javax.microedition.io.Connector.http`

Multiple permissions need to be separated with a comma. For example:

- `MIDlet-Permissions: javax.microedition.io.Connector.http, javax.microedition.io.Connector.sms`

The device and the certificate The device needs to have a certificate from a certificate authority (CA) to make the installation possible. It can be difficult to check the existence of that certificate as the certificate name may vary from device to device and the list of certificates is located in different places in different devices.

If the certificate is not there, but the application is signed, you can install the application by removing the following fields from the JAD:

- `MIDlet-Certificate-` fields
- `MIDlet-Jar-RSA-SHA1` field

Note: Before removing the fields, check that the signature in the JAD file and the root certificate in the device are valid from the perspective of the device. The easiest way to do this is to confirm that the date and the time of the device are correct. To find guidance on how to change the time and date of the device, please check the device manual.

The validity period To check the validity period of a root certificate in the device, locate the root certificate and the detailed view for the certificate should give information on the period when the certificate is valid. To check the validity of the certificate in the JAD file, follow these steps:

1. Open the JAD file with a text editor – notepad is just fine.
2. Copy the contents of the `MIDlet-Certificate-1-1:` field (text starting from the colon and ending to the carriage return) and paste it to an empty text file.
3. Save the text file.
4. Change the file extension to .cer
5. Open the file and the validity is presented in the "Valid from" field.

MicroEdition-Configuration and MicroEdition-Profile There are certain cases where the MicroEdition-Configuration and MicroEdition-Profile have not been included with correct configuration and profile information in the JAD and JAR manifest files and thus the application has not worked after it was signed. For example: MicroEdition-Configuration: CLDC-1.0 MicroEdition-Profile: MIDP-2.0

(C) Copyright Nokia 2014. All rights reserved.