Ant Tasks User's Guide

Java Card™ Platform, Version 2.2.2

Sun Microsystems, Inc.
www.sun.com

3-15-06

Contents

1. Introduction 1

2. Installing the Ant Tasks 3

System Requirements 3

Installing Apache Ant 3

Installing the Ant Tasks 4

Setting Up the Ant Tasks 4

Library Dependencies 5

3. Using the Ant Tasks 7

APDUTool 7

Errors 8

Examples 8

CapDump 10

Errors 10

Examples 10

Capgen 11

Errors 11

Examples 11

Converter 13

Parameters Specified As Nested Elements 14

AppletNameAID 14

Errors 14

Examples 14

DeployCap 16

Errors and Return Codes 16

Examples 16

Exp2Text 18

Errors 18

Examples 18

Maskgen 20

Parameters Specified As Nested Elements 20

JCAInputFile 20

Errors 21

Examples 21

Scriptgen 22

Errors 22

Examples 22

VerifyCap 24

Parameters Specified As Nested Elements 24

ExportFiles 24

Errors 24

Examples 25

VerifyExp 26

Parameters Specified As Nested Elements 26

ExportFiles 26

Errors 26

Examples 26

VerifyRev 28

Parameters Specified As Nested Elements 28

ExportFiles 28

Errors 28

Examples 28

4. Custom Types 31

AppletNameAID 31

Example 31

JCAInputFile 32

Examples 32

ExportFiles 32

Examples 32

5. NetBeans™ Software Integration 35

6. Ant Tasks Example 39

Preface

This book describes how to use the optional and unsupported Ant tasks included in the Ant tasks bundle of the Java Card™ development kit. These tasks are designed to work with version 1.6.2 of Apache Ant (as verified with Ant version).

Java Card technology combines a portion of the Java™ programming language with a runtime environment optimized for smart cards and related, small-memory embedded devices. The goal of Java Card technology is to bring many of the benefits of the Java programming language to the resource-constrained world of smart cards.

Who Should Use This Book

This book is intended to assist users of the Java Card development kit in using the tools included in the Ant tasks bundle, particularly if the users are already familiar with Ant. Use of Ant as described in this document is unsupported.

Before You Read This Book

Before reading this guide, you should be familiar with Apache Ant, the Java programming language, Java Card technology, and smart card technology. A good resource for becoming familiar with Java technology and Java Card technology is the Sun Microsystems, Inc. web site, located at:

`http://java.sun.com`

Related Books

The following documents might prove useful:

- *User's Guide for the Java Card Platform, Version 2.2.2*.
- *Programming Notes for the Java Card Platform, Version 2.2.2*.
- Apache Ant documentation at `http://ant.apache.org`.

Typographic Conventions

TABLE P-1 Typographic Conventions Used in This Book

Typeface	Meaning	Examples
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`

`AaBbCc123`	What you type, when contrasted with on-screen computer output	`%` **`su`**
		`Password:`

AaBbCc123	Book titles, new words or terms, words to be emphasized	Read Chapter 6 in the *User's Guide*. These are called *class* options. You *must* be superuser to do this.
	Command-line variable; replace with a real name or value	To delete a file, type `rm` *filename*.

Accessing Sun Documentation Online

Access Java platform technical documentation on the web at the Java Developer Connection™ program web site at:

`http://java.sun.com/reference/`

Sun Welcomes Your Comments

We are interested in improving our documentation and welcome your comments and suggestions. Email your comments to us at `docs@java.sun.com`.

CHAPTER 1

Introduction

The Java Card platform, version 2.2.2 development kit comes with a set of command line tools to help developers create, verify and test their Java Card-based applications. Those tools use Apache Ant, but users are not required to use Ant directly. For more information on the development tools and how Ant is used by them, refer to *Development Kit User's Guide for the Java Card Platform, Version 2.2.2*.

However, the Ant tasks as described in this book are included in the Ant tasks bundle, not the development kit. These Ant tasks are designed for Ant users who might wish to use them to make use of the development kit tools more efficient. Use of the Ant tasks as described in this book is strictly optional and not supported by Sun Microsystems, Inc.

Javadoc™ tool files for the Ant tasks are located in this bundle in HTML format at `java_card_kit-2_2_2/ant-tasks/docs/html/javadocs`. A compilation of the Javadoc tool files in PDF format is at `java_card_kit-2_2_2/ant-tasks/docs/pdf/ant-tasks-javadocs.pdf`.

CHAPTER 2

Installing the Ant Tasks

System Requirements

To use the current version of these tasks, you will need version 1.5.0_03 of the Java Runtime Environment and version 1.6.2 of Apache Ant installed on your system. You can obtain Apache Ant on documentation on Ant from the Apache web site at `http://ant.apache.org`.

The Ant tasks included in the Ant Tasks bundle have been tested on Solaris™ 10 operating system and Windows XP (Pro), though they might work on other platforms as well.

Installing Apache Ant

The development kit requires Ant to run its tools and the demos. Once Ant is installed, the use of Ant in the development kit will not be apparent. If you have already installed Apache Ant version 1.6.2 for use in the development kit, you can skip this procedure.

Note – Ant is supported for use within the development kit, but its use as described in this book is not supported, nor have the Ant tasks been thoroughly tested.

---

1. **Download and unzip Apache Ant in a separate directory.**

If you don't already have Apache Ant version 1.6.2 installed on your system, you must download it from their web site at `http://ant.apache.org`. Unzip the package in a directory that is separate from the development kit.

2. **Add Ant to your system path.**

   Add Ant's `bin` directory to your system path.

Installing the Ant Tasks

1. **Install Ant as described in "Installing Apache Ant" on page 3.**
2. **Unzip the Ant tasks bundle.**

   If you haven't already, unzip the Ant tasks bundle, which is included in the binary product and named `java_card_kit-2_2_2-rr-ant-tasks.zip`. When you unzip the Ant tasks bundle, the Ant tasks' Java Archive (JAR) file, `jctasks.jar`, is extracted into the subdirectory `java_card_kit-2_2_2/ant-tasks/lib`. This user's guide is extracted into the subdirectory `java_card_kit-2_2_2/ant-tasks/docs` in PDF and HTML format.
3. **Copy the file `jctasks.jar` to a directory that will serve as your Ant tasks home directory.**

   This directory will be called `JC_ANT_TASK_HOME` throughout this book.
4. **Add the file `JC_ANT_TASK_HOME/lib/jctasks.jar` to your classpath or put `jctasks.jar` into a directory named `ANT_HOME/lib`, from which it will automatically be picked up when Ant is run.**

Setting Up the Ant Tasks

The following XML must be added your `build.xml` file to use the Ant tasks in your build.

```
<!-- Definitions for tasks for Java Card tools -->

<taskdef name="apdutool"

  classname="com.sun.javacard.ant.tasks.APDUToolTask" />

<taskdef name="capgen"

  classname="com.sun.javacard.ant.tasks.CapgenTask" />

<taskdef name="maskgen"
```

```
  classname="com.sun.javacard.ant.tasks.MaskgenTask" />
<taskdef name="deploycap"
  classname="com.sun.javacard.ant.tasks.DeployCapTask" />
<taskdef name="exp2text"
  classname="com.sun.javacard.ant.tasks.Exp2TextTask" />
<taskdef name="convert"
  classname="com.sun.javacard.ant.tasks.ConverterTask" />
<taskdef name="verifyexport"
  classname="com.sun.javacard.ant.tasks.VerifyExpTask" />
<taskdef name="verifycap"
  classname="com.sun.javacard.ant.tasks.VerifyCapTask" />
<taskdef name="verifyrevision"
  classname="com.sun.javacard.ant.tasks.VerifyRevTask" />
<taskdef name="scriptgen"
  classname="com.sun.javacard.ant.tasks.ScriptgenTask" />
<typedef name="appletnameaid"
  classname="com.sun.javacard.ant.types.AppletNameAID" />
<typedef name="jcainputfile"
  classname="com.sun.javacard.ant.types.JCAInputFile" />
<typedef name="exportfiles"
  classname="org.apache.tools.ant.types.FileSet" />
```

Library Dependencies

The libraries from the Java Card development kit in TABLE 2-1 are needed in your classpath if you are using the indicated feature. Alternatively, you can specify the classpath nested element for each task to put the required JAR files in the classpath during build execution.

TABLE 2-1 Library Dependencies

LIBRARIES	FEATURES
converter.jar and offcardverifier.jar	Creating CAP, EXP or JCA files. Using `maskgen` to create a mask. Dumping contents of an EXP file in a text file.
offcardverifier.jar	Verifying EXP files, CAP files and verifying binary compatibility between two versions of an export file.
apdutool.jar and apduio.jar	Sending an APDU script to `cref`.
capdump.jar	Dumping contents of a CAP file.
scriptgen.jar	Generating a APDU script from a CAP file.
apdutool.jar, apduio.jar and scriptgen.jar	Installing a CAP file in `cref` and generate resulting EEPROM image.

CHAPTER **3**

Using the Ant Tasks

The eleven Ant tasks provided in the Ant tasks bundle might simplify the use of the development kit tools for Ant users. This chapter describes these Ant tasks and how to use them. Note that the JAR files for the tasks are expected to be in the system classpath, unless otherwise noted.

- "APDUTool" on page 7.
- "CapDump" on page 10.
- "Capgen" on page 11.
- "Converter" on page 13.
- "DeployCap" on page 16.
- "Exp2Text" on page 18.
- "Maskgen" on page 20.
- "Scriptgen" on page 22.
- "VerifyCap" on page 24.
- "VerifyExp" on page 26.
- "VerifyRev" on page 28.

APDUTool

Runs APDUTool to send the APDU script file to `cref` and check if all APDUs were sent correctly. You can set `CheckDownloadFailure=true` to stop the build if any response status is not 9000.

APDUTool is invoked in a different instance of the Java™ Virtual Machine[1] (VM) than the one being used by Ant.

TABLE 3-1 Parameters for APDUTool

Attribute	Description	Required
ScriptFile	Fully qualified path and name of the APDU script file.	Yes
CrefExe	Fully qualified path and name of `cref` executable.	Yes
OutEEFile	Output EEPROM file that will contain the EEPROM image after `cref` finishes execution.	Yes
CheckDownload Failure	Stops the build if any response status coming back from `cref` is not 9000.	No
classpath	Classpath to use for this task. If required JAR files are not already in the system classpath, you can specify this attribute to put them in the classpath when this task is executed.	No
dir	The directory to invoke the Java VM in.	No
InEEFile	Input EEPROM file for `cref`. If specified `cref` initiates using the EEPROM image stored in this file.	No
nobanner	Set this element to `true` if you do not want the APDUTool banner showing.	No
version	Prints the version number of APDUTool.	No

Errors

Execution of this task fails if any of the required elements are not supplied, if `apdutool.jar` and `apduio.jar` are not in the classpath, or if APDUTool returns an error code.

Examples

Runs APDUTool to send APDUs in APDU script file `test.scr` to `cref` and to check if all APDUs were sent correctly. Also checks that the response returned from the card was 9000.

```
<target name="APDUToolTarget" >
         <apdutool
              scriptFile="${samples.helloworld.script}"
              outEEFile="${samples.eeprom}/outEEFile"
```

1. The terms "Java Virtual Machine" and "JVM" mean a Virtual Machine for the Java(TM) platform.

```
              CrefExe="${jcardkit_home}/bin/cref.exe">
         </apdutool>
</target>
```

Run the APDUTool to install the APDU script in `test.scr` file to `cref` and check if the APDU commands were processed successfully. Classpath in this example is referenced by the `classpath` refid.

```
<target name="APDUToolTarget" >
  <apdutool
       scriptFile="${samples.helloworld.script}"
       outEEFile="${samples.eeprom}/outEEFile"
       CheckDownloadFailure="true"
       CrefExe="${jcardkit_home}/bin/cref.exe">
       <classpath refid="classpath"/>
  </apdutool>
</target>
```

Run APDUTool to install the APDU script in `test.scr` file to `cref`, which is initialized using a stored EEPROM image from the file `inEEFile`. Also check if the APDU commands were sent correctly. Classpath used in this example is referenced by the `classpath` refid.

```
<target name="APDUToolTarget" >
  <apdutool
       scriptFile="${samples.helloworld.script}"
       outEEFile="${samples.eeprom}/outEEFile"
       inEEFile="${samples.eeprom}/inEEFile"
       CheckDownloadFailure="true"
       CrefExe="${jcardkit_home}/bin/cref.exe">
       <classpath refid="classpath"/>
  </apdutool>
</target>
```

CapDump

Runs the CapDump tool to dump the contents of a CAP file.

TABLE 0-1 Parameters for CapDump

Attribute	Description	Required
CapFile	Fully qualified name of CAP file.	Yes
classpath	Classpath to use for this task. If required JAR files are not already in the system classpath, you can specify this attribute to put them in the classpath when this task is executed.	No
dir	The directory to invoke the Java VM in.	No

Errors

Execution of this task fails if CapFile element is not supplied, if `capdump.jar` is not in the classpath, or if CapDump returns an error code.

Examples

Run CapDump to dump the contents of the `test.cap` file.

```
<target name="CapDumpTarget" >
  <capdump
       CapFile="${samples.output}/test.cap"
  </capdump>
</target>
```

Run CapDump to dump the contents of the `test.cap` file. Classpath used in this example is referenced by the `classpath` refid

```
<target name="CapDumpTarget" >
  <capdump
       CapFile="${samples.output}/test.cap"
       <classpath refid="classpath"/>
  </capdump>
</target>
```

Capgen

Runs Capgen to generate a CAP file from a JCA file.

TABLE 3-2 Parameters for Capgen

Attribute	Description	Required
JCAFile	Fully qualified path and name of the input JCA file.	Yes
OutFile	Fully qualified path and name of the output CAP file.	No
classpath	Classpath to use for this task. If required JAR files are not already in the system classpath, you can specify this attribute to put them in the classpath when this task is executed.	No
dir	The directory to invoke the Java VM in.	No
nobanner	Set this element to `true` if you do not want the Capgen banner showing.	No
version	Prints Capgen version number.	No

Errors

Execution of this task fails if any of the required elements are not supplied, if `converter.jar` is not in the classpath, or if Capgen returns an error code.

Examples

Run Capgen to generate the `mathDemo.cap` file from the `mathDemo.jca` file.

```
<target name="CapgenTarget" >
   <capgen
         JCAFile="${sample.output}/mathDemo.jca"
         outfile="${sample.output}/mathDemo.cap">
         <classpath refid="classpath"/>
   </capgen>
</target>
```

Run Capgen to generate a `mathDemo.cap` file from the `mathDemo.jca` file. Classpath used in this example is referenced by the `classpath` refid.

```
<target name="CapgenTarget" >
   <capgen
         JCAFile="${sample.output}/mathDemo.jca"
         outfile="${sample.output}/mathDemo.cap">
```

```
        <classpath refid="classpath"/>
   </capgen>
</target>
```

The following example is the same as the previous example, except no output file is specified. Capgen generates `out.cap` in the directory in which the Java VM was invoked.

```
<target name="CapgenTarget" >
   <capgen
        JCAFile="${sample.output}/mathDemo.jca"/>
        <classpath refid="classpath"/>
   </capgen>
</target>
```

Converter

Runs Converter to generate CAP, EXP and JCA files from a Java technology-based package. By default the Java Card platform converter creates CAP and EXP files for the input package. However, if any one of the CAP, JCA or EXP flags are enabled, only the output files enabled are generated.

TABLE 3-3 Parameters for Converter

Attribute	Description	Required
PackageName	Fully qualified name of the package being converted.	Yes
PackageAID	AID of the package being converted.	Yes
MajorMinorVersion	Major and Minor version numbers of the package, for example, 1.2 (where 1 is major version number and 2 is minor version number).	Yes
CAP	If enabled tells the converter to create a CAP file.	No
EXP	If enabled tells the converter to create a EXP file.	No
JCA	If enabled tells the converter to create a JCA file.	No
ClassDir	The root directory of the class hierarchy. Specifies the directory where the converter will look for class files.	No
Int	If enabled turns on support the 32-bit integer type.	No
Debug	If enabled, enables generation of debugging information.	No
ExportPath	Root directories where the Converter will look for export files.	No
ExportMap	If enabled, tells the converter to use the token mapping from the pre-defined export file of the package being converted. The converter will look for the export file in the exportpath.	No
Outputdirectory	Sets the output directory where the output files will be placed.	No
Verbose	If enabled, enables verbose converter output.	No
noWarn	If enabled instructs the Converter to not report warning messages.	No
Mask	If enabled tells the Converter that this package is for mask, so restrictions on native methods are relaxed.	No
NoVerify	If enabled tells the Converter to turn off verification. Verification is turned on by default.	No

TABLE 3-3 Parameters for Converter

Attribute	Description	Required
classpath	Classpath to use for this task. If required JAR files are not already in the system classpath, you can specify this attribute to put them in the classpath when this task is executed.	No
dir	The directory to invoke the Java VM in.	No
nobanner	Set this element to `true` if you do not want the Capgen banner showing.	No
version	Prints Converter version number.	No

Parameters Specified As Nested Elements

AppletNameAID

Use nested element AppletNameAID to specify names and AIDs of applets belonging to the package being converted. For details regarding AppletNameAID type, see "AppletNameAID" on page 31.

Errors

Execution of this task fails if any of the required elements are not supplied, if `converter.jar` or `offcardverifier.jar` are not in the classpath, or if Converter returns an error code.

Examples

Run Converter to generate `helloworld.cap`, `helloworld.JCA` and `helloworld.EXP` files.

```
<target name="convert_HelloWorld.cap" >
   <convert
         JCA="true"
         EXP="true"
         CAP="true"
         packagename="com.sun.javacard.samples.HelloWorld"
         packageaid="0xa0:0x0:0x0:0x0:0x62:0x3:0x1:0xc:0x1"
         majorminorversion="1.0"
```

```
        classdir="${classroot}"
        outputdirectory="${classroot}">
        <AppletNameAID
            appletname="com.sun.javacard.samples.HelloWorld.HelloWorld"
            aid="0xa0:0x0:0x0:0x0:0x62:0x3:0x1:0xc:0x1:0x1"/>
        <exportpath refid="export"/>
        <classpath refid="classpath"/>
    </convert>
  </target>
```

In the following example the converter options are specified in helloworld.cfg file instead of being specified in the target itself. This example also shows how a classpath can be specified for a target and how a directory can be set in which the Java VM is invoked for the converter task.

```
<target name="convert_HelloWorld" >
  <convert
      dir="${samples}"
      Configfile="${samples.configDir}/helloworld.cfg">
      <classpath>
          <pathelement path="${samples}"/>
          <fileset dir="${lib}">
              <include name="**/converter.jar"/>
              <include name="**/offcardverifier.jar"/>
          </fileset>
      </classpath>
  </convert>
</target>
```

DeployCap

This task sends a CAP file to `cref` and hides the complexities of creating a script file, running `cref` and then running APDUTool to send the script to `cref`. The resulting EEPROM image is saved in the specified output file. This task automatically checks if installation was successful or not by checking status words returned by `cref`.

TABLE 3-4 Parameters for DeployCap

Attribute	Description	Required
CapFile	Fully qualified path and name of the CAP file which is to be sent to `cref`.	Yes
CrefExe	Fully qualified path and name of `cref` executable.	Yes
OutEEFile	Output EEPROM file that will contain the EEPROM image after `cref` finishes execution.	Yes
InEEFile	Input EEPROM file for `cref`. If specified `cref` initiates using the EEPROM image stored in this file.	No
classpath	Classpath to use for this task. If required JAR files are not already in the system classpath, you can specify this attribute to put them in the classpath when this task is executed.	No
dir	The directory to invoke the Java VM in.	No
nobanner	Set this element to `true` if you do not want the tool banner showing.	No

Errors and Return Codes

Execution of this task fails if any of the required elements are not supplied, if `apdutool.jar`, `apduio.jar` and `scriptgen.jar` are not in the classpath, or if APDUTool, Scriptgen or `cref` fail to execute.

Examples

The following example installs `helloworld.cap` file in `cref`. By default it is checked if the APDU commands were sent correctly. Classpath used in the above example is referenced by the `classpath` refid.

```
<target name="Deploy_Hello_world_CAP" >
  <deploycap
      CAPFile="${samples.output}/helloworld.cap"
      outEEFile="${samples.eeprom}/outEEFile"
      CrefExe="{JAVACARD_HOME}/bin/cref">
      <classpath refid="classpath"/>
  </deploycap>
</target>
```

The following example installs `helloworld.cap` file in `cref`, which in this case will be initialized with EEFile. The `cref` output EEPROM image will also be saved in the same EEFile. By default it is checked if the APDU commands were sent correctly. This example shows that the resulting EEPROM image can be stored in the same EEPROM image file that was used to initialize `cref`.

```
<target name="Deploy_Hello_world_CAP" >
  <deploycap
      CAPFile="${samples.output}/helloworld.cap"
      outEEFile="${samples.eeprom}/EEFile"
      inEEFile="${samples.eeprom}/EEFile"
      CrefExe="{JAVACARD_HOME}/bin/cref">
      <classpath refid="classpath"/>
  </deploycap>
</target>
```

Exp2Text

Run Exp2Text tool to convert the export file of a package to a text file.

TABLE 3-5 Parameters for Exp2Text

Attribute	Description	Required
PackageName	Fully qualified name of the package.	Yes
ClassDir	Root directory where the exp2text tool will look for the export file. If no ClassDir is specified, the directory in which the Java VM is invoked is taken as base dir.	No
OutputDir	The root directory for output.	No
classpath	Classpath to use for this task. If required JAR files are not already in the system classpath, you can specify this attribute to put them in the classpath when this task is executed.	No
dir	The directory to invoke the Java VM in.	No
nobanner	Set this element to `true` if you do not want the Exp2Text banner showing.	No
version	Prints Exp2Text version number.	No

Errors

Execution of this task fails if any of the required elements are not supplied, if `converter.jar` is not in the classpath, or if `Exp2Text` returns an error code.

Examples

Run `Exp2Text` to generate text file from the export file of package HelloWorld. This example assumes that `converter.jar` is already in classpath.

```
<target name="Exp2TextTarget" >
  <Exp2Text
       packagename="com.sun.javacard.samples.HelloWorld"
       classdir="${classroot}"
       outputdir="${classroot}">
  </Exp2Text>
</target>
```

Run Exp2Text to generate text file from the export file of package HelloWorld. `Classdir` and the root `outputdir` are both assumed to be the directory where the Java VM was invoked. Classpath used in this example is referenced by the `classpath` refid.

```
<target name="Exp2TextTarget" >
  <Exp2Text
       packagename="com.sun.javacard.samples.HelloWorld">
       <classpath refid="classpath"/>
  </Exp2Text>
</target>
```

Maskgen

Runs Maskgen to generate a mask for `cref`, depending on the generator used (see details below).

TABLE 3-6 Parameters for Maskgen

Attribute	Description	Required
Generator	Tells Maskgen for which platform is the mask to be generated. Possible choices are a51, `cref`, and size. For details see Maskgen documentation in the *User's Guide for the Java Card Platform, Version 2.2.2* (Sun Microsystems, Inc., 2006).	Yes
ConfigFile	Fully qualified path and name of generator specific configuration file.	No
DebugInfo	If enabled, tells Maskgen to generate location debug information for mask.	No
MemRefSize	Integer value that tells Maskgen what memory reference size to use in the mask. Two possible values for element are 16 and 32. Default value used by Maskgen is 32.	No
OutFile	Fully qualified path and name of the output mask file. If this element is not specified, default file name is `a.out` which will be generated in the directory where the Java VM is invoked.	No
classpath	Classpath to use for this task. If required JAR files are not already in the system classpath, you can specify this attribute to put them in the classpath when this task is executed.	No
dir	The directory to invoke the Java VM in.	No
nobanner	Set this element to `true` if you do not want the Capgen banner showing.	No
version	Prints Capgen version number.	No

Parameters Specified As Nested Elements

JCAInputFile

Use nested element JCAInputFile to specify names of input JCA files for Maskgen. Input JCA files are required to create a Mask file. The reason a standard FileSet to specify JCA file names is not used here is that Maskgen supports input file names that starts with an "@" symbol to specify an input file that contains a list of names of input JCA files. A file name that starts with "@" is not supported by any of the standard Ant types. See description for "JCAInputFile" on page 32 for details.

Errors

Execution of this task fails if any of the required elements are not supplied, if `converter.jar` is not in the classpath or if `Maskgen` returns an error code.

Examples

Run Maskgen to generate `mask.c` file from input JCA files specified in files `mask1.in` and `mask2.in`.

```
<target name="MasgenTarget" >
   <maskgen
        generator="cref"
        configfile="${maskDir}/mask.cfg"
        outfile="${crefDir}/common/mask.c" >
        <jcainputfile inputfile="@${maskDir}/mask1.in" / >
        <jcainputfile inputfile="@${maskDir}/mask2.in" / >
   </maskgen >
</target >
```

Run Maskgen to generate `mask.c` file from input JCA files specified in files `api.in` and installer and `helloworld` JCA files.

```
<target name="MasgenTarget" >
   <maskgen
        generator="cref"
        configfile="${maskDir}/mask.cfg"
        outfile="${crefDir}/common/mask.c" >
        <jcainputfile inputfile="@${maskDir}/api.in" / >
        <jcainputfile inputfile="${jcaDir}/installer.jca" / >
        <jcainputfile inputfile="${jcaDir}/helloworld.jca" / >
   </maskgen >
</target >
```

The following example is the same as the previous example, except no output file is specified and classpath is specified. Maskgen will generate the file `a.out` in the directory in which Java VM was invoked.

```
<target name="MasgenTarget" >
   <maskgen
        generator="cref"
        configfile="${maskDir}/mask.cfg"
        outfile="${crefDir}/common/mask.c" >
        <jcainputfile inputfile="@${maskDir}/api.in" / >
        <jcainputfile inputfile="${jcaDir}/installer.jca" / >
        <jcainputfile inputfile="${jcaDir}/helloworld.jca" / >
        <classpath refid="classpath"/>
   </maskgen >
</target >
```

Scriptgen

Runs Scriptgen to generate an APDU script file from a CAP file.

TABLE 3-7 Parameters for Scriptgen

Attribute	Description	Required
CapFile	Fully qualified path and name of the input CAP file.	Yes
OutFile	Fully qualified path and name of the output script file. If no output file name is specified, generated script will be output on the console.	No
PkgName	Fully qualified name of the package inside the CAP file.	No
NoBeginEnd	If enabled, instructs Scriptgen to suppress "CAP_BEGIN", "CAP_END" APDU commands.	No
classpath	Classpath to use for this task. If required JAR files are not already in the system classpath, you can specify this attribute to put them in the classpath when this task is executed.	No
dir	The directory to invoke the Java VM in.	No
nobanner	Set this element to `true` if you do not want the Scriptgen banner showing.	No
version	Prints Scriptgen version number.	No

Errors

Execution of this task fails if any of the required elements are not supplied, if `scriptgen.jar` is not in the classpath or if Scriptgen returns an error code.

Examples

Run Scriptgen to generate script file `helloWorld.scr` from `helloWorld.cap` file. Classpath used in this example is referenced by the `classpath` refid.

```
<target name="ScriptgenTarget" >
  <scriptgen
       noBeginEnd="true"
       noBanner="true"
       CapFile="${samples.helloworld.output}/HelloWorld.cap"
       outFile="${samples.helloworld.script}/helloWorld.scr" >
```

```
        <classpath refid="classpath" />
    </scriptgen >
</target >
```

VerifyCap

Runs off-card Java Card platform CAP file verifier to verify a CAP file. The Java Card platform off-card verifier is invoked in a separate instance of Java VM.

TABLE 3-8 Parameters for VerifyCap

Attribute	Description	Required
CapFile	Fully qualified path and name of CAP file that is to be verified.	Yes
PkgName	Fully qualified Name of the package inside the CAP file for which the CAP file was generated.	No
noWarn	If enabled, tells the verifier not to output any warning messages.	No
Verbose	If enabled, enables verbose converter output.	No
classpath	Classpath to use for this task. If required JAR files are not already in the system classpath, you can specify this attribute to put them in the classpath when this task is executed.	No
dir	The directory to invoke the Java VM in.	No
nobanner	Set this element to `true` if you want to suppress Verifier banner.	No
version	Prints the version number of the off-card verifier.	No

Parameters Specified As Nested Elements

ExportFiles

Use nested element ExportFiles to specify group of export files for packages imported by the package whose CAP file is being verified and the export file corresponding to the CAP being verified. For details regarding ExportFiles type see "ExportFiles" on page 32.

Errors

Execution of this task fails if any of the required elements are not supplied, if `offcardverifier.jar` is not in the classpath, or if Verifier returns an error code.

Examples

Run the Java Card platform off-card verifier to verify `HelloWorld.cap` file.

```
<target name="VerifyCapTarget" >
  <verifycap
       CapFile="${samples.helloworld.output}/HelloWorld.cap" >
       <exportfiles file="${samples.helloworld.output}/HelloWorld.exp" />
       <exportfiles file="${api_exports}/javacard/framework/javacard/framework.exp" />
       <exportfiles file="${api_exports}/java/lang/javacard/lang.exp" />
       <classpath refid="classpath"/>
  </verifycap>
</target>
```

VerifyExp

Runs off-card Java Card platform EXP file verifier to verify an EXP file. Java Card platform off-card verifier is invoked in a separate instance of Java VM.

TABLE 3-9 Parameters for VerifyExp

Attribute	Description	Required
noWarn	If enabled, tells the verifier not to output any warning messages.	No
Verbose	If enabled, enables verbose converter output.	No
classpath	Classpath to use for this task. If required JAR files are not already in the system classpath, you can specify this attribute to put them in the classpath when this task is executed.	No
dir	The directory to invoke the Java VM in.	No
nobanner	Set this element to `true` if you want to suppress Verifier banner.	No
version	Prints the version number of off-card verifier.	No

Parameters Specified As Nested Elements

ExportFiles

Use nested element ExportFiles to specify the EXP file being verified. For details regarding ExportFiles type see "ExportFiles" on page 32. VerifiyExp requires that only one input EXP file be specified. This tasks throws an error if more than one EXP files are specified.

Errors

Execution of this task fails if no EXP file is specified or if more than one EXP file is specified, if `offcardverifier.jar` is not in the classpath, or if Verifier returns an error code.

Examples

Run the Java Card platform off-card verifier to verify `HelloWorld.exp` file.

```
<target name="VerifyExpTarget" >
  <verifyExp
      <exportfiles file="${samples.helloworld.output}/HelloWorld.exp" />
      <classpath refid="classpath"/>
  </verifyExp>
</target>
```

VerifyRev

Runs off-card Java Card platform verifier to verify binary compatibility between two versions of an EXP file. Java Card platform off-card verifier is invoked in a separate instance of Java VM.

TABLE 3-10 Parameters for VerifyRev

Attribute	Description	Required
noWarn	If enabled, tells the verifier not to output any warning messages.	No
Verbose	If enabled, enables verbose converter output.	No
classpath	Classpath to use for this task. If required jar files are not already in the system classpath, you can specify this attribute to put them in the classpath when this task is executed.	No
dir	The directory to invoke the Java VM in.	No
nobanner	Set this element to `true` if you want to suppress Verifier banner.	No
version	Prints the version number of off-card verifier.	No

Parameters Specified As Nested Elements

ExportFiles

Use nested element ExportFiles to specify the EXP files being verified. For details regarding ExportFiles type see "ExportFiles" on page 32. VerifiyExp requires that exactly two input EXP files are specified. This tasks throws an error if more or less than two EXP files are specified.

Errors

Execution of this task fails if no EXP file is specified or if less or more than two EXP files are specified, if `offcardverifier.jar` is not in the classpath, or if Verifier returns an error code.

Examples

Run the Java Card platform off-card verifier to verify binary compatibility between two versions of `HelloWorld.exp` file.

```
<target name="VerifyExpTarget" >
  <verifyExp
       <exportfiles file="${samples.helloworld.output}/HelloWorld.exp" />
       <exportfiles file="${samples.helloworld.output.new}/HelloWorld.exp" />
       <classpath refid="classpath"/>
  </verifyExp>
</target>
```

CHAPTER 4

Custom Types

The custom types available for the Ant tasks are described in this chapter.

AppletNameAID

`AppletNameAID` groups together name and AID for a Java Card applet.

TABLE 4-1 Parameters for ApplietNameAID

Attribute	Description	Required
appletname	Fully qualified name of the Java Card applet.	Yes
aid	AID (Application Identifier) of the Java Card applet.	Yes

Example

Set the fully qualified name and AID for the `HelloWorld` applet.

```
<AppletNameAID
appletname="com.sun.javacard.samples.HelloWorld.HelloWorld"
aid="0xa0:0x0:0x0:0x0:0x62:0x3:0x1:0xc:0x1:0x1"/>
```

JCAInputFile

This type is a simple wrapper for a fully qualified JCA file name or a name of an input file that contains a list of input JCA files. In case the input file contains a list of input JCA files, the name of the file should be prepended with "@".

TABLE 4-2 Parameters for JCAInputFile

Attribute	Description	Required
inputfile	Fully qualified name of the input file	Yes

Examples

Set the fully qualified name of an input JCA file.

```
<jcainputfile
   inputfile="C:\jcas\common\com\sun\javacard\installer
\javacard\installer.jca" />
```

Set the fully qualified name of an input file that contains a list of JCA files.

```
<jcainputfile inputfile="@C:\jc\mathDemo.in" />
```

ExportFiles

This type is actually the Ant FileSet type. It is used to specify a group of export files for the off-card verifier. For details, see Apache Ant documentation for FileSet type.

Examples

The following example sets the fully qualified name of an input EXP file.

```
<exportfiles
   file="C:\samples\classes\com\sun\javacard\samples
\HelloWorld\javacard\HelloWorld.exp"
```

The following example groups all the files in the directory `${server.src}` that are EXP files and do not have the text `Test` in their names.

```
<exportfiles dir="${server.src}">
```

```
  <include name="**/*.exp"/>
  <exclude name="**/*Test*"/>
</exportfiles>
```

CHAPTER **5**

NetBeans™ Software Integration

NetBeans™ integrated development environmnet 4.1 creates Ant script for Java technology projects and uses this script to compile, build and run projects. The `build.xml` file for a NetBeans project is located in the project's root directory. Modify this `build.xml` file as explained in Chapter 2 to enable usage of the Ant tasks.

Netbeans software recognizes some pre-defined targets in the `build.xml` file that is generated for the project. Some of these targets exist, such as `clean` and `compile`, while XML for others is left empty. One of these targets is `-post-compile`. You may safely modify this target to incorporate usage of the Ant tasks in your project. An example of a modified `build.xml` file for a NetBeans project is included here. For information on NetBeans software, go to the NetBeans integrated development environment web site at `www.netbeans.org`.

CODE EXAMPLE 5-1 `build.xml` Modified for a NetBeans Project

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- You may freely edit this file. See commented blocks below for -->
<!-- some examples of how to customize the build. -->
<!-- (If you delete it and reopen the project it will be recreated.) -->
<project name="JavaPurse" default="default" basedir=".">
     <description>Builds, tests, and runs the project JavaPurse.</description>
     <import file="nbproject/build-impl.xml"/>
     <!--

     There exist several targets which are by default empty and which can be
     used for execution of your tasks. These targets are usually executed
     before and after some main targets. They are:

       -pre-init:                 called before initialization of project properties
       -post-init:                called after initialization of project properties
       -pre-compile:              called before javac compilation
       -post-compile:             called after javac compilation
       -pre-compile-single:       called before javac compilation of single file
       -post-compile-single:      called after javac compilation of single file
       -pre-compile-test:         called before javac compilation of JUnit tests
       -post-compile-test:        called after javac compilation of JUnit tests
       -pre-compile-test-single:  called before javac compilation of single JUnit test
```

CODE EXAMPLE 5-1 `build.xml` Modified for a NetBeans Project *(Continued)*

```
      -post-compile-test-single: called after javac compilation of single JUunit test
      -pre-jar:                  called before JAR building
      -post-jar:                 called after JAR building
      -post-clean:               called after cleaning build products

    (Targets beginning with '-' are not intended to be called on their own.)

    Example of inserting an obfuscator after compilation could look like this:

        <target name="-post-compile">
            <obfuscate>
                <fileset dir="${build.classes.dir}"/>
            </obfuscate>
        </target>

    For list of available properties check the imported
    nbproject/build-impl.xml file.


    Another way to customize the build is by overriding existing main targets.
    The targets of interest are:

      -init-macrodef-javac:      defines macro for javac compilation
      -init-macrodef-junit:      defines macro for junit execution
      -init-macrodef-debug:      defines macro for class debugging
      -init-macrodef-java:       defines macro for class execution
      -do-jar-with-manifest:     JAR building (if you are using a manifest)
      -do-jar-without-manifest: JAR building (if you are not using a manifest)
      run:                       execution of project
      -javadoc-build:            Javadoc generation
      test-report:               JUnit report generation

    An example of overriding the target for project execution could look like this:

        <target name="run" depends="JavaPurse-impl.jar">
            <exec dir="bin" executable="launcher.exe">
                <arg file="${dist.jar}"/>
            </exec>
        </target>

    Notice that the overridden target depends on the jar target and not only on
    the compile target as the regular run target does. Again, for a list of available
    properties which you can use, check the target you are overriding in the
    nbproject/build-impl.xml file.

    -->
    <!-- the directory structure under the base directory -->
    <property name="sourceroot" value="src" />
    <property name="classroot" value="build/classes" />
    <property name="exportmap" value="exportmap" />

    <!-- the directories where the Java Card(TM) export are located -->
     <!-- could go into a properties file-->
    <property name="jcardkit_home" value="C:\java_card_kit-2_2_2" />
    <property name="jcardkit_exports" value="${jcardkit_home}/api_export_files" />
```

CODE EXAMPLE 5-1 `build.xml` Modified for a NetBeans Project *(Continued)*

```
<property name="jcardkit_libs" value="${jcardkit_home}/lib" />

<!-- Definitions for tasks for Java Card tools -->
<taskdef name="apdutool"
     classname="com.sun.javacard.ant.tasks.APDUToolTask"/>
<taskdef name="capgen"
     classname="com.sun.javacard.ant.tasks.CapgenTask" />
<taskdef name="maskgen"
     classname="com.sun.javacard.ant.tasks.MaskgenTask" />
<taskdef name="deploycap"
     classname="com.sun.javacard.ant.tasks.DeployCapTask" />
<taskdef name="exp2text"
     classname="com.sun.javacard.ant.tasks.Exp2TextTask" />
<taskdef name="convert"
     classname="com.sun.javacard.ant.tasks.ConverterTask" />
<taskdef name="verifyexport"
     classname="com.sun.javacard.ant.tasks.VerifyExpTask" />
<taskdef name="verifycap"
     classname="com.sun.javacard.ant.tasks.VerifyCapTask" />
<taskdef name="verifyrevision"
     classname="com.sun.javacard.ant.tasks.VerifyRevTask" />
<taskdef name="scriptgen"
     classname="com.sun.javacard.ant.tasks.ScriptgenTask" />
<typedef name="appletnameaid"
     classname="com.sun.javacard.ant.types.AppletNameAID" />
<typedef name="jcainputfile"
     classname="com.sun.javacard.ant.types.JCAInputFile" />
<typedef name="exportfiles"
     classname="org.apache.tools.ant.types.FileSet" />

<!-- set the export path to the Java Card export files -->
<path id="export" description="set the export file path">
     <pathelement path="${jcardkit_exports}" />
     <pathelement path="build/classes"/>
</path>

     <!-- set the classpath at minimum to the Java Card API -->
     <!-- but also for all other API needed in the project-->
<path id="classpath" description="Sets the classpath to Java Card API and tools">
     <pathelement path="${jcardkit_home}/lib/api.jar"/>
     <pathelement path="${jcardkit_home}/lib/converter.jar"/>
     <pathelement path="${jcardkit_home}/lib/offcardverifier.jar"/>
     <pathelement path="${jcardkit_home}/lib/scriptgen.jar"/>
     <pathelement path="${jcardkit_home}/lib/apdutool.jar"/>
     <pathelement path="${jcardkit_home}/lib/apduio.jar"/>
     <pathelement path="."/>
</path>

<target name="convert_library"
 description="Build export file and CAP file for SampleLibrary">
     <convert
          EXP="true"
          CAP="true"
          packagename="com.sun.javacard.samples.SampleLibrary"
          packageaid="0xA0:0x0:0x0:0x0:0x62:0x3:0x1:0xC:0x4"
```

CODE EXAMPLE 5-1 `build.xml` Modified for a NetBeans Project *(Continued)*

```
              majorminorversion="1.0"
              classdir="${classroot}"
              outputdirectory="${classroot}">
              <exportpath refid="export"/>
              <classpath refid="classpath"/>
          </convert>
     </target>

     <target name="convert_purse" depends="convert_library"
      description="Build cap file for the JavaPurse package">
              <convert
               CAP="true"
               packagename="com.sun.javacard.samples.JavaPurse"
               packageaid="0xA0:0x0:0x0:0x0:0x62:0x3:0x1:0xC:0x2"
               majorminorversion="1.0"
               classdir="${classroot}"
               outputdirectory="${classroot}">
               <AppletNameAID
                   appletname="com.sun.javacard.samples.JavaPurse.JavaPurse"
                   aid="0xa0:0x0:0x0:0x0:0x62:0x3:0x1:0xc:0x2:0x1"/>
               <exportpath refid="export"/>
               <classpath refid="classpath"/>
          </convert>
      </target>

     <target name="-post-compile" depends="convert_purse">
     </target>
</project>
```

CHAPTER 6

Ant Tasks Example

Following is a complete `build.xml` file that can be used to compile, convert, verify, and deploy the JavaPurse demo applet that comes with Java Card development kit in `cref` and then run a demo script.

To use this example, copy and paste CODE EXAMPLE 6-1 in `build.xml` file. Place the `build.xml` file in <*Java Card Dev Kit Directory*>`/samples/src` directory. Create a subdirectory in `src` directory named `scripts` and place `testpurse.scr` (CODE EXAMPLE 6-2) in that directory. At this point you should be able to run Ant in the `src` directory to run the example.

CODE EXAMPLE 6-1 `build.xml`

```
<?xml version="1.0"?>
<project name="Java Card(TM) Tasks Sample" default="all" basedir=".">

  <!-- the directory structure under the base directory -->
  <property name="sourceroot" value="." />
  <property name="classroot" value="classes" />
  <property name="samples.eeprom" value="eeprom" />
  <property name="samples.scripts" value="scripts" />
  <property name="exportmap" value="exportmap" />

  <!-- the directories where the Java Card(TM) export are located could go -->
  <!-- into a properties file-->
  <property name="jcardkit_home" value="../.." />
  <property name="jcardkit_exports" value="${jcardkit_home}/api_export_files" />
  <property name="jcardkit_libs" value="${jcardkit_home}/lib" />

  <!-- Definitions for tasks for Java Card tools -->
  <taskdef name="apdutool"
    classname="com.sun.javacard.ant.tasks.APDUToolTask" />
  <taskdef name="capgen"
    classname="com.sun.javacard.ant.tasks.CapgenTask" />
  <taskdef name="maskgen"
    classname="com.sun.javacard.ant.tasks.MaskgenTask" />
  <taskdef name="deploycap"
    classname="com.sun.javacard.ant.tasks.DeployCapTask" />
  <taskdef name="exp2text"
    classname="com.sun.javacard.ant.tasks.Exp2TextTask" />
```

CODE EXAMPLE 6-1 `build.xml` *(Continued)*

```
<taskdef name="convert"
  classname="com.sun.javacard.ant.tasks.ConverterTask" />
<taskdef name="verifyexport"
  classname="com.sun.javacard.ant.tasks.VerifyExpTask" />
<taskdef name="verifycap"
  classname="com.sun.javacard.ant.tasks.VerifyCapTask" />
<taskdef name="verifyrevision"
  classname="com.sun.javacard.ant.tasks.VerifyRevTask" />
<taskdef name="scriptgen"
  classname="com.sun.javacard.ant.tasks.ScriptgenTask" />
<typedef name="appletnameaid"
  classname="com.sun.javacard.ant.types.AppletNameAID" />
<typedef name="jcainputfile"
  classname="com.sun.javacard.ant.types.JCAInputFile" />
<typedef name="exportfiles"
  classname="org.apache.tools.ant.types.FileSet" />

<!-- set the classpath at minimum to the Java Card API but also for -->
<!-- all other APIs needed in the project-->
<path id="classpath" description="Sets the classpath to Java Card API and tools">
     <pathelement path="${jcardkit_home}/lib/api.jar"/>
     <pathelement path="${jcardkit_home}/lib/converter.jar"/>
     <pathelement path="${jcardkit_home}/lib/offcardverifier.jar"/>
     <pathelement path="${jcardkit_home}/lib/scriptgen.jar"/>
     <pathelement path="${jcardkit_home}/lib/apdutool.jar"/>
     <pathelement path="${jcardkit_home}/lib/apduio.jar"/>
     <pathelement path="."/>
</path>

<!-- set the export path to the Java Card export files -->
<path id="export" description="set the export file path">
     <pathelement path="${jcardkit_exports}" />
     <pathelement path="./classes"/>
</path>

<!-- compile section -->
<target name="compile_sources" description="Build classes">
     <!-- Make destination directory -->
     <mkdir dir="${classroot}"/>
     <!-- Compile the java code from ${src} to ${classes} -->
     <javac debug="yes"
       optimize="no"
       srcdir="${sourceroot}/com/sun/javacard/samples/SampleLibrary"
       destdir="${classroot}">
       <classpath refid="classpath"/>
     </javac>
     <javac debug="yes"
       optimize="no"
       srcdir="${sourceroot}/com/sun/javacard/samples/JavaPurse"
       destdir="${classroot}">
     <classpath refid="classpath"/>
     </javac>
</target>

<!-- Convert SampleLibrary before converting JavaPurse because JavaPurse -->
```

CODE EXAMPLE 6-1 `build.xml` *(Continued)*

```
<!-- imports SampleLibrary -->
<target name="convert_library" depends="compile_sources"
        description="Build export file and CAP file for SampleLibrary">
     <convert
       EXP="true"
       CAP="true"
       packagename="com.sun.javacard.samples.SampleLibrary"
       packageaid="0xA0:0x0:0x0:0x0:0x62:0x3:0x1:0xC:0x4"
       majorminorversion="1.0"
       classdir="${classroot}"
       outputdirectory="${classroot}">
        <exportpath refid="export"/>
       <classpath refid="classpath"/>
     </convert>
</target>

<!-- Convert JavaPurse -->
<target name="convert_purse" depends="convert_library"
        description="Build cap file for the JavaPurse package">
    <convert
        CAP="true"
        packagename="com.sun.javacard.samples.JavaPurse"
        packageaid="0xA0:0x0:0x0:0x0:0x62:0x3:0x1:0xC:0x2"
        majorminorversion="1.0"
        classdir="${classroot}"
        outputdirectory="${classroot}">
     <AppletNameAID
            appletname="com.sun.javacard.samples.JavaPurse.JavaPurse"
            aid="0xa0:0x0:0x0:0x0:0x62:0x3:0x1:0xc:0x2:0x1"/>
     <exportpath refid="export"/>
     <classpath refid="classpath"/>
    </convert>
 </target>

<!-- Verify CAP file for JavaPurse -->
<target name="verify_purse" depends="convert_purse" >
  <verifycap
       CapFile="${classroot}/com/sun/javacard/samples/JavaPurse/javacard/JavaPurse.cap" >
     <exportfiles file="${classroot}/com/sun/javacard/samples/SampleLibrary/
                       javacard/SampleLibrary.exp" />
     <exportfiles dir="${jcardkit_exports}">
       <include name="**/javacard/framework/javacard/framework.exp"/>
       <include name="**/lang.exp"/>
     </exportfiles>
    <classpath refid="classpath"/>
   </verifycap>
</target>

<!-- Deploy SampleLibrary before deploying JavaPurse because -->
<!-- JavaPurse depends on SampleLibrary -->
<target name="deploy_sample_library" depends="convert_library" >
  <!-- Make EEPROM directory -->
  <mkdir dir="${samples.eeprom}"/>
  <deploycap
     outEEFile="${samples.eeprom}/EEFile"
```

CODE EXAMPLE 6-1 `build.xml` *(Continued)*

```
       CrefExe="${jcardkit_home}/bin/cref.exe"
       CapFile="${classroot}/com/sun/javacard/samples/SampleLibrary/javacard/SampleLibrary.cap" >
       <classpath refid="classpath"/>
    </deploycap>
  </target>

  <!-- Deploy JavaPurse using the resulting EEPROM image from deploying SampleLibrary -->
  <target name="deploy_java_purse" depends="convert_purse, deploy_sample_library" >
    <deploycap
       inEEFile="${samples.eeprom}/EEFile"
       outEEFile="${samples.eeprom}/EEFile"
       CrefExe="${jcardkit_home}/bin/cref.exe"
       CapFile="${classroot}/com/sun/javacard/samples/JavaPurse/javacard/JavaPurse.cap" >
       <classpath refid="classpath"/>
    </deploycap>
  </target>

  <!-- Now that JavaPurse is deployed we can run our test script -->
  <target name="run_test_script" depends="deploy_java_purse" >
    <apdutool
       scriptFile="${samples.scripts}/testpurse.scr"
       inEEFile="${samples.eeprom}/EEFile"
       outEEFile="${samples.eeprom}/EEFile"
       CheckDownloadFailure="false"
       CrefExe="${jcardkit_home}/bin/cref.exe">
       <classpath refid="classpath"/>
    </apdutool>
  </target>

  <!-- Clean output directories -->
  <target name="clean">
    <delete dir="${classroot}" />
    <delete dir="${samples.eeprom}" />
  </target>

  <!-- Clean, compile, convert, verify, deploy cap files and run test script -->
  <target name="all" depends="clean, run_test_script" />

</project>
```

The following code example is for `testpurse.scr`.

CODE EXAMPLE 6-2 `testpurse.scr`

```
//+
// Copyright 2005 Sun Microsystems, Inc. All rights reserved.
//-

powerup;

echo "****Select the installer applet";
0x00 0xA4 0x04 0x00 0x09 0xa0 0x00 0x00 0x00 0x62 0x03 0x01 0x08 0x01 0x7F;
// 90 00 = SW_NO_ERROR

echo "****create JavaPurse";
0x80 0xB8 0x00 0x00 0x0c 0x0a 0xa0 0x00 0x00 0x00 0x62 0x03 0x01 0x0c 0x2 0x01 0x00 0x7F;
```

CODE EXAMPLE 6-2 `testpurse.scr` *(Continued)*

```
///////////////////////////////////////////////////////////////////
// Initialize JavaPurse
///////////////////////////////////////////////////////////////////

echo "****Select JavaPurse";
0x00 0xa4 0x04 0x00 10 0xa0 0 0 0 0x62 3 1 0xc 2 1 127;
// 90 00 = SW_NO_ERROR

echo "****Initialize Parameter Update";
0x80 0x24 0x00 0x00 0x00 0x7F;
//00 00 00 00 0c 1f 63 00 01 90 00 = Purse ID : 0x00000000; ExpDate 12/31/99; PUN 1
//For the second and consecutive runs it can be 69 82

echo "****Complete Parameter Update: CAD ID 0x11223344; Set Master PIN 12345678";
0x80 0x26 0x00 0x00 0x1A 0x11 0x22 0x33 0x44 0x00 0x00 0x00 0x00 0xC1 0x08 0x01 0x02 0x03
0x04 0x05 0x06 0x07 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x7F;
// 00 00 00 00 00 00 00 00 90 00
// For second and consecutive runs it can be 91 04

echo "****Verify PIN : Master PIN";
0x00 0x20 0x00 0x81 0x08 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x7F;
// 90 00;

echo "****Initialize Parameter Update";
0x80 0x24 0x00 0x00 0x00 0x7F;
// 00 00 00 00 0c 1f 63 00 02 90 00 = Purse ID : 0x00000000; ExpDate 12/31/99; PUN 2

echo "****Complete Parameter Update: CAD ID 0x11223344; Set User PIN 1234";
0x80 0x26 0x00 0x00 0x16 0x11 0x22 0x33 0x44 0x00 0x00 0x00 0x00 0xC2 0x04 0x01 0x02 0x03
0x04 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x7F;
// 00 00 00 00 00 00 00 00 90 00;

echo "****Initialize Parameter Update";
0x80 0x24 0x00 0x00 0x00 0x7F;
// 00 00 00 00 0c 1f 63 00 03 90 00 = Purse ID : 0x00000000; ExpDate 12/31/99; PUN 3

echo "****Complete Parameter Update: CAD ID 0x11223344; Set ExpDate 12/31/98";
0x80 0x26 0x00 0x00 0x15 0x11 0x22 0x33 0x44 0x00 0x00 0x00 0x00 0xC5 0x03 0x0c 0x1f 0x62
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x7F;
// 00 00 00 00 00 00 00 00 90 00;

echo "****Initialize Parameter Update";
0x80 0x24 0x00 0x00 0x00 0x7F;
// 00 00 00 00 0c 1f 62 00 04 90 00 = Purse ID : 0x00000000; ExpDate 12/31/98; PUN 4

echo "****Complete Parameter Update: CAD ID 0x11223344; Set Purse ID 0x05050505";
0x80 0x26 0x00 0x00 0x16 0x11 0x22 0x33 0x44 0x00 0x00 0x00 0x00 0xC6 0x04 0x05 0x05 0x05
0x05 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x7F;
// 00 00 00 00 00 00 00 00 90 00;

echo "****Initialize Parameter Update";
0x80 0x24 0x00 0x00 0x00 0x7F;
// 05 05 05 05 0c 1f 62 00 05 90 00 = Purse ID : 0x05050505; ExpDate 12/31/98; PUN 5
```

CODE EXAMPLE 6-2 `testpurse.scr` *(Continued)*

```
echo "****Complete Parameter Update: CAD ID 0x11223344; Set Max Balance $320.00;";
0x80 0x26 0x00 0x00 0x14 0x11 0x22 0x33 0x44 0x00 0x00 0x00 0x00 0xC7 0x02 0x7D 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x7F;
// 00 00 00 00 00 00 00 00 90 00;

echo "****Initialize Parameter Update";
0x80 0x24 0x00 0x00 0x00 0x7F;
// 05 05 05 05 0c 1f 62 00 06 90 00 = Purse ID : 0x05050505; ExpDate 12/31/98; PUN 6

echo "****Complete Parameter Update: CAD ID 0x11223344; Set Max Transaction $30.00;";
0x80 0x26 0x00 0x00 0x14 0x11 0x22 0x33 0x44 0x00 0x00 0x00 0x00 0xC8 0x02 0x0B 0xB8 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x7F;
// 00 00 00 00 00 00 00 00 90 00;

echo "****Initialize Parameter Update";
0x80 0x24 0x00 0x00 0x00 0x7F;
// 05 05 05 05 0c 1f 62 00 07 90 00 = Purse ID : 0x05050505; ExpDate 12/31/98; PUN 7

echo "****Complete Parameter Update: CAD ID 0x11223344; Set Java Purse Version 2.1.0.1";
0x80 0x26 0x00 0x00 0x16 0x11 0x22 0x33 0x44 0x00 0x00 0x00 0x00 0xC9 0x04 0x02 0x01 0x00
0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x7F;
// 00 00 00 00 00 00 00 00 90 00;

echo "****Initialize Parameter Update";
0x80 0x24 0x00 0x00 0x00 0x7F;
// 05 05 05 05 0c 1f 62 00 08 90 00 = Purse ID : 0x05050505; ExpDate 12/31/98; PUN 8

echo "****Complete Parameter Update: CAD ID 0x11223344; Loyalty1 =
0xa0,00,00,00,62,03,01,0c,05,01 ";
0x80 0x26 0x00 0x00 0x1E 0x11 0x22 0x33 0x44 0x00 0x00 0x00 0x00 0xCA 0x0C 0x33 0x55 0xA0
0x00 0x00 0x00 0x62 0x03 0x01 0x0C 0x05 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x7F;
// 00 00 00 00 00 00 00 00 90 00;

/////////////////////////////////////////////////////////////////////
// End of initialization session, all values are set up.
/////////////////////////////////////////////////////////////////////

/////////////////////////////////////////////////////////////////////
// Regular  transaction session  at CAD 22446688 in the Bank
/////////////////////////////////////////////////////////////////////

echo "****Select JavaPurse";
0x00 0xa4 0x04 0x00 10 0xa0 0 0 0 0x62 3 1 0xc 2 1 127;
// 90 00 = SW_NO_ERROR

echo "****Verify PIN (User PIN 01020304)";
0x00 0x20 0x00 0x82 0x04 0x01 0x02 0x03 0x04 0x00;
// 90 00;

echo "****Initialize Transaction: Credit $250.00 ";
0x80 0x20 0x01 0x00 0x0a 0x61 0xa8 0x22 0x44 0x66 0x88 0x00 0x00 0x00 0x00 0x7F;
// 05 05 05 05 0c 1f 62 00 00 00 01 00 00 00 00 00 00 00 00 90 00
//= Purse ID : 0x05050505; ExpDate 12/31/98; TN=1
```

CODE EXAMPLE 6-2 `testpurse.scr` *(Continued)*

```
echo "****Complete Transaction: Date 10/27/97; Time 15:33";
0x80 0x22 0x00 0x00 0x0d 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x0a 0x1b 0x61 0x0f 0x21
0x7F;
// 61 a8 00 00 00 00 00 00 00 00 90 00  = Purse Balance $250.00;

echo "****Initialize Transaction: Debit $25.00;";
0x80 0x20 0x02 0x00 0x0a 0x09 0xc4 0x22 0x44 0x66 0x88 0x00 0x00 0x00 0x00 0x7F;
// 05 05 05 05 0c 1f 62 61 a8 00 02 00 00 00 00 00 00 00 00 90 00;
//= Purse ID : 0x05050505; ExpDate 12/31/98; TN=2

echo "****Complete Transaction: Date 10/27/97; Time 15:35";
0x80 0x22 0x00 0x00 0x0d 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x0a 0x1b 0x61 0x0f 0x23
0x7F;
// 57 e4 00 00 00 00 00 00 00 00 90 00  = Purse Balance $225.00;

/////////////////////////////////////////////////////////////////////
// Regular  transaction session  at CAD 33557799 in a store
/////////////////////////////////////////////////////////////////////

echo "****Select JavaPurse";
0x00 0xa4 0x04 0x00 10 0xa0 0 0 0 0x62 3 1 0xc 2 1 127;
// 90 00 = SW_NO_ERROR

echo "****Verify PIN (User PIN 01020304)";
0x00 0x20 0x00 0x82 0x04 0x01 0x02 0x03 0x04 0x00;
// 90 00;

echo "****Initialize Transaction: Debit $22.95";
0x80 0x20 0x02 0x00 0x0a 0x08 0xf7 0x33 0x55 0x77 0x99 0x00 0x00 0x00 0x00 0x7F;
// 05 05 05 05 0c 1f 62 57 e4 00 03 00 00 00 00 00 00 00 00 90 00;
//= Purse ID : 0x05050505; ExpDate 12/31/98; TN=3

echo "****Complete Transaction: Date 10/27/97; Time 17:45";
0x80 0x22 0x00 0x00 0x0d 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x0a 0x1b 0x61 0x11 0x2d
0x7F;
// 4e ed 00 00 00 00 00 00 00 00 90 00  = Purse Balance $202.05

/////////////////////////////////////////////////////////////////////
// A session with various errors at CAD 33445566
/////////////////////////////////////////////////////////////////////

echo "****Select JavaPurse";
0x00 0xa4 0x04 0x00 10 0xa0 0 0 0 0x62 3 1 0xc 2 1 127;
// 90 00 = SW_NO_ERROR

echo "****Initialize Transaction: Debit $22.95";
0x80 0x20 0x02 0x00 0x0a 0x08 0xf7 0x33 0x44 0x55 0x66 0x00 0x00 0x00 0x00 0x7F;
// 69 82 = SW "Security Status Not Satisfied" : must present PIN first

echo "****Verify PIN (User PIN 01030507)";
0x00 0x20 0x00 0x82 0x04 0x01 0x03 0x05 0x07 0x00;
// 69 c4 = SW_PIN_FAILED, 4 tries remained
```

CODE EXAMPLE 6-2 `testpurse.scr` *(Continued)*

```
echo "****Initialize Transaction: Debit $22.95";
0x80 0x20 0x02 0x00 0x0a 0x08 0xf7 0x33 0x44 0x55 0x66 0x00 0x00 0x00 0x00 0x7F;
// 69 82 = SW "Security Status Not Satisfied"

echo "****Verify PIN (User PIN 01020304)";
0x00 0x20 0x00 0x82 0x04 0x01 0x02 0x03 0x04 0x00;
// 90 00 = SW_NO_ERROR

echo "****Complete Transaction: Date 10/28/97; Time 18:45";
0x80 0x22 0x00 0x00 0x0d 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x0a 0x1c 0x61 0x12 0x2d
0x7F;
// 91 04 = SW_COMMAND_OUT_OF_SEQUENCE: Complete command should follow valid Initialize

echo "****Initialize Transaction: Debit $22.95";
0x80 0x20 0x02 0x00 0x0a 0x08 0xf7 0x33 0x44 0x55 0x66 0x00 0x00 0x00 0x00 0x7F;
// 05 05 05 05 0c 1f 62 4e ed 00 04 00 00 00 00 00 00 00 00 90 00  = TN = 4; Balance =
$202.05

echo "****Complete Transaction: Date 10/28/97; Time 18:48";
0x80 0x22 0x00 0x00 0x0d 0x11 0x11 0x11 0x11 0x11 0x11 0x11 0x11 0x0a 0x1c 0x61 0x12 0x30
0x7F;
// 91 05 = SW_WRONG_SIGNATURE: This attempt of transaction is recorded in the log

echo "****Complete Transaction: Date 10/28/97; Time 18:50;";
0x80 0x22 0x00 0x00 0x0d 0x35 0xa9 0x3b 0x26 0x50 0x58 0x97 0x93 0x0a 0x1c 0x61 0x12 0x32
0x7F;
// 91 04 = SW_COMMAND_OUT_OF_SEQUENCE
// (Transaction with a wrong signature is in a way completed,
// We can't retry with another signature.)

echo "****Initialize transaction: Debit $9.86";
0x80 0x20 0x02 0x00 0x0a 0x03 0xda 0x33 0x44 0x55 0x66 0x00 0x00 0x00 0x00 0x7F;
// 05 05 05 05 0c 1f 62 4e ed 00 05 00 00 00 00 00 00 00 00 90 00 = TN = 5; Balance =
$202.05

echo "****Complete Transaction: Date 10/28/97; Time 18:53";
0x80 0x22 0x00 0x00 0x0d 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x0a 0x1c 0x61 0x12 0x35
0x7F;
// 4b 13 00 00 00 00 00 00 00 00 90 00 = Balance = $192.19

echo "****Initialize transaction: Debit $30.01";
0x80 0x20 0x02 0x00 0x0a 0x0b 0xb9 0x33 0x44 0x55 0x66 0x00 0x00 0x00 0x00 0x7F;
// 91 03 = SW_AMOUNT_TOO_HIGH (The Max Amount was set to $30.00)

echo "****Initialize transaction: Credit $127.82";
0x80 0x20 0x01 0x00 0x0a 0x31 0xee 0x33 0x44 0x55 0x66 0x00 0x00 0x00 0x00 0x7F;
// 91 01 = SW_CREDIT_TOO_HIGH (The Max Balance was set to $320.00,
// this transaction would bring it to 320.01)

/////////////////////////////////////////////////////////////////////
// Session of reading balance and log at CAD 22446688 in the Bank
/////////////////////////////////////////////////////////////////////

echo "****Select JavaPurse";
```

CODE EXAMPLE 6-2 `testpurse.scr` *(Continued)*

```
0x00 0xa4 0x04 0x00 10 0xa0 0 0 0 0x62 3 1 0xc 2 1 127;
// 90 00 = SW_NO_ERROR

echo "****Verify PIN (User PIN 01020304)";
0x00 0x20 0x00 0x82 0x04 0x01 0x02 0x03 0x04 0x00;
// 90 00;

echo "****Read the only record in Balances file : ";
echo "****SFI = 4 (00100), record N is specified in P1 => P2 = 00100100 = 0x24";
0x00 0xb2 0x01 0x24 0x00 0x7F;
// 4b 13 7d 00 0b b8 90 00 = Balance = $192.19,
// Max Balance = $320.00, Max Transaction = $30;
echo "*****Read the first record in log file";
echo "****SFI = 3 (00011), record N is specified in P1 => P2 = 00011100 = 0x1c";
0x00 0xb2 0x01 0x1c 0x00 0x7F;
// 00 05 02 03 da 33 44 55 66 0a 1c 61 12 35 4b 13 90 00 90 00
// TN = 5; Transaction Type = DEBIT(02); Amount = $9.86(03da); CAD ID 33445566;
// Date 10/28/97 (0a 1c 61); Time 18:53(12 35); Balance $192.19 (4b 13),
// SW = NO_ERROR (9000)

echo "****Read the second record in log file";
echo "****FI = 3 (00011), record N is specified in P1 => P2 = 00011100 = 0x1c";
0x00 0xb2 0x02 0x1c 0x00 0x7F;
// 00 04 02 08 f7 33 44 55 66 0a 1c 61 12 30 4e ed 91 05 90 00;
// TN = 4; Transaction Type = DEBIT(02); Amount = $22.95(08f7); CAD ID 33445566;
// Date 10/28/97 (0a 1c 61); Time 18:53(12 35); Balance $202.05 (4eed),
// SW_WRONG_SIGNATURE (9105)
// Attempt of the transaction is recorded, but balance wasn't change, see next record.

echo "****Read the third record in log file";
echo "****SFI = 3 (00011), record N is specified in P1 => P2 = 00011100 = 0x1c";
0x00 0xb2 0x03 0x1c 0x00 0x7F;
// 00 03 02 08 f7 33 55 77 99 0a 1b 61 12 2d 4e ed 90 00 90 00
// TN = 3; Transaction Type = DEBIT(02); Amount = $22.95(08f7); CAD ID 33557799;
// Date 10/27/97 (0a 1b 61); Time 18:45(12 2d); Balance $202.05 (4eed), SW = NO_ERROR (9000)

echo "****Read the fifth record in log file";
echo "**** = 3 (00011), record N is specified in P1 => P2 = 00011100 = 0x1c";
0x00 0xb2 0x05 0x1c 0x00 0x7F;
// 00 01 01 61 a8 22 44 66 88 0a 1b 61 0f 21 61 a8 90 00 90 00;
// TN = 1; Transaction Type = CREDIT(01); Amount = $250.00(61a8); CAD ID 22446688;
// Date 10/27/97 (0a 1b 61); Time 15:33(0f 21); Balance $250.00 (61a8), SW = NO_ERROR (9000)

echo "****Read the sixth record in log file";
echo "****SFI = 3 (00011), record N is specified in P1 => P2 = 00011100 = 0x1c";
0x00 0xb2 0x06 0x1c 0x00 0x7F;
// 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 90 00
// Empty record

echo "****Read Expiration Date from Parameters file";
echo "****SFI = 2 (00010), record tag 0xc5 is in P1 => P2 = 00010000 = 0x10;";
0x00 0xb2 0xc5 0x10 0x00 0x7F;
// 69 82 : SW Security status not satisfied -
// One has to present Master PIN to read Parameters
echo "****Verify PIN : Master PIN";
```

CODE EXAMPLE 6-2 `testpurse.scr` *(Continued)*

```
0x00 0x20 0x00 0x81 0x08 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x00;
// 90 00;

echo "****Read Expiration Date from Parameters file";
echo "****SFI = 2 (00010), record tag 0xc5 is in P1 => P2 = 00010000 = 0x10;";
0x00 0xb2 0xc5 0x10 0x00 0x7F;
// c5 03 0c 1f 62 90 00   = Tag = 0xc5, Exp. Date = 12/31/98 (0c 1f 62)

echo "****Select File: select EF under current DF (P1 = 0x02); FID = 0x9103";
0x00 0xa4 0x02 0x0c 0x02 0x91 0x03 0x00;
// 90 00;

echo "****Read the first record in the selected file";
echo "****currently selected file, record N is specified in P1 => P2 = 00000100 = 0x04";
0x00 0xb2 0x01 0x04 0x00 0x7F;
// 00 05 02 03 da 33 44 55 66 0a 1c 61 12 35 4b 13 90 00 90 00
// TN = 5; Transaction Type = DEBIT(02); Amount = $9.86(03da); CAD ID 33445566;
// Date 10/28/97 (0a 1c 61); Time 18:53(12 35); Balance $192.19 (4b 13),
// SW = NO_ERROR (9000)
// *** SCRIPT END ***
powerdown;
```

Index

A
APDUTool, 3-7

C
CapDump, 3-10
Capgen, 3-11
Converter, 3-13

D
DeployCap, 3-16

E
example, Ant Tasks, 6-39
EXP2Text, 3-18
ExportFiles, 4-32

H
home directory, 2-4

I
installation, 2-3

L
library dependencies, 2-5

M
Maskgen, 3-20
message URL http
    //ant.apache.org, 2-3

N
NetBeans integration, 5-35

S
Scriptgen, 3-22
Setup Ant tasks, 2-4
system requirements, 2-3

V
VerifyCap, 3-24
VerifyExp, 3-26
VerifyRev, 3-28