Java Card Tools for Together Control Center

Wojciech Mostowski
Chalmers University of Technology, Göteborg, Sweden
Computing Science Department
e-mail: `woj@cs.chalmers.se`

Abstract

This is a description of the Java Card Tools package for Together Control Center. The package supports the development of Java Card applets (writing, compiling, installing, testing, etc.) inside the Together Control Center CASE tool.

1 Introduction

Java Card technology [2] provides means to program Smart Cards using a subset of the Java language. Nowadays there is a great variety of different Java Card devices/platforms available on the market. Although they all have to conform to the Java Card specification, they usually differ slightly from each other. It's very common that each Java Card device producer provides its own Java Card development environment (it can be a complete Integrated Development Environment or just a set of tools/scripts/compilers). None of those tools provide a uniform, high level, UML supported environment for creating Java Card applications. Some of those tools are not very user friendly either. The Java Card Tools package tries to solve those inconveniences. It is an extension of a commercial CASE tool to provide support for different Java Card devices/platforms. The main goal is to provide a push-button technology inside the CASE tool that makes Java Card development easy and uniform for different kinds of Java Card devices and overcome vendor specific Graphical User Interfaces at the same time. The CASE tool in question is Together Control Center (later referred to as TogetherCC) from TogetherSoft [8]. There are two reasons for this choice. First of all TogetherCC is a state-of-the-art CASE tool with excellent UML support and open architecture, which makes writing extensions (in Java) an easy task. Second TogetherCC is the CASE tool used (as a base) in the KeY system. The KeY project [1, 5] aims at integrating object-oriented design with formal methods. The target language for the KeY project is Java Card due to its relative simplicity. So Java Card Tools can be seen as an extra support for the KeY system.

In Section 2 we describe the two Java Card platforms that are currently supported by Java Card Tools in detail. Section 3 presents the capabilities of Java Card Tools and describes an example usage. Section 4 gives some of the implementation details, Section 5 gives some directions for future extensions and finally Section 6 gives some pointers to the Java Card Tools documentation and download web site.

2 Java Card Platforms Supported

Currently there are two Java Card platforms/kits supported by Java Card Tools, however, extending it to support other platforms is possible and the Java Card Tools architecture makes writing extensions a relatively easy task. In the following subsections we describe the supported platforms in more detail.

2.1 Sun Java Card Development Kit

Sun Java Card Development Kit (jcdk in short) is a reference implementation of the Java Card development kit and tools. On its own the kit does not operate on any real Java Card devices, but such a device can be simulated or emulated by the kit tools. The package includes:

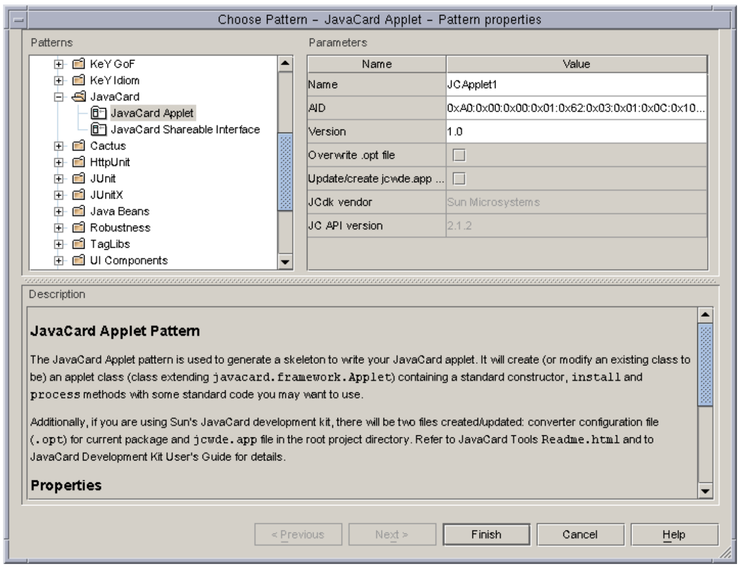
- Class file verifier and converter (`converter` tool), which makes sure that a given Java Card applet complies to Java Card language restrictions and creates a `.cap` file suitable for downloading to a Java Card device.
- `jcwde` tool (Java Card Workstation Development Environment), which is a simple Java Card simulator (e.g. it does not allow saving the state of a smart card between subsequent runs).
- `cref` tool (C reference implementation of a Java Card environment). This tool serves as a fully operational Java Card emulator. It operates on Java Card EEPROM images and allows saving a smart card's state after each session in form of such an EEPROM image.
- `apdutool`, which is used to send Application Protocol Data Units to a simulated/emulated Java Card. APDUs are the only means of communication between a smart card and the host application/system.

2.2 Dallas Semiconductor Java Powered iButtons

Dallas Semiconductor Java Powered iButton [4] is a Java Card device embedded in a small button (buttons are more durable than normal smart cards). iButtons implement Java Card API version 2.0. A development environment for iButtons (iB-IDE) is provided free of charge and can be downloaded from the web [3]. iB-IDE provides the following tools and functionality:

- an integrated development environment for creating both Java Card applets and host applications. Skeleton code can be generated for both with the help of a project wizard.
- an APDU sender, which provides low level access to iButtons: downloading and removing applets from iButtons, changing iButton settings (e.g. iButton password), sending arbitrary APDU packets to iButton, etc.
- iButton emulator, which can be used for testing applets before they are downloaded to the real iButton.

iB-IDE is built on top of low level libraries with well documented API, which makes it possible and relatively easy to reuse the libraries and build a new set of tools to operate on iButtons.



Name	Value
Name	JCApplet1
AID	0xA0:0x00:0x00:0x01:0x62:0x03:0x01:0x0C:0x10...
Version	1.0
Overwrite .opt file	☐
Update/create jcwde.app ...	☐
JCdk vendor	Sun Microsystems
JC API version	2.1.2

JavaCard Applet Pattern

The JavaCard Applet pattern is used to generate a skeleton to write your JavaCard applet. It will create (or modify an existing class to be) an applet class (class extending `javacard.framework.Applet`) containing a standard constructor, `install` and `process` methods with some standard code you may want to use.

Additionally, if you are using Sun's JavaCard development kit, there will be two files created/updated: converter configuration file (`.opt`) for current package and `jcwde.app` file in the root project directory. Refer to JavaCard Tools `Readme.html` and to JavaCard Development Kit User's Guide for details.

Properties

Figure 1: *Java Card Applet* pattern

3 Support for Java Card Development

Java Card Tools is a uniform front-end to the tools and libraries described above. Since the tool set is embedded in TogetherCC it is possible to use the full support of the CASE tool and operate with Java Card devices and emulators at the same time.

The support for Java Card can be divided into two main parts:

- Java Card patterns (code skeletons),
- user friendly, easily accessible commands and tools to test applets by means of an APDU sender, which is used to "talk to" Java Card devices (either emulated/simulated or real). Those commands also include helper commands to prepare and download applets to Java Card devices.

3.1 Java Card Patterns

There are two patterns available at the moment, namely *Java Card Applet* and *Java Card Shareable Interface*. The latter is only available when Java Card API version 2.1 or higher is used (i.e. it's not applicable when iButtons are used).

As an example, Figure 1 shows the dialog of the *Java Card Applet* pattern. After applying this pattern to the project the skeleton code for a Java Card applet is created as well as some files necessary for the development kit used (in this case Sun jcdk). The skeleton applet code generated by the pattern looks as follows:

```
/* Generated by Together */
```

```
package testapplet;

import javacard.framework.*;
public class JCApplet1 extends Applet {
    protected JCApplet1(){
        // Write init code here
        register();
    }

    /**
     * @param bArray array with initialisation data
     * @param bOffset offset into this array
     * @param bLength length of the data
     */
    public static void install(byte[] bArray,
                               short bOffset, byte bLength){
        new JCApplet1();
    }

    /**
     * @param apdu the incoming apdu packet to process
     */
    public void process(APDU apdu){
        byte buffer[] = apdu.getBuffer();
        if ((buffer[ISO7816.OFFSET_CLA] == ISO7816.CLA_ISO7816) &&
            (buffer[ISO7816.OFFSET_INS] == ISO7816.INS_SELECT)) {
            // that was the SELECT APDU
        }
    }
}
```

The *Java Card Shareable Interface* pattern simply creates an empty Java Card shareable interface like the following:

```
/* Generated by Together */

package testapplet;

import javacard.framework.*;
public interface MyInterface extends Shareable {
}
```

3.2 Installing and Testing Applets

The remaining Java Card Tools functionality is accessed through TogetherCC's class diagram context (pop-up) menu (*Java Card tools. . .* menu group). Figure 2 shows the full structure of this menu for both Sun Java Card Development Kit and Dallas Semiconductor iButtons. Two menu commands are always present regardless of the Java Card kit used, namely *Display Java Card environment info* and *Rediscover Java Card environment info*. The first one displays all the necessary information about the Java Card kit that is currently used and the second one reconstructs this information in case the user changed the kit. In the following we demonstrate how some of the other commands listed in Figure 2 are used for Java Card applet installation, testing and management.

Sun Java Card Development Kit:

- Display Java Card environment info
- Rediscover Java Card environment info
- Converter tools...
    - Create/update `.opt` file for this package/applet
    - Run converter for this package/applet
- Jcwde tools...
    - Create/update `jcwde.app` file
    - Start APDU sender
- Cref tools...
    - Create virtual card EEPROM image
    - Start APDU sender

Dallas Semiconductor iButtons:

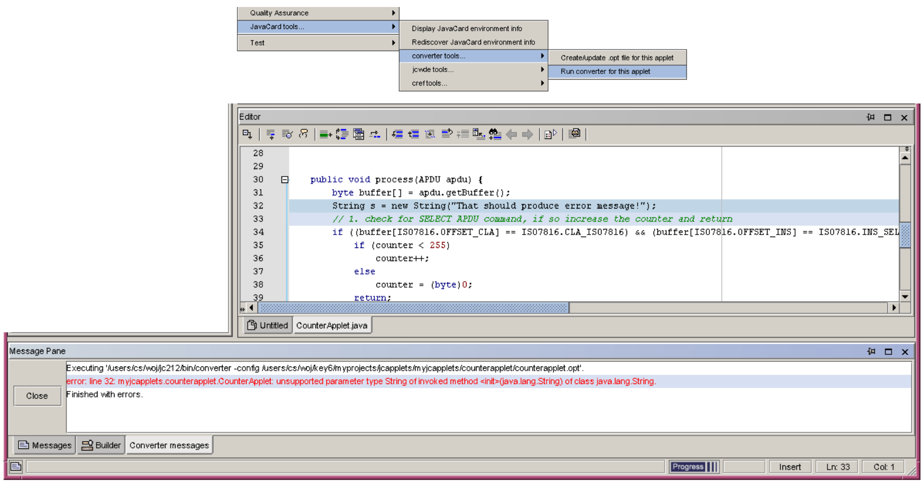
- Display Java Card environment info
- Rediscover Java Card environment info
- JiBlet...
    - Build JiBlet for this applet
- Applet administration...
    - Get iButton information
    - Load this applet to iButton
    - List applets installed on iButton
    - Remove this applet from iButton
    - Remove applet from iButton by name
    - Master erase iButton
    - Set new password for iButton
- APDU sender...
    - Start APDU sender for this applet
    - Start APDU sender by name

Figure 2: Java Card Tools pop-up menu structure

3.2.1 Preparing the Applet

Suppose a Java Card applet is ready for testing at some point. The first thing that has to be done is applet compilation. The standard TogetherCC tools are used for that. After that, the compiled `.class` file(s) need to be converted to a proper format. In case of Sun's jcdk that is a `.cap` file. To create a `.cap` file one needs to invoke *Run converter for this applet* command from the Java Card Tools pop-up menu. The messages from the converter are displayed in TogetherCC's message windows with 'jump to error location' feature. In the example shown in Figure 3 a Java Card applet uses a forbidden class `String`. By clicking on a message the source of the error is displayed in the editor window.

When Java Powered iButton is used then the JiBlet file (`.jib`) needs to be built by invoking *Build JiBlet for this applet* command from Java Card Tools menu. It works almost in the same way as the `converter` tool.



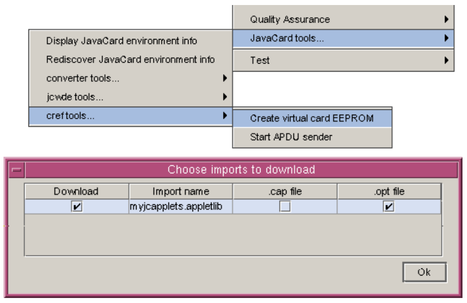
```
    public void process(APDU apdu) {
        byte buffer[] = apdu.getBuffer();
        String s = new String("That should produce error message!");
        // 1. check for SELECT APDU command, if so increase the counter and return
        if ((buffer[ISO7816.OFFSET_CLA] == ISO7816.CLA_ISO7816) && (buffer[ISO7816.OFFSET_INS] == ISO7816.INS_SEL
            if (counter < 255)
                counter++;
            else
                counter = (byte)0;
            return;
```

```
Executing '/users/cs/woj/jc212/bin/converter -config /users/cs/woj/key6/myprojects/jcapplets/myjcapplets/counterapplet/counterapplet.opt'.
error: line 32: myjcapplets.counterapplet.CounterApplet: unsupported parameter type String of invoked method <init>(java.lang.String) of class java.lang.String.
Finished with errors.
```

Figure 3: The `converter` tool

3.2.2 Testing Applets with Sun Jcdk Tools

After a `.cap` file for an applet is created, we can create a card's EEPROM image with that applet and run JAVA CARD emulator on the EEPROM image. To create an EEPROM image the *Create virtual card EEPROM image* command has to be invoked from the *Cref tools* submenu. If the applet uses some additional libraries, which are not part of the current package the user is asked whether those libraries should be included in the card's EEPROM image (that's usually the case), see Figure 4. Then an appropriate script is invoked and the EEPROM image is created with all the necessary messages displayed in TogetherCC's message window.



Choose imports to download

Download	Import name	.cap file	.opt file
✓	myjcapplets.appletlib	☐	✓

Figure 4: Creating a card EEPROM image

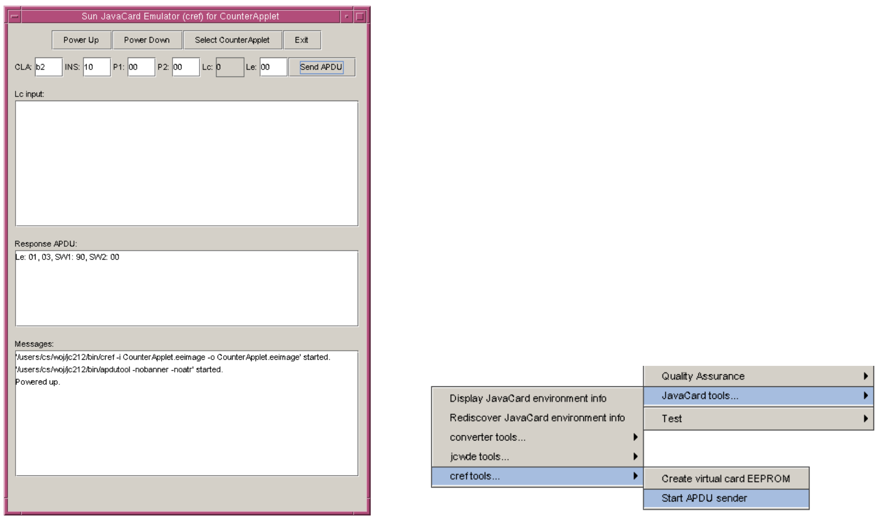
After the EEPROM image is created we can start an APDU sender for an emulated card. This is done by invoking *Cref tools/Start APDU sender* command. A window like the one in Figure 5 is displayed.

By pressing the 'Power up' button the `cref` emulator is started up. Then an applet can be selected on a virtual card by pressing the 'Select applet' button. Then any APDU packet can be constructed and sent to the emulator. The results will be displayed in the response panel. After pressing 'Power down' the current EEPROM image of the emulated card is saved, so that it can be used again for further testing. Pressing 'Exit' closes the APDU sender window (and also makes sure that the EEPROM image is saved).

Another option is to test the applet using the `jcwde` card simulator. It works in a similar way to `cref` except that the EEPROM image is not used and therefore does not have to be created.

3.2.3 Testing Applets on iButtons

Once a JiBlet file is created for an applet, it can be downloaded to the iButton. This is done by invoking *Load this applet to iButton* command from the *Applet administration* submenu. If there is more than one iButton attached to the computer the user will be asked to choose one for downloading the applet. After successful download an APDU sender can be started to communicate with the iButton. This is done by invoking *Start APDU sender for this applet* command. Exactly the same window as described earlier pops up (see Figure 6). Again 'Power up' button should be pressed to initialise the iButton, then the given applet can be selected by pressing 'Select applet' and then the user can start sending arbitrary APDUs to the applet and watch the responses.



CLA: b2 INS: 10 P1: 00 P2: 00 Lc: 0 Le: 00

Lc input:

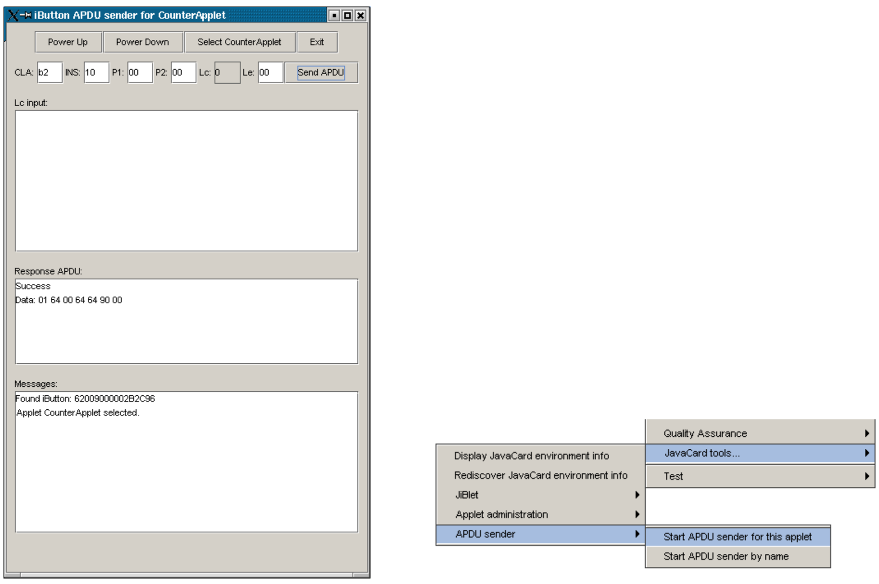
Response APDU:

```
Le: 01, 03, SW1: 90, SW2: 00
```

Messages:

```
'/users/cs/woj/jc212/bin/cref -i CounterApplet.eeimage -o CounterApplet.eeimage' started.
'/users/cs/woj/jc212/bin/apdutool -nobanner -noatr' started.
Powered up.
```

Figure 5: APDU sender window running `cref` emulator


iButton APDU sender for CounterApplet
Power Up
Power Down
Select CounterApplet
Exit
Send APDU
Display JavaCard environment info
Rediscover JavaCard environment info
JiBlet
Applet administration
APDU sender
Quality Assurance
JavaCard tools...
Test
Start APDU sender for this applet
Start APDU sender by name

CLA: b2 INS: 10 P1: 00 P2: 00 Lc: 0 Le: 00

Lc input:

Response APDU:

```
Success
Data: 01 64 00 64 64 90 00
```

Messages:

```
Found iButton: 62009000002B2C96
Applet CounterApplet selected.
```

Figure 6: APDU sender window for an iButton

In the current version of JAVA CARD Tools there is no possibility to operate on an emulated iButton, only real iButtons. This is due to the lack of API documentation for the iButton emulator library.

Other iButton commands. There are a number of other, very useful commands in the *Applet administration* submenu for iButtons. They allow the following things:

- listing all the applets installed on an iButton,
- removing applets from iButtons,
- getting full iButton device information (ID string, firmware version, free memory, etc.),
- master erasing an iButton,
- changing the access password for an iButton.

4 A Few Words About The Implementation

As already mentioned JAVA CARD Tools is just a front-end to a set of tools and libraries. As TogetherCC provides an open JAVA API, writing plug-ins and extensions is a relatively simple task—most of the internals of TogetherCC are available to the programmer. JAVA CARD Tools provides their support by extending TogetherCC's class diagram pop-up menu and adding new patterns to the TogetherCC pattern collection. To interface the tool set with Sun JAVA CARD Development Kit, processes running required tools (`apdutool`, `cref`, ...) are started inside TogetherCC and the messages are passed to and from appropriate windows (TogetherCC message window, APDU sender window, etc.). The iButton support and communication is provided by the JiBlet JAVA library (`JiB.jar`) and iButton 1-Wire JAVA library (`OneWireAPI.jar`). Operating and communicating with iButtons is implemented by a few simple calls to those libraries. Both of those libraries are provided with iButton Integrated Development Environment free of charge.

JAVA CARD Tools package relies on TogetherCC's API, however there are no obstacles to make the package work with another CASE tool as long as the tool provides the necessary functionality and API for JAVA CARD Tools to function properly (e.g. access to a project's class path, access to the tool's editor, etc.). Also, as mentioned earlier, the JAVA CARD Tools package is easily extensible to support other JAVA CARD development kits and devices. For example, since most of the APDU sender functionality is independent of the device used, to enable sending APDUs to a JAVA CARD device with APDU sender it is basically enough to write a JAVA class that will communicate with the device and then just plug it in to the APDU sender class through a very simple interface.

5 Possible Extensions

One of the things that should be considered as a future extension is a generic support for any JAVA CARD platform that implements Open Card API [7]. The menu structure could be improved by abstracting the common commands for different development kits (e.g. *Prepare the applet for download*). Such menu abstraction would make the menus more uniform, independent of the actual Java Card platform and easier to use for unexperienced users. On the other hand, however, experienced users familiar with a given development kit may actually prefer the concrete version of the menu to have more control over the actual tools they invoke with menu commands.

Furthermore some small extensions are possible, like having a set of an appropriately formed APDUs for all commands predefined in an applet to choose from in APDU sender window, etc. Also some more useful Java Card patterns and code skeletons can be added to the pattern collection.

6 Further Information

A detailed User Guide and Installation Guide in html format can be found in Java Card Tools package as well as on the package web site [6]. The package itself (currently version 2.0—`jctools-2.0.zip`) can be downloaded from this page too.

References

[1] Wolfgang Ahrendt, Thomas Baar, Bernhard Beckert, Martin Giese, Reiner Hähnle, Wolfram Menzel, Wojciech Mostowski, and Peter H. Schmitt. The KeY system: Integrating object-oriented design and formal methods. In Ralf-Detlef Kutsche and Herbert Weber, editors, *Fundamental Approaches to Software Engineering. 5th International Conference, FASE 2002 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002 Grenoble, France, April 2002, Proceedings*, volume 2306 of *LNCS*, pages 327–330. Springer, 2002.

[2] Zhiqun Chen. *Java Card Technology for Smart Cards*. Addison Wesley, 2000.

[3] iB-IDE homepage. `http://www.ibutton.com/iB-IDE/`.

[4] iButton homepage. `http://www.ibutton.com/`.

[5] KeY project homepage. `http://i12www.ira.uka.de/~projekt/`.

[6] Wojciech Mostowski. Java Card Tools for Together Control Center webpage. `http://www.cs.chalmers.se/~woj/javacard/`.

[7] Open Card homepage. `http://www.opencard.org/`.

[8] TogetherSoft homepage. `http://www.togethersoft.com/`.