GemXpresso RAD III Version 3.2

Getting Started
All information herein is either public information or is the property of and owned solely by Gemplus S.A. who shall have and keep the sole right to file patent applications or any other kind of intellectual property protection in connection with such information.

Nothing herein shall be construed as implying or granting to you any rights, by license, grant or otherwise, under any intellectual and/or industrial property rights of or concerning any of Gemplus’ information.

This document can be used for informational, non-commercial, internal and personal use only provided that:

- The copyright notice below, the confidentiality and proprietary legend and this full warning notice appear in all copies.
- This document shall not be posted on any network computer or broadcast in any media and no modification of any part of this document shall be made.

Use for any other purpose is expressly prohibited and may result in severe civil and criminal liabilities.

The information contained in this document is provided “AS IS” without any warranty of any kind. Unless otherwise expressly agreed in writing, Gemplus makes no warranty as to the value or accuracy of information contained herein. The document could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Furthermore, Gemplus reserves the right to make any change or improvement in the specifications data, information, and the like described herein, at any time.

Gemplus hereby disclaims all warranties and conditions with regard to the information contained herein, including all implied warranties of merchantability, fitness for a particular purpose, title and non-infringement. In no event shall Gemplus be liable, whether in contract, tort or otherwise, for any indirect, special or consequential damages or any damages whatsoever including but not limited to damages resulting from loss of use, data, profits, revenues, or customers, arising out of or in connection with the use or performance of information contained in this document.

Gemplus does not and shall not warrant that this product will be resistant to all possible attacks and shall not incur, and disclaims, any liability in this respect. Even if each product is compliant with current security standards in force on the date of their design, security mechanisms’ resistance necessarily evolves according to the state of the art in security and notably under the emergence of new attacks. Under no circumstances, shall Gemplus be held liable for any third party actions and in particular in case of any successful attack against systems or equipment incorporating Gemplus products.

Gemplus disclaims any liability with respect to security for direct, indirect, incidental or consequential damages that result from any use of its products. It is further stressed that independent testing and verification by the person using the product is particularly encouraged, especially in any application in which defective, incorrect or insecure functioning could result in damage to persons or property, denial of service or loss of privacy.

© Copyright 2002 Gemplus S.A. All rights reserved. Gemplus, the Gemplus logo, and GemXpresso are trademarks and service marks of Gemplus S.A. and are registered in certain countries. All other trademarks and service marks, whether registered or not in specific countries, are the property of their respective owners. Certain Smart Cards produced by Gemplus are covered by Bull CP8 Patents.

This product includes software developed by the Apache Software Foundation (http://www.apache.org).

GEMPLUS, B.P. 100, 13881 GEMENOS CEDEX, FRANCE.
Tel: +33 (0)4.42.36.50.00 Fax: +33 (0)4.42.36.50.90
Printed in France. Document Reference: DOC107569A0
May 7, 2002
### Contents

**Preface**
- Who Should Read This Book ........................................ vii
- Conventions Used in This Document ............................... viii
- Contact Our Hotline .................................................. ix

**Chapter 1**  **Introduction** ........................................ 1
- The Java Card Environment .......................................... 1
- The Open Platform Specification .................................... 2
- Java Card Concepts and Terminology .............................. 3
  - Applets .............................................................. 3
  - Client Applications ............................................... 4
  - Packages ............................................................ 4
  - CAP Files ........................................................... 4
  - JAR Files ............................................................ 4
  - SAP Files ........................................................... 5
- Converting the Applet to a Loadable Format .................... 5
  - Projects ............................................................ 5
  - AIDs ................................................................. 6
  - Targets .............................................................. 7
- The GemXpresso RAD III Tools ...................................... 7
- GemXpresso RAD III Development Scenarios ..................... 9
  - Tools Used During the Development Phase ..................... 9
  - Tools Used During the Conversion Phase ....................... 11
  - Tools Used During the Test Phase ............................... 11

**Chapter 2**  **Installation and Configuration** ....................... 13
- Hardware and Software Requirements ............................. 13
- Hardware Requirements .............................................. 13
- Software Requirements ................................................ 14
Installation Overview ................................................................. 14
Installing GemXpresso RAD III .................................................. 15
Minimal Configuration for Using the GemXpresso RAD III Components ................................................. 16
If an Environment Error Occurs ................................................... 18

Chapter 3 Getting Started with GemXpresso RAD III 19
Examining the OPPurse Source Code ............................................. 19
A Typical Development Cycle ..................................................... 19

Chapter 4 Creating the OPPurse Project 21
Creating the Project with VisualCafé .............................................. 21
Creating the Project with JBuilder ................................................ 22
Using the Import GemXpresso Project Wizard ................................ 23
Exploring the Project Setup with VisualCafé ................................... 25
Exploring the Project Setup with JBuilder ....................................... 26

Chapter 5 Preparing the Applet for Loading 27
Introduction .................................................................................. 27
Opening the Project Editor Window .............................................. 27
Converting Files ........................................................................... 29

Chapter 6 Loading the Applet 33
Setting up the JCardManager ....................................................... 33
Starting the GSE (Card Simulator) ................................................ 33
Starting the JCardManager .......................................................... 35
Performing Authentication .......................................................... 36
Displaying the Contents of the Card .............................................. 36
Loading OPPurse into the GSE ..................................................... 38
Installing OPPurse in the GSE ....................................................... 39
Selecting OPPurse for Use .......................................................... 41

Chapter 7 Testing the Applet 43
Testing the Applet’s Functionality .................................................. 43
Sending User-defined APDU Commands ................................. 43
Getting the Current Balance of the Electronic Purse ....................... 44
Crediting the Electronic Purse ....................................................... 45
Debiting the Electronic Purse ........................................................ 47
### Contents

Saving the Commands List ................................................................. 48

**Chapter 8**  
**Next Steps...**  
The GemXpresso Family of Products ................................................. 52

**Terminology**  
Abbreviations .............................................................................. 53  
Glossary ....................................................................................... 56

**For More Information**  
Standards and Specifications ............................................................ 61  
Recommended Reading .................................................................. 61
List of Figures

Figure 1 - The Java Card Environment ................................................. 1
Figure 2 - The Compilation, Conversion, and Loading Process ...................... 5
Figure 3 - The AID Format ................................................................. 6
Figure 4 - The Development Phase ....................................................... 10
Figure 5 - The Conversion Phase ......................................................... 11
Figure 6 - The Test Phase ................................................................. 12
Figure 7 - Development Flow with GemXpresso RAD Tools ......................... 20
Figure 8 - VisualCafé's New Project Window ........................................... 21
Figure 9 - JBuilder's Object Gallery Window .......................................... 22
Figure 10 - First Page of Import Project Wizard ....................................... 23
Figure 11 - Second Page of Project Creation Wizard ................................... 24
Figure 12 - The Project Editor Window .................................................. 28
Figure 13 - The Convert Window ......................................................... 30
Figure 14 - Location of the Converted Files ........................................... 31
Figure 15 - The Gemplus Simulation Environment Launcher Window ............. 34
Figure 16 - The JCardManager Window .................................................. 35
Figure 17 - JCardManager Window with Selected Card and Terminal .............. 35
Figure 18 - The Gemplus Simulation Environment Window ......................... 37
Figure 19 - Uploading a File with the JCardManager .................................. 38
Figure 20 - Uploaded Package in the Card Simulator Window ....................... 39
Figure 21 - The Installed Applet ......................................................... 40
Figure 22 - OPPurse Property Sheet on the GSE GUI Window ....................... 41
Figure 23 - Edit Command List Window ................................................. 49

List of Tables

Table 1 - JavaCard and OP Specification Compatibility ............................... 2
Table 2 - Development Stages and Steps ................................................ 8
Table 3 - Accessing GemXpresso RAD III Tools ...................................... 9
Table 4 - getBalance APDU Parameters ............................................... 44
Table 5 - VerifyPIN APDU Parameters .................................................. 46
Table 6 - Credit APDU Parameters ....................................................... 47
Table 7 - Debit APDU Parameters ....................................................... 48
Preface

This document is designed to help you learn about GemXpresso RAD III by using it. It guides you step by step from the creation of a new Java Card project, through to the installation and testing of an applet.

The examples described in this document all use the OPPurse sample applet provided with GemXpresso RAD III. OPPurse is a fully functional Electronic Purse application based on both the Java Card 2.1.1 and Open Platform 2.0.1 standards. It runs on any of the cards included with the version of GemXpresso RAD III you have purchased.

The examples describe a basic, but complete, applet development cycle. Topics include:

- How to install GemXpresso RAD III
- How to create a Java Card applet project in an integrated development environment (IDE)
- How to prepare the project for loading
- How to load and install the applet
- How to test the applet

Who Should Read This Book

You need no specific knowledge about smart cards, but you are strongly recommended to read the standards and documents listed in the bibliography of the GemXpresso RAD III Version 3.2 User’s Guide.
Conventions Used in This Document

**bold** Command and window names are shown in bold. For example:
... the **JCardManager** window...

> Menu selection sequences are shown using the > symbol to link the selections. For example:
... select **Start** > **Programs** > **Gemplus Applications**...

**italics** Cross references and book titles are shown in *italics*.

**notation**
- By default, a numeric value is expressed in decimal notation.
- Whenever a value is expressed in binary, it is followed by the letter “b”. For example the decimal value 13 expressed in binary becomes **1101b**.
- A hexadecimal number is followed by the letter “h”, or preceded by “0x”. For example, the decimal value 13 expressed in hexadecimal becomes **0Dh** or **0x0D**.

**installdir**
This variable is used throughout this book to indicate the directory in which GemXpresso RAD III is installed. For example, when the following directory is shown in this book:  
```text
installdir\bin
```
you may need to substitute the following directory:
```text
c:\Gemplus\gemxpresso.rad3\bin
```

**cardname**
This variable is used throughout this book to indicate a particular type of card. The possible card types are:
- GXP211_PK (GemXpresso 211 Compact PK)
- GXP211_PK_IS (GemXpresso 211 Compact PK with limited cryptography)
- GXP211v2 (GemXpresso 211 V2)
- GXP211v2_IS (GemXpresso 211 V2 with limited cryptography)
- GXPLite-Generic (GemXpresso Lite Generic)
- GXPPro-R3 (GemXpresso Pro R3)

For example, if you are using a GemXpresso Pro R3 card, substitute “GXPPro-R3” for **cardname** wherever it appears in the text.
Contact Our Hotline

If you do not find the information you need in this manual, or if you find errors, contact the Gemplus hotline by phone, fax, or email. In your email, please include the document reference number, your job function, and the name of your company. (You will find the document reference number at the bottom of the legal notice on the inside front cover.)

**Corporate and EMEA**
Hotline: +33 (0)4 42 36 50 50
Hot Fax: +33 (0)4 42 36 50 98
Email: hotline@gemplus.com

**Americas**
Hotline: 1 (877) 436-7233
Hot Fax: 1 (215) 390-1586
Email: hotlineusa@gemplus.com

**From Our Web Site**
http://www.gemplus.com
Introduction

This chapter outlines a Java Card environment and explains the main Java Card concepts and terminology. It also introduces the GemXpresso RAD III tools to use during the development, debugging, conversion and test phases of a project.

The Java Card Environment

“Figure 1 - The Java Card Environment” shows the main components of a Java Card application.

Java Cards contain a microprocessor, but have no means of directly receiving input or displaying output. They must therefore be connected to a card reader, also known as a card acceptance device (CAD), which is in turn connected to a workstation such as a personal computer (PC) by a serial cable or over a TCP/IP network connection.

Applets, (or “Java Card applications”), are normally developed in conjunction with a client application that runs on the workstation. The client application controls all input and output operations on behalf of the applet, and sends requests for processing to the applet. The applet acts as a server, responding to the client application’s requests.
The card in a card reader and the client application communicate using application protocol data unit (APDU) commands. APDUs contain command instructions, data and responses. Refer to the GemXpresso RAD III Version 3.2 User’s Guide for detailed information on APDUs.

The Open Platform Specification

Visa’s open platform (OP) is a generic framework for the management of multi-application smart cards. It extends the Java Card environment by adding mechanisms for securely managing the applications on the card. OP includes several components:

- A set of commands to manage the life cycle of the card and its applications, load and install the applications on the card, and manage the card’s security by, for example, updating keys and setting up a secure channel between the card and the terminal.
- An API, consisting of a single Java package, visa.openplatform, that can be used by application developers to access the OP features, and in particular the application’s life cycle and the secure messaging mechanism.
- A documented specification, which fully defines the commands available and the principles of interoperability between the Java Card and OP card environments. The current version is OP 2.0.1.

The Gemplus GemXpresso card family is fully compatible with the OP 2.0.1 standard.

Note: Some cards are compatible with earlier versions as shown in the following table:

<table>
<thead>
<tr>
<th>Card Profile</th>
<th>JavaCard Spec Version</th>
<th>OP Spec Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>GemXpresso 211 V2 &amp; IS</td>
<td>2.1</td>
<td>2.0</td>
</tr>
<tr>
<td>GemXpresso 211 PK &amp; PK_IS</td>
<td>2.1.1</td>
<td>2.0.1</td>
</tr>
<tr>
<td>GemXpresso Lite Generic</td>
<td>2.1.1</td>
<td>2.0.1</td>
</tr>
<tr>
<td>GemXpresso Pro R3</td>
<td>2.1.1</td>
<td>2.0.1</td>
</tr>
</tbody>
</table>

Table 1 - JavaCard and OP Specification Compatibility

To obtain the OP specification, go to www.globalplatform.org.
Java Card Concepts and Terminology

Applets

An applet is a Java program designed to work within the Java Card environment. Apart from being programmed in Java, a Java Card applet is very different from a web browser-based Java applet.

**Note:** An applet is often referred to in specifications as a “Java Card application”. The open platform (OP) documentation, for example, is not specific to Java Cards and uses the term “application” throughout.

Applets can be pre-installed on a card during the manufacturing process. They can also, however, be downloaded to a card at subsequent stages of the card’s life cycle using “over the air” techniques, for example. Many different applets can be installed on the same card, and multiple instances of the same applet can be created (“instantiated”) and run on the same card. The only constraint is the amount of memory available (although today this is still a very significant constraint).

Two types of card available today can run applets:

- Cards compatible only with the Java Card environment (“Java Cards”).
- Cards compatible with both the Java Card and open platform environments (“Java Card-OP cards”).

All GemXpresso cards are Java Card-OP cards.

Applets written for Java Cards must be written using exclusively Java Card API methods.

Applets written for Java Card-OP cards can be written using either exclusively Java Card API methods (Java Card applets), or a combination of Java Card and OP methods (Java Card-OP applets).

When writing Java source code, the type of API methods used (Java Card or Java Card-OP) is an important issue, which influences, for example:

- The choice of GemXpresso RAD III project template (if you are using and IDE and one of the supplied templates to prototype the applet).
- The type of OCF Card Service required.
- The applet loading and installation procedure.
- The possible targets available (one or more specific card type, or the card simulator only).
Client Applications

A client application is a software program running on a PC or workstation that communicates with an applet that has been uploaded and installed on a card. Client applications are usually written at the same time as the applet code.

Client applications use the appropriate opencard framework (OCF) APIs, as follows:

- Java Card applets can be developed with OP client applications using the OP CardService, or with client applications using the Gemplus-proprietary instruction set of the GSE CardService.
- Java Card-OP applets are developed with OP client applications using the OP CardService. These client applications use the OP CardService.

Packages

A package is a Java programming term to describe a collection of related classes and interfaces. In global system for mobile communications (GSM) environments, it is also known as an “executable load file”. A single package can contain one or more applets. Since an entire package can be uploaded to a card, it is therefore easy to upload several applets at the same time. A client application’s class must belong to a different package than the applet’s classes.

CAP Files

A CAP file is a portable representation of a Java Card applet. CAP files are in a binary format capable of being loaded directly onto a Java Card, and are used by the card’s Java Card Virtual Machine to execute the applet.

For a complete description of the Java Card CAP File format, see Sun’s Java Card 2.1 Virtual Machine Specification.

You can use the CapDump utility (see “Chapter 8 - The CapDump Utility”) to view the contents of a CAP file.

JAR Files

Because a single package can generate many CAP files, the Java Card specification recommends that CAP files be stored in a Java Archive (JAR) file before being loaded onto a card. The JAR file is stored in a directory called \javacard immediately below the package’s directory. For example, the JAR file for the package com.gemplus.examples.OPPurse is placed in the subdirectory com\gemplus\examples\OPPurse\javacard.
To view the contents of a JAR file, you can use tools provided for this purpose by IDEs such as JBuilder or VisualCafé. Because JAR files use the ZIP compression algorithm, you can also view their contents using a tool such as WinZip.

**SAP Files**

SAP files are a Gemplus-proprietary version of the JAR file format containing CAP files. They cannot be uploaded to a real card, but once loaded into the GSE environment, they accurately simulate an applet in almost all other ways.

**Converting the Applet to a Loadable Format**

Before the applet can be loaded on a SIM card, it must be converted to byte code that can be loaded onto the SIM card and interpreted by the Java Virtual Machine on the SIM card. This involves converting the .Class file to a .CAP (for real cards) or .SAP (for simulated cards) format. GemXpresso RAD III provides the GxpConverter tool to facilitate this conversion, as shown in “Figure 2 - The Compilation, Conversion, and Loading Process”:

![Figure 2 - The Compilation, Conversion, and Loading Process](image)

**Projects**

A project refers to one or more applets, together with the necessary system classes, with or without libraries. If you are using an integrated development environment such as JBuilder or VisualCafé, a project file groups together all the necessary files and environment variables.
AIDs

For security reasons, client applications must be able to uniquely identify any installed applet. Every package, applet, and instance of an applet loaded on a card must therefore be assigned a unique identifier, known as an application identifier (AID). An AID is a string of between 5 and 16 hexadecimal bytes.

The first five bytes of an AID (the RID) indicate the Application Provider, that is, the company supplying the package or applet. Rules governing the choice of RID depend on the type of card the applet is to be loaded onto:

- For cards that support Java Card 2.1.1 (see Table 1, “JavaCard and OP Specification Compatibility”, on page 2), a different RID can be used for each package and its installed applet.
- For all other card types, the same RID must be used for all the packages and applets in a project.

To obtain an RID, your company must register with the International Standards Organization (ISO). This procedure is described in ISO7816-5 “Numbering system and registration procedure for application identifiers”.

The remaining bytes (up to 11) of an AID contain the proprietary identifier extension (PIX). The PIX is chosen by the Application Provider to uniquely identify a package or applet. Your company is then responsible for assigning PIXs to individual applications.

As mentioned previously, three types of AID are used in a project:

- The package AID. An AID assigned to a package.
- The applet AID. An AID assigned to an applet.
- The applet instance AID. An AID assigned to an instance of an applet.

An applet AID cannot have the same value as the AID of any other package or any other applet stored on the same card.
**Targets**

Applets can be loaded onto two types of target:

- A Java Card, which can be either Java Card or Java Card-OP compatible.
- The Gemplus simulation environment (GSE), which simulates a card/card reader combination. The GSE makes it possible to develop applets without having access to actual Java Cards or a card reader.

The GSE is described in detail in the *GemXpresso RAD III Version 3.2 User’s Guide*.

**The GemXpresso RAD III Tools**

GemXpresso RAD III provides appropriate tools for each stage of development and testing, as follows:

- **Quick-start tools**: Templates providing ready-to-compile skeleton code for applets. The Project Editor defines project environment variables.
- **Pre-loading tools**: The GxpConverter prepares the applet for loading into the card.
- **Loading tools and post-loading tools**: The JCardManager loads packages into a card or the GSE, and exchanges APDU commands with a loaded applet. The GxpLoader is a command-line utility for loading packages. The Deployment Editor creates configuration files for use by other components allowing batch execution of load and install commands.
- **Simulation tools**: The GSE simulates the reaction of the card for testing and debugging the functionality of an applet. GSE GUI provides a graphical use interface to the GSE.
- **Miscellaneous utilities**: Various file conversion tools are included. The CapDump tool allows you to visualize the content of a loadable CAP file.
The following table lists the GemXpresso RAD III tools and the corresponding development stages and target types required.

<table>
<thead>
<tr>
<th>Development Tools</th>
<th>Development Stages</th>
<th>Target Required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Quick-start tools:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Templates*</td>
<td>Code development and compilation</td>
<td>Applet, or applet and client application</td>
</tr>
<tr>
<td>Project Editor</td>
<td>Project environment setup</td>
<td>Applet, for one or more card types</td>
</tr>
<tr>
<td><strong>Pre-loading tools:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GxpConverter</td>
<td>Converting to loadable CAP files</td>
<td>Applet, for one or more card types</td>
</tr>
<tr>
<td><strong>Loading tools and Post-loading tools:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JCardManager</td>
<td>CAP file loading, command exchange, tracing, scanning</td>
<td>Applet</td>
</tr>
<tr>
<td>GxpLoader</td>
<td>One step conversion, loading and installing of applets.</td>
<td>Applet</td>
</tr>
<tr>
<td>Deployment Editor</td>
<td>Creation of configuration files and scripts for use with the GxpLoader</td>
<td>-</td>
</tr>
<tr>
<td><strong>Simulation and test tools:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GSE</td>
<td>Testing and debugging applets</td>
<td>Standalone applet, or applet and client application</td>
</tr>
<tr>
<td>GSE GUI</td>
<td>Testing and debugging applets</td>
<td>Standalone applet, or applet and client application</td>
</tr>
<tr>
<td><strong>Miscellaneous utilities:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GCF to GXCF Script Converter</td>
<td>Converts GCF scripts to the new XML-based GXCF format.</td>
<td>n/a</td>
</tr>
<tr>
<td>CapDump</td>
<td>Viewing the contents of a CAP file</td>
<td>Applet</td>
</tr>
</tbody>
</table>

* Table 2 - Development Stages and Steps

* Templates are only available to users of GemXpresso RAD III using VisualCafé or JBuilder.
The GemXpresso RAD III tools can be launched from within the plug-ins provided for JBuilder and VisualCafé, from within the JCardManager, or from the command line using supplied batch files. Refer to “Table 3 - Accessing GemXpresso RAD III Tools”.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Accessible from within:</th>
<th>JBuilder/VisualCafé Plug-in</th>
<th>JCard Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>Templates</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Project Editor</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GxpConverter</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>JCardManager</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GSE</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GSE GUI</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CapDump</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Deployment Editor</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>GCF to GXCF Script Converter</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

| Table 3 - Accessing GemXpresso RAD III Tools |

**GemXpresso RAD III Development Scenarios**

The GemXpresso RAD III tools allow you a wide range of flexibility when developing applets and client applications. There is usually more than one way of accomplishing a task.

The following figures illustrate some typical scenarios for the development, conversion and test phases of a project.

**Tools Used During the Development Phase**

For rapid development of a prototype applet, and optionally a client application, use one of the supplied Project Templates to generate skeleton code and a project definition within JBuilder or VisualCafé. Add custom functionality to the code, then set breakpoints and build Class files. Create loadable files using the GxpConverter, then use either the GxpLoader or JCardManager to load the files into the GSE. Optionally, use the Deployment Editor to create a deployment file for use by the GxpLoader or JCardManager containing all necessary load and install parameters.
When a breakpoint is triggered by an applet loaded in the GSE, control is returned to the corresponding line of code within the IDE.

Figure 4 - The Development Phase
Tools Used During the Conversion Phase

Once debugging is complete, you can convert Class files to a loadable format for multiple targets. Use the **Project Editor** to define package and applet AIDs and locations for the converted files, then start the **GxpConverter** to convert the Class files to loadable files for each required target card type. You can also access the Project Editor and GxpConverter from within the **JCardManager**.

![Figure 5 - The Conversion Phase](image)

Tools Used During the Test Phase

Once converted, an applet can be further tested in the **GSE (card simulator)**, and eventually in a real card and card reader. Using the **JCardManager**, the converted CAP files can be uploaded to the chosen target and the applet installed and selected. APDU commands can then be issued to test functionality and analyze the applet’s responses.
The **CapDump** utility is accessible from within the JCardManager to view the contents of the converted CAP files in a readable form.

**Figure 6 - The Test Phase**
Installation and Configuration

In this chapter, you learn how to install GemXpresso RAD III.

Hardware and Software Requirements

Hardware Requirements

The following hardware is required to install and use GemXpresso RAD III:

- An Intel Pentium PC with:
  - Pentium 166 MHz minimum (Pentium II 350 MHz is recommended).
  - 128 MB RAM minimum (256 MB is recommended).
  - 20-30 MB of available disk space, depending on the configuration.
  - Support for Super VGA resolutions (800 x 600). 1024 x 768 is recommended.
  - A CD-ROM drive.
- Optionally, one OCF-compatible card reader, such as the Gemplus GemPC410 (formerly called GCR410).
- Optionally, one of the following types of smart card:
  - A Java Card, compliant with the Java Card 2.1 or Java Card 2.1.1 standards only.
  - A Java Card-OP card, fully compliant with both the Java Card standard and Visa’s open platform (OP) specification. All GemXpresso cards are Java Card-OP cards.
Software Requirements

GemXpresso RAD III runs under the following operating systems:

- Windows 98
- Windows NT 4
- Windows 2000
- RedHat Linux 6.2
- Sun Solaris 8

**Note:** The RAD III will run under Windows XP, but it has not been qualified.

In each case, a Java 2-compatible IDE such as Visual Café or JBuilder must be installed on the installation machine.

Additional software required depends on the development environment being used:

- With VisualCafé version 4.0 (Expert and Enterprise Editions) and JBuilder versions 3.5, 4.0, 5.0 and 6.0, the GemXpresso RAD III environment is installed as a plug-in and a **GemXpresso RAD** menu is available on the IDE’s menu bar, from which many of the tools can be launched directly and display their output in the IDE’s messages window.
- With any other Java IDE or JDK 2 compatible development kit, the GemXpresso RAD tools run in stand-alone mode, and must be launched from the command line or as batch files.

**Caution:** It is highly recommended that you use JDK 1.2.2. The Java Communication API (Comm.jar) delivered with GemXpresso RAD III has been developed and tested for compatibility with the JDK 1.2.2.

- If you are using the GSE card simulator, you must install the TCP/IP networking protocol.

Installation Overview

**Note:** Before beginning installation, uninstall any previous versions of GemXpresso RAD on your machine.

The GemXpresso RAD III environment is installed from a CD-ROM. The installation program installs any additional components required. If you have a card reader installed, it is configured for use. The Gemplus simulation environment (GSE) is automatically installed on your workstation during installation.
Installing GemXpresso RAD III

To install GemXpresso RAD III:

1. Insert the CD-ROM into the CD-ROM drive on your system.

2. The installation program starts automatically. If for any reason installation does not start automatically, run the drive:\autorun\AutoRun.exe program on the installation CD-ROM, where drive is the drive letter of the CD-ROM drive. This program guides you through the installation process.

   **Note:** The default software installation directory is:
   C:|Gemplus|GemXpresso.rad3
   You can specify a different drive or directory. To avoid confusion, the installation directory is referred to as “installdir” throughout this book.

3. You are prompted to select optional components to install, which may include:
   – Java Communication API 2.0
   – OpenCard Framework 1.2
   – Acrobat Reader 4.0
   – Sun’s JDK 1.2.2

   After installation, a “Read Me” file appears automatically, giving additional information about hardware and software requirements. It is recommended that you read this file, either at the time of installation, or later by selecting Start > Programs > Gemplus Applications > GemXpresso RAD III > ReadMe.txt
Minimal Configuration for Using the GemXpresso RAD III Components

The user must have sufficient access rights to read all files in the GemXpresso RAD III installation directory and its subdirectories.

The user must also have Read access rights on the following files (where jbuilderdir is the directory in which JBuilder is installed and vcafedir is the directory in which VisualCafé is installed):

- jbuilderdir\bin\jbuilder.config
- jbuilderdir\lib\ext\gemplus.properties

If you are using JBuilder Version 3.5 or 4:
- jbuilderdir\lib\ext\gxpradplugin-borland.jar

If you are using JBuilder Version 5:
- jbuilderdir\lib\ext\gxpradplugin-jb5.jar

If you are using JBuilder Version 6:
- jbuilderdir\lib\ext\gxpradplugin-jb6.jar

If you are using VisualCafé Version 4:
- vcafedir\bin\sc.ini
- vcafedir\bin\components\gemplus.properties
- vcafedir\bin\components\gxpradplugin-symantec.jar

In addition, you must have Read access to the following directories and subdirectories:

- jbuilderdir\JBUILDERjdkdir\*
- vcafedir\VCAFejdkdir\*
- RADjdkdir\*

Where:

- jbuilderjdkdir is the directory containing JBuilder’s JDK, for example c:\JBuilder6\jdk1.3.1.
- vcafejdkdir is the directory containing VisualCafé’s JDK, for example c:\VisualCafe4\Java2.

RADjdkdir is the directory containing the JDK that you configure for use with GemXpresso RAD III, for example c:\jdk1.3.1.

You must also have Write access to the following directories:

- RADdir
- RADdir\conf
- RADdir\resources\cmd
Installation and Configuration

The files RADdir\conf\*.\* and RADdir\resources\cmd\*.\* must not be in Read-Only mode.

If the files included in subdirectories of RADdir\examples are in Read-Only mode, GemXpresso RAD will display a warning that these files are in read-only mode. To avoid receiving such messages, ensure that the following files are NOT in Read-Only mode:

RADdir\examples\GXP211_PK\CryptoApplet\CryptoApplet.gxp
RADdir\examples\GXP211_PK\OPPurse\OPPurse.gxp
RADdir\examples\GXP211_PK\OPPurse\OPPurse.gxd
RADdir\examples\GXP211_PK\Purse\Purse.gxp
RADdir\examples\GXP211_PK\Purse\Purse.gxd
RADdir\examples\GXP211_PK\ examples\GXP211_PK_IS\CryptoApplet\CryptoApplet.gxp
RADdir\examples\GXP211_PK_IS\OPPurse\OPPurse.gxp
RADdir\examples\GXP211_PK_IS\OPPurse\OPPurse.gxd
RADdir\examples\GXP211_PK_IS\Purse\Purse.gxp
RADdir\examples\GXP211_PK_IS\Purse\Purse.gxd
RADdir\examples\GXP211V2\OPPurse\OPPurse.gxp
RADdir\examples\GXP211V2\OPPurse\OPPurse.gxd
RADdir\examples\GXP211V2\Purse\Purse.gxp
RADdir\examples\GXP211V2\Purse\Purse.gxd
RADdir\examples\GXP211V2\ examples\GXP211V2_IS\OPPurse\OPPurse.gxp
RADdir\examples\GXP211V2_IS\OPPurse\OPPurse.gxd
RADdir\examples\GXP211V2_IS\Purse\Purse.gxp
RADdir\examples\GXP211V2_IS\Purse\Purse.gxd
RADdir\examples\GXPLite-Generic\OPPurse\OPPurse.gxp
RADdir\examples\GXPLite-Generic\OPPurse\OPPurse.gxd
RADdir\examples\GXPLite-Generic\Purse\Purse.gxp
RADdir\examples\GXPLite-Generic\Purse\Purse.gxd
RADdir\examples\GXPPro-R3\CryptoApplet\CryptoApplet.gxp
RADdir\examples\GXPPro-R3\OPPurse\OPPurse.gxp
RADdir\examples\GXPPro-R3\OPPurse\OPPurse.gxd
RADdir\examples\GXPPro-R3\Purse\Purse.gxp
RADdir\examples\GXPPro-R3\Purse\Purse.gxd
If an Environment Error Occurs

If the following error appears during installation:
“Out of Environment space”
the environment space on the local workstation is insufficient. To correct this problem, edit the CONFIG.SYS file and include the line:
shell=c:\command.com c: \ /p /e:2048
Getting Started with GemXpresso RAD III

Examining the OPPurse Source Code

“Chapter 4 - Creating the OPPurse Project” describes how to set up a project within one of the IDEs supported by GemXpresso RAD III. Before starting, you are advised to familiarize yourself with Java Card code by looking at the OPPurse sample applet’s Java source file, OPPurse.java, in your IDE or a text editor. This file is located in:

`installdir\examples\cardname\OPPurse\src\com\gemplus\examples\oppurse`

A printed version of the source code appears in the *GemXpresso RAD III Version 3.2 User’s Guide.*

The OPPurse sample applet illustrates open platform (OP) security and applet management mechanisms, and provides a basic debit/credit (electronic purse) application with:

- A `getBalance()` method
- A PIN-protected `debit()` method
- A `credit()` method including a required mutual authentication with a correct PIN code presentation
- A `VerifyPin()` method, which may change the card’s life cycle state from `SELECTABLE` to `BLOCKED`
- A personalized answer to reset (ATR) value

A Typical Development Cycle

The following flow chart, “Figure 7 - Development Flow with GemXpresso RAD Tools”, shows the basic development process followed in this document for the OPPurse sample applet.
Figure 7 - Development Flow with GemXpresso RAD Tools
Creating the OPPurse Project

GemXpresso RAD III includes:

- **Templates** for the most common types of Java Card development projects. Templates are available to generate skeleton Java code that is ready to compile, or to import existing code into an IDE project. You can then concentrate on adding functionality.

- **Wizards** that gather the information necessary to build and configure a skeleton Java Card project, such as Java package names and default applet AID values.

In this chapter, you learn how to use one of the provided templates and its associated wizard to create a project for the sample OPPurse applet.

Creating the Project with VisualCafé

To create the OPPurse project from within the VisualCafé IDE:

1. Choose **File > New Project**.
2. Select the **Import GemXpresso Project** icon on the **New Project** window:

   ![Figure 8 - VisualCafé's New Project Window](image)

   - **Figure 8 - VisualCafé’s New Project Window**

**Creating the Project with JBuilder**

To create the OPPurse project within the JBuilder IDE:

1. Choose **File > New**.
2. On the **Object Gallery** window, click the **Gemplus** property sheet:
   ![Figure 9 - JBuilder’s Object Gallery Window](image)
   
3. Select **Import GemXpresso Project** and click **OK**.
4. The Import GemXpresso Project wizard starts, as described in the next section.
Using the Import GemXpresso Project Wizard

When importing a GemXpresso project, the following wizard window is displayed:

Specify:

1. **GemXpresso project file to import.** A GemXpresso Project file contains information about an applet, such as the package and applet AIDs to use when loading the applet onto a card. A sample GemXpresso Project file is provided for each of the sample applets. To use the sample file for OPPurse, click **Browse**, navigate to the `installdir\examples\cardname\OPPurse` directory and select the `OPPurse.gxp` project file.

2. In the **Java source files** list box, click **Add** and browse to the following directory:
   
   `installdir\examples\cardname\OPPurse\src\com\gemplus\examples\oppurse`

   Select `OPPurse.java` and click **Open**.

3. The OPPurse project does not reference any external libraries, so leave the **Library files** list empty.
4. Click **Next** to display the second page of the wizard:

![Second Page of Project Creation Wizard](image)

**Figure 11 - Second Page of Project Creation Wizard**

On this window, you configure the target card types for which the applet's files will eventually be converted. Initially, the **Available cards** list shows each available card type, depending on the version of GemXpresso RAD III you have purchased.

5. Select a card type in the **Available cards** list and click the **Add** button to move this card type to the **Selected cards** list. Notice that the card appears as the **GSE target** at the bottom of the window to indicate that it will be used by the Gemplus simulation environment (GSE - the card simulator) as the default card type.

6. In the **Output path** field, enter the directory where converted packages (also known as "executable load files") for your project will be placed:

   ```
   installdir\examples\cardname\OPPurse\oncard
   ```

7. Click **Finish** to complete the wizard.

If you are using VisualCafé, continue with the following section. Otherwise, go to “Exploring the Project Setup with JBuilder” on page 26.
Exploring the Project Setup with VisualCafé

When the import wizard finishes, a VisualCafé project is created and activated. Begin by exploring the following project settings, which are based on the information you provided to the project wizard:

- The applet’s source file, OPPurse.java is inserted in the project.
- The source directory (Project > Options > Directories > Source Files) is set to projectdir\src, where projectdir is the name of the directory you specified for the project name. This is the root directory for Java files belonging to the project. GemXpresso RAD III automatically creates subdirectories below this directory and stores the project’s Java files in the appropriate directory, conforming with Java conventions.
- The output directory (Project > Options > Directories > Output Files) is set to projectdir\out, where projectdir is the name of the directory you specified for the project file name. This is the root directory for output files such as compiled Class files. GemXpresso RAD III automatically creates subdirectories below this directory and stores Class files in the appropriate directory, conforming with Java conventions.
- The GSE (card simulator) is configured as follows (Project > Options > Project):
  Main Class: com.gemplus.javacard.gse.Simulator
  Program Arguments: -port 5000 -card cardname
  By specifying the main class of the GSE, you are able to debug the applet with the card simulator (GSE) once the applet has been installed in the GSE environment.
- The following GemXpresso library files are included in the Classpath (Tools > Environment Options > Internal VM):
  gse_cardname.jar
  cryptix-jce-api.jar
  cryptix-gemxpresso.jar or cryptix-gemxpresso-is.jar

Note: Depending on the version of GemXpresso RAD III you have purchased, either cryptix-gemxpresso.jar (full cryptography), or cryptix-gemxpresso-is.jar (limited cryptography) is installed.

Continue with “Chapter 5 - Preparing the Applet for Loading”, which describes how to prepare the applet’s class files for loading into the card simulator.
Exploring the Project Setup with JBuilder

When the import wizard finishes, a JBuilder project is created and activated. Begin by exploring the following project settings, which are based on the information you provided to the project wizard:

- The applet’s source file, OPPurse.java is inserted in the project.
- The source file directory (Project > Project Properties > Paths > Source) is set to projectdir/src, where projectdir is the name of the directory you specified for the project name. This is the root directory for the project’s Java files. GemXpresso RAD III automatically creates subdirectories below this directory and stores Java files in the appropriate directory, conforming with Java conventions.
- The output path directory (Project > Project Properties > Paths > Output Path) is set to projectdir/out, where projectdir is the name of the directory you specified for the project file name. This is the root directory for output files such as compiled Class files. GemXpresso RAD III automatically creates subdirectories below this directory and stores Class files in the appropriate directory, conforming with Java conventions.
- The GSE (card simulator) is configured as follows (Project > Project Properties > Run):
  
  **Main class:** com.gemplus.javacard.gse.Simulator
  
  **Application parameters:** -port 5000 -card cardname
  
  By specifying the main class of the GSE, you are able to debug the applet with the GSE once the applet has been installed in a simulated card.

- The following library files are configured (Properties > Paths > Required Libraries):
  
  GemXpresso - GSE - cardname (gse_cardname.jar)
  
  GemXpresso - Crypto-1 (cryptix-jce-api.jar and cryptix-gemxpresso.jar or cryptix-gemxpresso-is.jar)

**Note:** Depending on the version of GemXpresso RAD III you have purchased, either cryptix-gemxpresso.jar (full cryptography), or cryptix-gemxpresso-is.jar (limited cryptography) is installed.

Continue with “Chapter 5 - Preparing the Applet for Loading”, which describes how to prepare the applet’s class files for loading into the card simulator.
In this chapter, you will learn how to convert your applet to a format ready for loading into a card using GemXpresso RAD III’s Project Editor and GxpConverter tools.

Introduction

The Class files produced by compiling the OPPurse applet’s source code cannot be directly loaded into a card or the GSE (card simulator). They must first be converted into the appropriate loadable format:

- **CAP** files for a card, stored inside a Java Archive (JAR) file
- **SAP** files for the GSE (card simulator)

You use the following GemXpresso RAD III tools to create loadable files:

- The **Project Editor** to define the target card types for conversion and to configure the conversion environment. The Project Editor generates an XML-format project file.
- The **GxpConverter** to convert the Class files to a loadable format.

Opening the Project Editor Window

Whenever you use one of the supplied templates, GemXpresso RAD III writes the information you supply about the project—the package AID, output paths, and so on—into a GemXpresso project file (.GXP), which is placed in the same directory as the JBuilder or VisualCafé project file. This project file can subsequently be modified—or created from scratch if you did not use a template to create the project—using the Project Editor tool.

The Project Editor can be accessed directly from within the GemXpresso RAD III plug-in for JBuilder or VisualCafé, or it can be started by running the `ProjectEditor.bat` batch file in the `installdir\bin` directory.
To run the Project Editor from within an IDE:

1. Build the OPPurse project to ensure that all menu items on the plug-in menu are active.

2. Choose **GemXpresso RAD > Edit GemXpresso Project**. The Project Editor window is displayed:

   ![Figure 12 - The Project Editor Window](image)

---

28
Preparing the Applet for Loading

The upper part of the Project Editor window contains information about the applet:

- **Project file name**: OPPurse.gxp. The name of the project file.
- **Class path**: .\out. The directory, relative to the directory containing the project file, in which compiled Class files are to be placed.
- **Package name**: com.gemplus.examples.oppurse. The package name specified in the Java source file.
- **Package AID**: A0 00 00 00 18 FF 00 00 00 00 00 00 00 00 01 01, read from the project file.
- **Version**: 1 and 0, the default values for the major and minor version numbers of the applet, read from the project file.
- **In the Default AIDs box**, the Is applet box is selected to indicate that this is an applet. Oppurse.class appears as the Applet name, and the Default applet AID appears as A0 00 00 00 18 FF 00 00 00 00 00 00 00 00 01 02.

Adding Target Card Types

The Target cards for this conversion area of the Project Editor window initially shows a property sheet for each of the card types you selected on the Import GemXpresso Project wizard.

To add support for other card types at this time:

1. Click Select cards.
2. Select one or more card types to add in the Available cards list and click Add.
3. Click OK to return to the Project Editor window. The Project Editor adds the additional card type or types to the OPPurse.gxp project file.

Converting Files

You use the GxpConverter utility to convert a package’s Class files to the CAP file (for a card) or SAP file (for the GSE) formats necessary for loading onto a card or the card simulator, respectively.

All that the GxpConverter utility needs to perform these conversions is the GemXpresso Project file previously created using the Project Editor.
To convert files for the OPPurse applet:

1. In JBuilder or VisualCafé, choose GemXpresso RAD > Convert to display the Convert window:

   ![Convert Window](image)

   **Figure 13 - The Convert Window**

2. Leave the **Compute size of the package** option selected to calculate the size of the applet’s package (executable load file) after conversion.

3. Click **Convert** to begin conversion. Progress messages are displayed in the Messages window of the IDE. The GxpConverter tool is called to perform file conversion for the specified target card types. The output from the GxpConverter tool is displayed in the Messages window of JBuilder or VisualCafé.

4. At the end of the conversion messages, the estimated size in bytes of the applet and the system overhead in bytes for installing the applet are displayed. For example:

   ```
   --> ComputeSize output: Image size of the package: com.gemplus.examples.oppurse is 1464 bytes for the cardname card
   --> ComputeSize output: System overhead for install of the package: com.gemplus.examples.oppurse is 54 bytes
   --> ComputeSize output: Compute memory size completed
   ```

   The GxpConverter creates the following files for each card type:

   - A Java Archive (JAR) file containing CAP files (.CAP) ready to be loaded onto a card.
   - A SAP file (.SAP) ready to be loaded into the GSE (card simulator)
   - An export file (.EXP)
   - A JCA file (.JCA)
Preparing the Applet for Loading

The files are placed in the directory you specified as the output path, with the package name appended in standard Java programming form:

The next chapter shows you how to use these files to load an applet into the GSE (card simulator).

Figure 14 - Location of the Converted Files

The next chapter shows you how to use these files to load an applet into the GSE (card simulator).
Loading the Applet

In this chapter you learn how to load an applet into GemXpresso RAD III's card simulator using the JCardManager. This involves:

- Setting up the JCardManager
- Mutually authenticating the terminal and the card simulator
- Loading the applet
- Installing the applet
- Selecting the applet for use

Setting up the JCardManager

Begin by launching the graphical user interface of the GSE—the GSE GUI—then the JCardManager. With the GSE as the selected target, the JCardManager can be used to issue all commands necessary to load and install the applet.

Starting the GSE (Card Simulator)

To start the GSE GUI:

2. When the **Gemplus Simulation Environment Launcher** window appears, select a card type from the **Card type** list:

![Gemplus Simulation Environment Launcher Window](image)

*Figure 15 - The Gemplus Simulation Environment Launcher Window*

3. Click **Start**. The GSE GUI starts.
Starting the JCardManager

To start the JCardManager:

1. Select Start > Programs > Gemplus Applications > GemXpresso RAD III > JCard Manager. The JCardManager window is displayed:

![Figure 16 - The JCardManager Window](image)

Notice that the JCardManager automatically detects that the GSE is running and the Terminal box displays the name of the terminal.

**Note:** You could equally well have started the JCardManager first, then the GSE GUI.

2. Check that the JCardManager window shows the desired Card and Terminal combination:

![Figure 17 - JCardManager Window with Selected Card and Terminal](image)
Performing Authentication

The first step in any communication session between an off-card entity (typically a client application such as the JCardManager) and a card is to undertake mutual authentication. This involves comparing keys, which are stored in a key file:

- For “IS” versions of cards, this key file contains the same static keys as those stored in the card. Authentication is simply a test that the keys are identical.
- For non-IS versions, the key file contains the value of a “mother key”. By default, the value of this key in the key file is filled with zeros. The real value of the mother key is provided by Gemplus when the card is delivered to you.

The authentication is mutual in the sense that both entities perform the check: the card checks that the keys in the key file match its own and the off-card entity checks that the keys on the card match its own keys stored in the key file.

After successful authentication, a secure communications channel is established between the two entities, along which other commands can be sent.

To perform mutual authentication:

1. In the OP 2.0.1 Commands list, choose the Authenticate command.
2. In the Target file box, browse and check that the correct default key file for the target card type is selected. Key files are named:
   
   \installdir\resources\targets\cardname.properties
3. In the Security Level area of the window, ensure the No Security option is selected to avoid unnecessary ciphering time computation.
4. Ensure the default options Use the default key set version and Use the security domain of the target file are selected.
5. Click Go to run the authentication process. The result of the authentication is displayed in the messages area of the window. The final line should be:

     <- 90 00

     indicating that authentication was successful.

Displaying the Contents of the Card

Once authentication has been successfully completed, you can view which files are stored on the target card. This can be done using the JCardManager’s built-in Card Explorer facility, or using the GSE GUI.
Display the **Gemplus Simulation Environment** window:

![Gemplus Simulation Environment Window](image)

**Figure 18 - The Gemplus Simulation Environment Window**

The most important items of information displayed are:

- The **OpenPlatform** property sheet in the status information area shows status information relating to the card being simulated.
- The **Java Card** property sheet displays the AIDs of the currently selected applet (usually the Card Manager applet) and the default applet.
- No custom files have yet been uploaded to the card, so the only files on the card are system files that cannot be deleted. These are indicated by the ![icon](image) icon.
- The **Package view** displays a list of all system packages.
- The **Applet view** displays a list of all system applets. The **Card Manager** is the only pre-installed system applet on a Java Card-OP card.
Loading OPPurse into the GSE

To load the OPPurse applet into the GSE:

1. Switch to the JCardManager window.

2. Ensure that Simulator is selected as the Target in the JCardManager window.

3. In the OP 2.0.1 Commands list, select the Upload file into a card command.

4. In the Input file box, click the browse button and select the SAP file to upload. (SAP files are the equivalent of the CAP load file format, but for simulated cards). For the converted OPPurse applet, this file is:
   
   ```
   installdir\examples\cardname\oncard\com\gemplus\examples\oppurse\javacard\oppurse.sap
   ```
   
   Click Open to select the file.

5. In the Package AID box, enter the OPPurse applet’s package AID (“A0 00 00 00 18 FF 00 00 00 00 00 00 00 00 00 01 01”).

6. The Security Domain AID box identifies the security domain that will manage the loaded package. For the OPPurse application, the default security domain (the Card Manager applet) is used, so you can leave this box blank.

7. In the Load Parameters box, leave the default value unchanged.

8. Click the Go button to begin loading the file into the selected target. The JCardManager verifies the command’s parameters and begins loading the file:

   ![Figure 19 - Uploading a File with the JCardManager](image)

9. As the loading continues, status message are continuously displayed in the messages area of the JCardManager window. The last few lines indicate that the package has been successfully loaded:

   CMD: Load File
   -> 80 E8 80 0F 42 E1 4F 2A 2F 0E A9 62 3C A7 4A F0 BC 2A C5
   71 7C A567 AF 3C 99 A1 54 C3 BC 62 34 8C 93 8D 92 6C 98 D9
   3C F3 92 B7 A8 AB 3L 1F 07 48 59 50 CD FC 0A 65 AS E5 90 1B
   EB EE FF 01 2B 2A EE 5E AE 14 00 00 00
   <- 80 50 00
   block 18 loaded
   package loaded in 2 s
10. Switch to the GSE GUI window. The fully-qualified name of the uploaded package is now displayed in the Package view. Double-click `com.gemplus.examples.oppurse` to display the applets contained in the package. Double-click the OPPurse applet icon to display the AID of the applet:

![Figure 20 - Uploaded Package in the Card Simulator Window](image)

### Installing OPPurse in the GSE

Notice that the OPPurse package appears in the Package view but does not yet appear in the Applet view as a usable applet. To make the applet usable, you must issue the Install command from within the JCardManager:

1. Switch back to the JCardManager window.
2. In the OP 2.0.1 Commands list, select the Install command.
3. In the Package AID box, enter the package AID used during the installation:
   
   ```
   A0 00 00 00 18 FF 00 00 00 00 00 00 00 00 01 01
   ```

4. In the Applet AID box, enter the applet AID used during the installation:

   ```
   A0 00 00 00 18 FF 00 00 00 00 00 00 00 00 01 02
   ```

   **Tip:** To save time, copy and paste the package AID value and edit it as necessary.

5. In the Instance AID box, enter the applet instance AID:

   ```
   A0 00 00 00 18 FF 00 00 00 00 00 00 00 00 01 02
   ```

   **Note:** For security reasons, the OPPurse applet cannot have multiple instances, so the instance AID must always be the same as the applet AID.
6. Select the **Installation Parameters** option, then type **00 64** in the **Application Specific Param.** box. This is the hexadecimal equivalent of 100, and specifies the default balance of the electronic purse. The maximum value allowed is 27 70h, or 10,000 decimal.

7. Ensure that the **Make this applet SELECTABLE** option is selected.

8. Click **Go** to install the applet. The status of the installation procedure is displayed in the messages area of the window. The return code **90 00** indicates that installation was successful:

```
CMD: InstallApplication
+-80 E6 0C 00 3A 10 A0 00 00 00 18 FF 00 00 00 00 00 00 00 01 01 10 A0 00 00 00 01 02 10 A0 00 00 00 18 FF 00 00 00 00 00 00 00 00 01 02 00 04 C9 02 00 64 00 00
<- 90 00
```

9. Switch back to the GSE GUI window. The **OPPurse** applet is now displayed in the **Applet view**. Notice that, because this is the first instance created of the applet, “**OPPurse#0**” is displayed. Double-click the applet icon to display the AID of the installed applet:

![Figure 21 - The Installed Applet](image)

Notice that the applet’s **State** is now **SELECTABLE**, meaning that it can be selected for use.
Notice also that the status area of the window now displays an OPPurse#0 property sheet, containing a variety of information about the installed applet:

<table>
<thead>
<tr>
<th>JavaCard</th>
<th>OpenPlatform</th>
<th>OPPurse#0</th>
</tr>
</thead>
<tbody>
<tr>
<td>validUI</td>
<td>(private)</td>
<td>boolean</td>
</tr>
<tr>
<td>balance</td>
<td>(private)</td>
<td>short</td>
</tr>
<tr>
<td>securityObject</td>
<td>(private)</td>
<td>object</td>
</tr>
<tr>
<td>secureChannel (package)</td>
<td></td>
<td>byte</td>
</tr>
<tr>
<td>authenticationDone (private)</td>
<td></td>
<td>boolean</td>
</tr>
<tr>
<td>enciphered (private)</td>
<td></td>
<td>boolean</td>
</tr>
<tr>
<td>channelOpened (private)</td>
<td></td>
<td>boolean</td>
</tr>
<tr>
<td>previousState (private)</td>
<td></td>
<td>byte</td>
</tr>
</tbody>
</table>

Figure 22 - OPPurse Property Sheet on the GSE GUI Window

Selecting OPPurse for Use

To select the OPPurse applet:

1. Switch to the JCardManager window.
2. From the OP 2.0.1 Commands list on the JCardManager window, choose the Select command.
3. Enter the Applet AID of your applet, for the OPPurse example: `A0 00 00 00 18 FF 00 00 00 00 00 00 00 00 01 02`
4. Click Go to send the command. The result of the command is displayed both in the messages area of the JCardManager window and in the Card Simulator window:

```text
CMD: Select
-> 00 A4 04 00 10 A0 00 00 00 18 FF 00 00 00 00 00 00 00 00 00 00 01 02 00
< 90 00
```

The next chapter describes how to test the applet by sending commands to it and checking its responses.
Testing the Applet

In this chapter, you learn how use the JCardManager to send APDU commands to the applet and check its responses.

Testing the Applet’s Functionality

The JCardManager can be used to test the functionality of an applet installed in either a real card or a simulated card running in the GSE.

Before testing, you must have completed the following:

- Loading the applet in the card or GSE as described in “Chapter 6 - Loading the Applet”.
- Installing and selecting the applet as described in “Chapter 6 - Loading the Applet”.

You can then send APDU commands to the installed applet in order to test its responses, which are displayed in the messages area of the JCardManager window.

If they are not already running, launch the card simulator and the JCardManager, as described in “Chapter 6 - Loading the Applet”.

Sending User-defined APDU Commands

Applet commands take the form of user-defined APDUs. The applet must be programmed to understand these APDU commands and respond appropriately. The OPPurse applet, for example, supports the following custom APDU commands:

- Get Balance
- Debit
- Credit
- Get PIN
- Verify PIN

These commands are used in the examples that follow.
Getting the Current Balance of the Electronic Purse

The `getBalance` APDU returns the current balance of the electronic purse that is managed by the OPPurse applet. The initial balance of the electronic purse was set when the applet was installed (see “Installing OPPurse in the GSE” on page 39).

To send a `getBalance` APDU to the OPPurse applet:

1. In the OP 2.0.1 Commands list, select the Send APDU command.
2. In the Command Name list, type `getBalance`.
3. Enter the following APDU parameters:

<table>
<thead>
<tr>
<th>APDU Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLA</td>
<td>90</td>
</tr>
<tr>
<td>INS</td>
<td>10</td>
</tr>
<tr>
<td>P1</td>
<td>00</td>
</tr>
<tr>
<td>P2</td>
<td>00</td>
</tr>
<tr>
<td>Lc</td>
<td>00</td>
</tr>
<tr>
<td>Le</td>
<td>02</td>
</tr>
<tr>
<td>Data to Send</td>
<td>Leave empty</td>
</tr>
</tbody>
</table>

Table 4 - `getBalance` APDU Parameters

4. Click Go to send the APDU command to the application. The result is displayed in the messages area of the JCardManager window and in the Card Simulator window, for example:

```
======================== Get Balance
CMD: Get Balance
-> 90 10 00 00 02 00 64 02
<- 00 64 90 00
```

The result of the command is “00 64”, which is following by the success code “90 00”. The value “00 64” is the initial value of the electronic purse that was assigned when the applet was uploaded.
Crediting the Electronic Purse

From the security point of view, crediting an account is a far more sensitive operation than obtaining the balance of or debiting an account. For this reason, the OPPurse applet requires additional security conditions to be satisfied before the electronic purse can be credited:

- An Authenticate command must be issued against the currently selected application (OPPurse).
- The PIN code must be initialized if it has not already.
- The PIN code must be submitted with a VerifyPIN APDU command.

Only if these conditions are satisfied, can a Credit APDU command be sent to the applet specifying the amount to credit to the existing balance.

Step 1: Authenticating the OPPurse Application

To authenticate the OPPurse application:
1. In the OP 2.0.1 Commands list, select Authenticate.
2. Check that the selected card type matches the key file selected in the Target File box (cardname.properties)
3. In the Application Selection area, select Currently selected application.
4. Click Go.

A successful response ("90 00") indicates that authentication was successful.

Step 2: Initializing the PIN Code

The next step is to initialize the card’s PIN code. For a real card, this is only necessary once in the card’s life cycle. For the GSE, by contrast, it must be performed once for each session.

To initialize the PIN code:
1. Select the Change PIN command from the OP 2.0.1 Commands list.
2. Select the Change PIN button.
3. In the New PIN code box, enter a 4-digit PIN code for the card, for example “1111”.
4. Click Go to send the command. The response should be:
   
   <- 90 00

   indicating that the PIN code was changed successfully.
Step 3: Submitting a PIN Code to the OPPurse Applet

VerifyPIN is a custom APDU defined in the OPPurse applet code that is used to check that a PIN code submitted with the command matches the value of the PIN code stored on the card.

To issue the VerifyPIN APDU command:
1. In the OP 2.0.1 Commands list, select Send APDU.
2. In the Command Name list, type VerifyPIN.
3. Enter the following APDU parameters:

<table>
<thead>
<tr>
<th>APDU Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLA</td>
<td>90</td>
</tr>
<tr>
<td>INS</td>
<td>16 (identifies the VerifyPIN APDU command)</td>
</tr>
<tr>
<td>P1</td>
<td>00</td>
</tr>
<tr>
<td>P2</td>
<td>00</td>
</tr>
<tr>
<td>Lc</td>
<td>04 (calculated automatically from the number of bytes in the Data to send field)</td>
</tr>
<tr>
<td>Le</td>
<td>00 (no data is expected in response)</td>
</tr>
<tr>
<td>Data to send</td>
<td>&quot;1111&quot; (enclose the value in quotation marks to indicate a decimal value)</td>
</tr>
</tbody>
</table>

Table 5 - VerifyPIN APDU Parameters

4. Click Go to send the command. The response should be:

```
========================
verifyPIN
CMD: verifyPIN
-> 90 16 00 00 04 31 31 31 31 00
<- 90 00
```

Step 4: Sending a Credit APDU to the OPPurse applet

Once all security conditions have been satisfied, the electronic purse can be credited.

To send a Credit APDU command:
1. In the OP 2.0.1 Commands list, select Send APDU.
2. In the Command Name list, type Credit.
3. Enter the following APDU parameters:

<table>
<thead>
<tr>
<th>APDU Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLA</td>
<td>90</td>
</tr>
<tr>
<td>INS</td>
<td>14 (identifies the Credit APDU command)</td>
</tr>
<tr>
<td>P1</td>
<td>00</td>
</tr>
<tr>
<td>P2</td>
<td>00</td>
</tr>
<tr>
<td>Lc</td>
<td>02 (calculated automatically)</td>
</tr>
<tr>
<td>Le</td>
<td>00 (no response data is expected)</td>
</tr>
<tr>
<td>Data to send</td>
<td>14 (value to be credited to the balance on two bytes)</td>
</tr>
</tbody>
</table>

Table 6 - Credit APDU Parameters

4. Click Go to send the command. The response should be:

```
=======================  Credit
CMD: Credit
-> 90 14 00 00 02 00 14 00
<- 90 00
```

5. Issue a new getBalance APDU command to check that the amount has been credited correctly. Set Lc to zero and Le to “2” to indicate that two bytes of response data are expected. The response should be:

```
=======================  getBalance
CMD: getBalance
-> 90 10 00 00 00 02
<- 00 78 90 00
```

This shows that the balance is now 78h, which is correct (64h initially, plus the 14h that was credited).

**Debiting the Electronic Purse**

To debit the electronic account, you issue a Debit APDU command.

To send a Debit APDU command:

1. In the OP 2.0.1 Commands list, select Send APDU.
2. In the Command Name list, type Debit.
3. Enter the following APDU parameters:

<table>
<thead>
<tr>
<th>APDU Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLA</td>
<td>90</td>
</tr>
<tr>
<td>INS</td>
<td>12 (identifies the Debit APDU command)</td>
</tr>
<tr>
<td>P1</td>
<td>00</td>
</tr>
<tr>
<td>P2</td>
<td>00</td>
</tr>
<tr>
<td>Lc</td>
<td>02 (calculated automatically)</td>
</tr>
<tr>
<td>Le</td>
<td>02 (request that the new balance be returned on two bytes)</td>
</tr>
<tr>
<td><strong>Data to send</strong></td>
<td>30 (value to be debited from the balance on two bytes)</td>
</tr>
</tbody>
</table>

Table 7 - Debit APDU Parameters

4. Click Go to send the command. The response should be:

```
=======================  debit
CMD: debit
-> 90 12 00 00 02 00 30 02
<- 00 48 90 00
```

Check that this balance is correct (78h - 30h = 48h).

**Saving the Commands List**

The JCardManager provides a facility to allow you to save previously issued APDU commands and parameters to a file. The next time an applet is loaded, you can then restore the commands file, select a command from the command name list, and the parameters are entered automatically.
To save the custom APDU commands for the OPPurse applet:

1. In the Command Parameters area, click Edit Command List. The Edit Command List window is displayed:

![Edit Command List Window]

Figure 23 - Edit Command List Window

2. Remove any duplicate commands from the list by selecting the command and clicking Remove. You can select multiple commands by pressing and holding down the Shift or Ctrl key.

3. Click Save AMF.

4. Navigate to the `installdir\examples\cardname\OPPurse` directory and give the command list file a name, for example, `OPPurse_commands.amf`. Click OK to save the command list.
Next Steps...

The GemXpresso RAD III environment contains all the components for the complete development of applets and, if required, their corresponding client applications. In this Getting Started guide you have had the chance to work through the basic tools, with a provided example.

You can now proceed to the next steps:

- Developing your own code
- Using the tools to their fullest capacity, especially the JCardManager and GSE (card simulator)

The two main problems in developing applications for smart cards have been solved by GemXpresso RAD III, namely:

- Restrictive memory space
- Debugging difficulties

GemXpresso RAD III offers Java Card developers components for:

- Efficient source code design and development of Java card applets and their auxiliary client applications (templates, project definition).
- Debugging capabilities, from the Java IDE (VisualCafé, JBuilder, or any other JDK1.2 compatible environment) together with those of the GemXpresso Simulation Environment (GSE), which simulates the applet running in a card.
The GemXpresso Family of Products

The GemXpresso RAD III environment has been developed in conjunction with the GemXpresso family of Java Card-OP compatible smart cards, particularly:

- the multi-application GemXpresso 211 cards
  - GemXpresso 211
  - GemXpresso 211IS
  - GemXpresso 211 PK
  - GemXpresso 211 PK IS
- GemXpresso Lite cards
- GemXpresso Pro R3 cards.

All cards are fully compliant with the Java Card and Open Platform standards. To know which cards are compatible with which versions of the standards, refer to Table 1, "JavaCard and OP Specification Compatibility", on page 2.
Terminology

Abbreviations

AID  application identifier
APDU  application protocol data unit
API  application programming interface
ATF  APDU trace format
ATR  answer to reset
CAD  card acceptance device
CAP  card applet package
CBC  cipher block chaining
CLA  class byte
CPLC  card production life cycle
DES  data encryption standard
ECB  electronic code book
EXP  prefix for Java Card export files
FSD  full serialized data
GSE  Gemplus simulation environment
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GXCF</td>
<td>XML-based script format</td>
</tr>
<tr>
<td>ICV</td>
<td>initial chaining vector</td>
</tr>
<tr>
<td>IDE</td>
<td>integrated development environment</td>
</tr>
<tr>
<td>INS</td>
<td>instruction byte</td>
</tr>
<tr>
<td>IS</td>
<td>international sample (card version)</td>
</tr>
<tr>
<td>ISO</td>
<td>International Standards Organization</td>
</tr>
<tr>
<td>JAR</td>
<td>Java archive file</td>
</tr>
<tr>
<td>JCA</td>
<td>Java card assembler</td>
</tr>
<tr>
<td>JCRE</td>
<td>Java card runtime environment</td>
</tr>
<tr>
<td>JCVM</td>
<td>Java card virtual machine</td>
</tr>
<tr>
<td>JDK</td>
<td>Java development kit</td>
</tr>
<tr>
<td>JVM</td>
<td>Java virtual machine</td>
</tr>
<tr>
<td>Lc</td>
<td>data length</td>
</tr>
<tr>
<td>Le</td>
<td>expected length of data to be returned</td>
</tr>
<tr>
<td>MAC</td>
<td>message authentication code</td>
</tr>
<tr>
<td>OCF</td>
<td>opencard framework</td>
</tr>
<tr>
<td>OP</td>
<td>open platform</td>
</tr>
<tr>
<td>PC/SC</td>
<td>personal computer/smart card</td>
</tr>
<tr>
<td>PIX</td>
<td>proprietary identifier extension</td>
</tr>
<tr>
<td>PK</td>
<td>public key</td>
</tr>
<tr>
<td>RAD</td>
<td>rapid application development</td>
</tr>
<tr>
<td>RID</td>
<td>registered identifier</td>
</tr>
<tr>
<td>SAP</td>
<td>simulator applet package (GSE-proprietary load format)</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>SSD</td>
<td>system serialized data</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>transmission control protocol/internet protocol</td>
</tr>
<tr>
<td>VM</td>
<td>virtual machine</td>
</tr>
</tbody>
</table>
# Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applet</td>
<td>In Java Card terminology, a Java Card applet is an independent Java application loaded into a Java Card.</td>
</tr>
<tr>
<td>Application Identifier</td>
<td>A string of between 5 and 16 bytes that identifies a package or an application in a card and which corresponds to the naming scheme defined in ISO7816-5. It may contain a registered application provider number. If it contains no application provider number, then this identification may be ambiguous.</td>
</tr>
<tr>
<td>Application Protocol</td>
<td>Standard communication messaging protocol between a card acceptance device and a smart card.</td>
</tr>
<tr>
<td>Application Provider</td>
<td>The entity that owns an application and is responsible for the application’s behavior.</td>
</tr>
<tr>
<td>ATF file</td>
<td>A Gemplus APDU Trace Format script file.</td>
</tr>
<tr>
<td>ATR</td>
<td>When a card is inserted into a card reader, it stimulates a contact, which provokes the terminal to reset itself by sending a reset signal to the card. The card’s response is called an answer to reset (ATR). The ATR is described in two standards: ISO 7816-3: Electronic signals and transmission protocols, which defines the two possible low-level communication (or transport) protocols between the terminal and the card. It is strongly advised to refer to this standard for details. ISO 7816-4: Interindustry commands for interchange, which defines a set of standard commands for smart cards, as well as a hierarchical file system structure for cards. These commands are the basis of most existing card protocols.</td>
</tr>
<tr>
<td>Bytecode</td>
<td>A VM instruction code as a sequence of binary bytes.</td>
</tr>
<tr>
<td>CAP files</td>
<td>A file format that can be loaded into a card. CAP files are generated by the GxpConverter tool.</td>
</tr>
<tr>
<td>Class files</td>
<td>A compiled Java code file.</td>
</tr>
<tr>
<td>Client application</td>
<td>An application that requests services from a server application—typically an applet in the Java Card environment.</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Context</td>
<td>An image, stored in a file, of the current contents of a card, including the AID and life cycle status of each package and applet on the card. Loading a context into the GSE (card simulator) is equivalent to inserting a personalized smart card into a card reader. See also “Full context” and “System context”.</td>
</tr>
<tr>
<td>Conversion</td>
<td>A post-compilation step necessary to convert Java Class bytes into a form (.CAP files) understood by the card.</td>
</tr>
<tr>
<td>Converter</td>
<td>A utility provided by Sun to convert Java Class files into a form understood by the card.</td>
</tr>
<tr>
<td>Full context</td>
<td>An image, stored in a file, of system packages and applets, custom packages and applets stored on a card, for example, packages that you have uploaded and installed. See also “Context”.</td>
</tr>
<tr>
<td>GSE</td>
<td>Gemplus simulation environment. A software environment that simulates a Java Card context. Used for testing applet functionality before loading the applet into a card.</td>
</tr>
<tr>
<td>JAR file</td>
<td>Java archive file. A compressed file containing Java Class files, or CAP load files. Used by the GxpLoader, for example.</td>
</tr>
<tr>
<td>Java Card-OP applet</td>
<td>An applet that is fully compliant with the Java Card 2.1.1 and Visa’s Open Platform 2.0.1 specifications.</td>
</tr>
<tr>
<td>Java-OP Card</td>
<td>Smart cards that are fully compliant with both the Java Card 2.1.1 and Visa’s Open Platform 2.0.1 specifications.</td>
</tr>
<tr>
<td>JCA file</td>
<td>A java card assembler file, generated by the GxpConverter or Sun’s Converter tool.</td>
</tr>
<tr>
<td>JCardManager</td>
<td>A GemXpresso RAD III tool that simulates a client application, enabling a user to exchange commands with an applet installed in either a card or the GSE.</td>
</tr>
<tr>
<td>Key index</td>
<td>Identifies a key within a key set.</td>
</tr>
<tr>
<td>Key set</td>
<td>A set of encryption keys used to authenticate the owner.</td>
</tr>
<tr>
<td>Key version</td>
<td>A number between 0 and 127 identifying the key version to use.</td>
</tr>
<tr>
<td>Load File</td>
<td>The physical data files that are uploaded to Java-OP cards to modify the card’s contents.</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
</tr>
<tr>
<td>MAC (message authentication code)</td>
<td>A symmetric cryptographic transformation of data that provides data origin authentication and data integrity.</td>
</tr>
<tr>
<td>GxpLoader</td>
<td>A GemXpresso RAD III tool used to convert, load, and install files into target cards or the GSE (card simulator) in one step.</td>
</tr>
<tr>
<td>Package</td>
<td>A Java term to describe a collection of related classes and interfaces. A package can contain, for example, a client application, or several applets.</td>
</tr>
<tr>
<td>PC/SC</td>
<td>The PC/SC (personal computer/smart card) Workgroup was formed in May 1996 to resolve interoperability issues between PCs and smart cards. Members include Gemplus, IBM, Sun Microsystems, Toshiba, Groupe Bull, Hewlett-Packard, Microsoft, Schlumberger, and Siemens Nixdorf.</td>
</tr>
<tr>
<td>Project</td>
<td>A collection of one or more applets, together with the necessary system classes, with or without libraries. Also used to describe an IDE (VisualCafé or JBuilder) project.</td>
</tr>
<tr>
<td>RID</td>
<td>The first five bytes of an AID, registered with ISO, indicating the Application Provider.</td>
</tr>
<tr>
<td>SCR file</td>
<td>A Sun script file.</td>
</tr>
<tr>
<td>Scripts</td>
<td>A list of commands. Gemplus scripts (.atf) or Sun scripts (.scr) can be exchanged with a card or the GSE.</td>
</tr>
<tr>
<td>Size tool</td>
<td>A tool used to calculate the amount of memory that an applet occupies in a card. Provided by Sun as a command-line utility, in GemXpresso RAD III the Size tool is integrated in the GxpConverter utility.</td>
</tr>
<tr>
<td>System context</td>
<td>An image, stored in a file, containing details of all system packages and applets stored on a card. See also &quot;Context&quot;.</td>
</tr>
<tr>
<td>Target</td>
<td>The entity in which a Java Card applet or package is eventually installed. The target can be either the GSE (Gemplus simulation environment) or a specific type of card in a card reader.</td>
</tr>
<tr>
<td>Terminology</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Target file</td>
<td>The target file contains the characteristics of the card. It is required by the GxpLoader tool.</td>
</tr>
<tr>
<td>Template</td>
<td>A predefined project type provided by GemXpresso RAD III for the JBuilder and VisualCafé IDEs. Each template provides a wizard to help you specify project-related information. Some templates generate ready-to-compile skeleton Java code, to which the developer need only add the applet or client application’s functionality.</td>
</tr>
</tbody>
</table>
For More Information

GemXpresso RAD III comes with the following documents:

- *GemXpresso RAD III Version 3.2 Command Reference*, which describes the OP commands supported by individual cards. A different version is available for each card type.

Standards and Specifications

*Open Platform Card Specification Version 2.0.1’, April 7th 2000
Java Card 2.1.1 Virtual Machine Specification

Recommended Reading

For more information about Java applet development for smart cards, see:
