OpenCard Framework
General Information Web Document
OpenCard Framework
General Information Web Document
OpenCard Framework — General Information Web Document

Second Edition

October, 1998

Copyright OpenCard Consortium.

This Web Document provides information about the OpenCard Framework.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

IBM welcomes your comments. Please address your comments or questions to the following address:

IBM Deutschland Entwicklung GmbH
Information Development, Dept. 3248
Schoenaicher Strasse 220
71032 Boeblingen
Germany

FAX (Germany): 07031 16 3456
FAX (Other Countries): (+49) 7031 16 3456

Questions regarding this publication should be sent to:

- IBM Mail Exchange: DEIBMBM9 at IBMMail
- Internet e-mail: idepubs@vnet.ibm.com

Make sure to include the following in your comment or note:

- Title and order number of this Web Document
- Page number or topic related to your comment

Technical questions concerning OCF should be posted to the OCFmailing list at http://www.opencard.org/

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.
Trademarks and service marks

These terms are trademarks or service marks of the IBM Corporation in the United States or other countries:

- IBM
- AIX

These terms are trademarks of other companies:

<table>
<thead>
<tr>
<th>Term</th>
<th>Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCF</td>
<td>OPENCARD CONSORTIUM</td>
</tr>
<tr>
<td>OPENCARD</td>
<td>OPENCARD CONSORTIUM</td>
</tr>
<tr>
<td>SOLARIS</td>
<td>SUN MICROSYSTEMS, INC.</td>
</tr>
</tbody>
</table>

JAVA and all JAVA-based trademarks and logos are trademarks or registered trademarks of SUN MICROSYSTEMS, INCORPORATED, in the United States and/or other countries.

WINDOWS, WINDOWS NT, and WINDOWS 95 are registered trademarks of MICROSOFT CORPORATION in the United States and/or other countries.

Other company, product, and service names may be trademarks or service marks of others.
## Contents

**Trademarks and service marks** ........................................... iii

**Figures** ........................................................................... vii

**Preface** ............................................................................... ix

About this document ......................................................... ix

For whom this document was written ................................. ix

About this product ............................................................... ix

Additional information ........................................................ ix

Font conventions used in this document ............................... ix

**Chapter 1. OpenCard Framework — General information web document** ................................. 1

Abstract .............................................................................. 1

Introduction ......................................................................... 1

OpenCard objectives .......................................................... 2

OpenCard architecture ....................................................... 2

The CardTerminal layer ...................................................... 3

The CardService layer ....................................................... 3

The ApplicationManagement component ........................... 4

Configuration ..................................................................... 4

Scope ................................................................................. 4

Summary ............................................................................ 5

Availability .......................................................................... 6

**Chapter 2. Internet Broker Demo** ...................................... 7

Introduction ......................................................................... 7

Carrying out a stock order ................................................. 7

Security capabilities ......................................................... 7

Issuing a card ..................................................................... 8

Obtaining information from a card .................................... 8

Application tour .................................................................. 8

Downloading the Internet Broker Demo ............................. 8

Starting the Internet Broker Demo ..................................... 8

The scenario ...................................................................... 8

Programming the application ............................................ 11

Conclusion ......................................................................... 13
Figures

1. Main parts of the OpenCard Framework architecture ................. 3
2. Summary of benefits provided by OpenCard Framework .............. 5
3. The scenario of the Internet Broker Demo. .......................... 9
4. The protocol used by the Internet Broker Demo. ..................... 10
Preface

About this document

The purpose of this web document is to introduce you to

- the basic architecture, concepts, and objectives of the OpenCard Framework (OCF),
- the numerous benefits which the OpenCard Framework offers to the various different players in the smart card industry,
- the “Internet Broker” example application.

For whom this document was written

This web document was written for

- anyone who wants an overall perspective on the OCF architecture
- anyone interested in the considerable advantages OCF offers to smart card holders, smart card issuers, card terminal vendors, etc.,
- application programmers wishing to familiarize themselves with basic OCF concepts and code and to see just how much easier programming is made. In this context, the Internet Broker Demo at the end of this document should prove especially interesting to them.

About this product

The term OpenCard Framework refers to the OpenCard architecture and the appropriate class libraries (also referred to as OCF Reference Implementation).

Additional information

In addition to this General Information Web Document, three other sources of information are available:

- the OpenCard Framework 1.0 Programmer’s Guide and
- the OCF Reference Implementation downloadable from the Internet website at http://www.opencard.org/ from which you can obtain documentation, binaries, class files, and source code.
- the OCF mailing list at http://www.opencard.org/ for posting OCF-related questions and communicating with other parties interested in OCF.

Font conventions used in this document

*Italics* are used for

- special terms and abbreviations (when used for the first time);
- the titles of documents, papers, books, and other publications.

*Boldface* is used for

- emphasis.

Typewriter font is used for
• anything (in particular sample code, the names of applications, classes, methods, etc.) that would be typed verbatim.

**Boldface typewriter font** is used to mark
• OCF-specific source code.

**SMALLCAPS** are used for
• names (such as company names, product names, registered trademarks, etc.).
Chapter 1. OpenCard Framework — General information web document

Abstract

Standardized application platforms and standardized, easy-to-use frameworks to communicate with smart cards of any flavor and card terminals of any make are key factors for implementing smart card-enabled solutions and smart card-based services. OPEN CARD FRAMEWORK (OCF) capitalizes on the broad, cross-platform benefits of JAVA, providing an open architecture and a set of common APIs (Application Program Interfaces) geared for this purpose.

OPEN CARD FRAMEWORK provides benefits to all of the players in the smart card industry:

With OPEN CARD FRAMEWORK

- **Smart card solution providers** can develop, build, and deploy smart card-aware products without having to worry about platform, card terminal, or smart card-specific interfaces.

- **Card and card terminal manufacturers** can compete in terms of functionality and gain access to new market segments because the orientation towards a single vendor is reduced.

- **Card issuers** benefit from OCF by being able to select from a broader range of smart card-aware solutions.

- **Card holders** enjoy a broader range of possible uses to which smart cards can be put and are afforded greater convenience and security.

Introduction

The Internet's impact upon society, everyday life, and work is often compared to that of the telephone at the beginning of the 20th century. In very much the same way that businessespeople and their customers started using the telephone to conduct business and deal with day-to-day issues back then, we are now switching from the telephone to the Internet. The more we rely on Internet technology — such as World Wide Web (WWW), electronic mail, and Internet telephony — the more dependent we are on having some means of making our communications secure and authenticated. At the same time, mobility is gaining in importance. General network-centric applications — where resources are located throughout the Internet and access to them is possible from any location — require authenticated access and secured transactions. We hence need technology that will provide security without limiting mobility.

*Smart cards* represent an ideal solution: They are small and easy to carry around, yet have enough processing power and data storage to store user profiles, encrypt and decrypt data, and support electronic commerce applications. But unfortunately, the *APIs* which support smart cards and card terminals are typically of a proprietary and complex nature — making it difficult to write general-purpose applications. Moreover, once you’ve selected a particular smart card, your choice of card terminals and platforms is severely limited. This is because the *APIs* and shared libraries (*DLLs*, that is: dynamic link libraries) available for any particular smart card will usually work only with a very specific card terminal (at most a small set of card terminals from the same card terminal vendor).
Application developers cannot foresee the environment in which their smart card-enabled application (possibly downloaded from the Internet by card holders) will be executed; neither the smart card nor the card terminal can be known in advance.

This situation is clearly not satisfactory for smart card solution developers.

The **Java**-based **OpenCard Framework**
- serves as an application platform for smart cards and card terminals of any flavor, and
- provides standardized, easy-to-use high-level APIs.

### OpenCard objectives

When looking at smart card applications from the point-of-view of application developers, at least three parties with a crucial role can be identified:

**Card Terminal Vendors**
The card terminal vendors provide us with the actual card readers (the “card acceptance devices” or card terminals in OCF parlance). Each vendor carries a more or less broad spectrum of card terminals, ranging from the very simple units to more sophisticated models (featuring perhaps displays, PIN pads, or even biometric input devices). Unfortunately, card terminal vendors have not yet agreed on a common standard interface, and a number of different protocols are allowed for.

**Card Operating System Providers**
There are also numerous competing companies offering various different card operating systems and APIs. This results in a wide variety of commands and response codes.

**Card Issuers**
These are the actual entities which issue smart cards to customers, and it is they who decide where to place the card-resident applications on the cards they issue to their customers. Thus, code locations can vary considerably.

The goal is to reduce dependence upon each of these parties (as well as upon the platform providers). Ultimately, neither the kind of card terminal nor the brand of smart card would then be of any relevance.

Thus, we can identify the following objectives for OCF:
- card terminal vendor independence,
- card operating system provider independence, and
- card issuer independence.

In addition, we want OCF to be easy to use and expandable.

### OpenCard architecture

To address the requirements and objectives described in the previous section, the core architecture of the **OpenCard Framework** features two main parts: the CardTerminal layer and the CardService layer. Further, the problem of card issuer independence is addressed separately by OCF’s ApplicationManagement component.
The CardTerminal layer

The CardTerminal layer provides access to physical card terminals and inserted smart cards for which appropriate OCF-compliant drivers are made available by manufacturers. Also included are JAVA APIs for accessing PC/SC-supported card terminals. Hence, the CardTerminal layer enables OCF to handle the broad range of card terminals in use — now and in the future.

The CardService layer

The CardService layer makes it possible for the OpenCard Framework to deal with the wide variety of card operating systems in existence and the various different functions they may offer.

Among OCF’s many CardServices are the FileAccessCardService and the SignatureCardService.

Figure 1. Main parts of the OpenCard Framework architecture
The FileAccessCardService provides a fairly complete set of interfaces and (abstract) classes making the ISO file system’s functions available to the programmer. As with the rest of the framework, these classes and interfaces have been designed to fit seamlessly into the existing JAVa programming model. Thus, if you’re familiar with JAVa’s way of dealing with files and carrying out input-output operations with them, you’ll find the FileAccessCardService very easy to use for your purposes.

The SignatureCardService offers you methods to create and verify digital signatures based on such public key algorithms as RSA and DSA. Additional services allow private and public keys to be imported to the smart card or key pairs to be generated directly on the smart card.

The ApplicationManagement component

With the introduction of multiple applications which can be loaded onto a single smart card, new dependencies — such as which applications are available on the card or where an application and its data are physically located on the card — are created. This is where the ApplicationManagement component comes in. The ApplicationManagement component is capable of

- locating and selecting card-resident applications on any given smart card,
- listing the applications which a particular smart card supports,
- installing and uninstalling applications on smart cards,
- blocking and unblocking applications on smart cards,

and thus solves the problem of card issuer dependency.

Configuration

Although configuration is an important issue for any framework providing access to system-external resources (such as card terminals and smart cards), it was decided not to address configuration issues within the context of OCF.

The reasoning behind this decision is that OCF will be employed on a wide variety of platforms, ranging from PCs to Network Computers to set-top-boxes to smart phones and possible other platforms. Many of these platforms do not offer a file system that could be use to store configuration data — in fact, each platform will probably differ markedly from the next, and it would exceed the scope of OCF to define a configuration sub-framework. Instead, OCF relies on the respective platform provider to offer some kind of configuration service for setting up system properties, and OCF uses these system properties to obtain configuration information (such as what card terminals and card services are available).

Scope

OCF is perfectly suited to support all JAVa-enabled platforms such as PERSONAL COMPUTERS (PCs), NETCOMPUTERS (NCs), or any type of smart card-enabled device such as automatic teller machines (ATMs), point-of-sales terminals, set-top boxes, and emerging hand-held devices. It also enables interactions with existing PERSONAL COMPUTER / SMARTCARD (PC/SC) 1.0 supported card terminal.

The prerequisites for OCF are as follows:
Any JAVA 1.1 compliant platform such as AIX, LINUX, SOLARIS, WINDOWS ‘95, and WINDOWS NT.

There are no prerequisites for use with IBM NCs because OCF is delivered with the operating system.

Summary

This brief introduction has familiarized you with the OpenCard Framework and its architecture. We’ve shown you how OpenCard Framework provides benefits to all of the players in the smart card industry:

- **Vendor independence**: Smart card solution providers can choose cards and terminals from different suppliers and are no longer tied to one particular vendor.

- **Increased clientele**: Smart card solution providers as well as card and card terminal suppliers can access new market segments and can reach more customers.

- **Lower development costs**: Application developers save the extra cost of porting their applications to different platforms and benefit from lower skill requirements to accomplish a given task.

- **Improved competitiveness**: Smart card solution providers as well as card and card terminal suppliers can compete in terms of functionality and are less vulnerable to the predominance of a single vendor.

- **Less development effort**: Application developers inherit functionality conferred by the framework, which reduces their development effort.

- **Improved time-to-market**: Application developers profit from shorter development cycles by programming against a high-level API.

**Figure 2. Summary of benefits provided by OpenCard Framework**

OCF’s architecture and reference implementation provide both the blueprint and the scaffolding for tapping the potential harbored by smart card technology: An environment in which computers, smart cards, card and card terminal suppliers, and solution providers all interact together cooperatively.

With OpenCard Framework, **smart card solution providers** can build and deploy smart card-aware products without having to worry about platforms, terminals, or smart card-specific interfaces. They can therefore concentrate on developing actual solutions, secure in the knowledge that these solutions will be portable and inter-operable on equipment from a variety of different suppliers and JAVA-supporting platforms.

OpenCard Framework enables **card and card terminal manufacturers** to compete in terms of functionality and gain access to new market segments by reducing the orientation towards a single vendor. Thanks to the functionality provided by OCF’s open architecture, they will also have less developmental effort.
Card issuers benefit from OCF by being able to select from a broader range of smart card-aware solutions. They can also select from a larger number of other potential card issuers to cooperate in making specific applications available on several different distinguished cards.

And finally, with OPENCARD FRAMEWORK, card holders are ultimately the biggest winners. This is because OCF expands the range of possible uses to which smart cards can be put and improves their convenience and security.

Availability

All of the OPENCARD FRAMEWORK can be downloaded from the OPENCARD CONSORTIUM’s WWW site (http://www.opencard.org/): Binaries (JAR files), source code, and documentation.

You can also download example applications such as the OCF—based Internet Stock Broker Demo discussed below. This example application uses such CardServices as the SignatureCardService and the FileAccessCardService to digitally sign and verify orders for buying stocks over the Internet.

Also included in the binary code is a PC/SC adaptation layer utilizing MICROSOFT’s PC/SC code to access PC/SC—conform card terminals on the WINDOWS 95/NT INTEL platform. PC/SC Migration Interface Readers are also supported.

Further, the OPENCARD web site offers a platform for interested parties to post OPENCARD-related questions and exchange information via a mailing list. The subscription process is posted under "Contacting Us..." on the OPENCARD homepage.
Chapter 2. Internet Broker Demo

Introduction

Background and Positioning

The Internet Broker Demo is an example application intended to show you, the application programmer, how much easier it is to implement certain applications using OCF. For further information, we suggest that you also take a look at the OCF entry page http://www.opencard.org/. There you’ll find other interesting information about OPENCARD and the OPENCARD FRAMEWORK.

Ordinarily, to program an application like the Internet Stock Broker without OCF, you would have to know the specifications of both the card and the card terminal. Further, you would have to write code which communicates only with this specific terminal and this specific card.

But with the OPENCARD FRAMEWORK, you can work much easier and faster.

The following demo application is a technology demo and illustrates some of the major features supported by the OPENCARD FRAMEWORK. However, it is not a complete Secure Authentication solution.

The Internet Broker Demo has four major functions:
1. Carrying out a stock order
2. Demonstrating security capabilities
3. Issuing a card
4. Obtaining information from a card

Carrying out a stock order

In this sample application, a customer can buy some shares of the 12 founding members of the OPENCARD CONSORTIUM by carrying out a stock order and using a smart card to generate digital signatures.¹

Security capabilities

With this demo, three attempted security breaches can be simulated (in real life, these security breaches would be carried out without the customer’s knowledge or approval). The “United Internet Criminals Inc.” is a fictitious organization which launches simulated security breaches in the customer’s data link and which can
- read his order data (only for non-encrypted (“clear text”) transfers) —also called “interception”
- change his order data (only for non-encrypted (“clear text”) transfers) —also called “forgery”
- place the same order repeatedly —also called “replaying”

¹ Since not all of the founding members are traded as shares, some have been included in the demo with fictitious share values. These companies are marked with brackets around their names.
Issuing a card

Before issuing a new card to a customer, you have to copy the portion of the application data that is unique to each card. In this case, the card holder’s name and his e-mail address have been placed on the card.

Obtaining information from a card

This function allows the application programmer to read the card holder’s data written when the card was issued.

Application tour

This section gives you a quick tour of the Internet Broker Demo.

Downloading the Internet Broker Demo

You can download the Internet Broker Demo using the OPENCARD download tool (http://www.opencard.org/download/code/1.0/IB10Demo.exe).

Starting the Internet Broker Demo

To start the Internet Stock Broker application in MICROSOFT WINDOWS '95 or WINDOWS NT, click on the "run Internet Broker Demo" icon. To install the Internet Broker demo, refer to the readme file README.html accompanying the demo.

When you start the demo, you should see four windows:
  • the main "Internet Broker" window
  • the "Signature Card Issuer" window
  • the "Internet Brokerage Server" window
  • and the "United Internet Criminals Inc." window.

These are the main parts of the Internet Broker Demo. In the trace window, you can see what’s going on with OCF during the request; you can also see the APDUs (Application Protocol Data Units) being sent to and from the card. The trace shows you the interactions between OCF and the card. When looking at the trace later on, you’ll see how easy it was to program the “wait for card” function with the number of low-level interactions necessary to accomplish this function.

The scenario

This scenario is intended to describe the advantages of smart cards for you, the card holder, for example: to place stockmarket orders using a smart card to generate digital signatures.
Let’s assume that you’re sitting in front of your Network Computer connected to your favorite broker over the Internet. Let’s also assume that you’re also using your smart card to generate digital signatures for authorizing stock orders.

**Placing an order**

Let’s say that you’ve connected to your favorite Internet broker and want to place a new order over the Internet, for example to buy ten additional shares of IBM.

- When you use your smart card, your broker can be sure that you, the smart card holder, have placed the order — and not somebody else who was watching you at your keyboard the last time you entered your password.
- Your smart card is portable. You can put it in your pocket and have it with you whenever you need it.
- The merchant / broker can be certain that the transaction is valid and won’t be repudiated later.

After selecting the shares you want to purchase, click on the “Send Order” button. You’ll then be asked for your password, which is on your smart card in encrypted form. Enter password as your demo password.

For application programmers interested in the “nuts and bolts” of how this order was processed:

After the customer clicked on the “Send Order” button, the order was generated and a request was sent to the smart card to sign the order. When the customer entered the password, the order was signed. To generate the digital signature, the smart card calculates a hash using the SHA-1 algorithm and encrypts the hash using card holder’s private key. Because the “encrypt” box in the “Internet Broker” window is selected, the whole message is encrypted before it’s sent over the Internet to the broker. (We’ll later deselect the “encrypt” box and see what happens).

The message is now sent over the Internet and received by the Internet broker (look at the "Internet Brokerage Server” window). The message format is checked.
and the order number is read. The broker now checks if he already received an order with the same number. (We’ll see later why this is important). If this is okay, he reads the sender’s name and the order itself. To verify the digital signature, the "Internet Brokerage Server" calculates the hash using the SHA-1 algorithm, decrypts the received digital signature using the customer’s public key, and compares the result of the decryption and the calculated hash. If these two match, the "Internet Brokerage Server" can be sure that the order was sent by the card holder and not changed or sent by someone else.

Possible security breaches

Now maximize the “United Internet Criminals Inc.” window. The window should appear in the middle of your window, between the "Internet Brokerage Server” and the “Internet Broker” windows. Enable the “Intercept” box in the “United Internet Criminals Inc.” window. The “United Internet Criminals Inc.” will now intercept all orders which the customer sends to his Internet broker, and the Internet broker will receive these orders only if they are forwarded by the “United Internet Criminals Inc.”.

Reading order data: Deselect the “encrypt” box and place a new order. Let’s say that the customer want to buy five shares of Bull. After selecting the stock, press the “Send” button in the “Internet Broker” window.

As you can see, nothing was received by the “Internet Brokerage Server” because “United Internet Criminals Inc.” intercepted the order. If you now click the “Send” button in the “United Criminals Inc.” window, the order is forwarded to the Internet broker and processed normally.

By enabling the "Encrypt" box in the "Internet Broker” window, you can prevent the data from being understood by unauthorized persons who’ve tapped the data link. The data will now be encrypted before being sent.

Placing the same order repeatedly: Now that the “United Internet Criminals Inc.” has a copy of the order, it would be possible for it to place the order multiple times; the customer then wouldn’t buy only five shares of Bull, but 10, 15, or even more.
But because of the internal order count, the "Internet Brokerage Server" realizes that the order is being replayed and doesn't process it again.

**Changing order data:** Let's place a new order for interception by the "United Internet Criminals Inc.". For example, let's now assume that the customer wants to buy 20 shares of NETSCAPE. After selecting the stock, click the "Send Order" button in the "Internet Broker" window ("encrypt" is disabled). If we now look at the "United Internet Criminals Inc." window, we see the complete order in clear text. We'll change the order by increasing the number of ordered stocks from 20 to 200 and click the "Send" button to send the order to the "Internet Brokerage Server".

The "Internet Brokerage Server" has realized that the order was changed. This is possible because the "Internet Brokerage Server" used his public key and checked the digital signature generated by the smart card with the customer's private key. If the results don't match, the "Internet Brokerage Server" knows that he didn't receive the original message and thus doesn't process the order.

### Programming the application

This section outlines the OPENCARD programming necessary to implement the Internet Broker demo program. The basic operations that must be carried out for initialization and termination, waiting for card insertion, accessing data on the card, and generating a digital signature are illustrated.

#### Initialization

Before a smart card can be accessed, OCF must be initialized. This gets the necessary CardTerminal components loaded for the installed hardware and begins the processing necessary to determine when a card has been inserted.

Also, an application can declare itself to be a listener for OPENCARD events so that it will automatically be informed when a card is inserted or removed. To do this, an application class must implement the CTLListener interface and register itself as a listener with the CardTerminalRegistry. This is also typically done during the initialization phase.

```java
// Initialize the framework
SmartCard.start();

// register the new SignatureCard as a Card Terminal Event Listener
CardTerminalRegistry.getRegistry().addCTListener(this);
```

As soon as the initialization has been carried out, the application class implementing the CTLListener interface will be informed whenever a smart card is inserted or removed from the terminal. The CardTerminalRegistry informs the application class of these events by calling the cardInserted() and cardRemoved() methods. The application can react appropriately to the event, displaying necessary messages, etc.

#### Setting up after card insertion

In the following example implementation of cardInserted(), the SmartCard object representing the inserted card is retrieved from the event. The SmartCard object is capable of providing all available card services to the application. In this case, the FileAccessCardService and the SignatureCardService are obtained. These card services are needed in order to read data from files on the card as well as to digitally sign data.
Some types of smart card access require the card holder to enter a password or to undergo some other form of card holder verification (CHV). If the application requires this type of access, it must provide a function that will allow the card service to obtain the password to the user. This is done by implementing the CHVDialog interface and passing a corresponding instantiated object to the card service using the setCHVDialog() method.

```java
public void cardInserted(CardTerminalEvent ctEvent) {
    try {
        fileService = (FileAccessCardService)
            card.getCardService(FileAccessCardService.class, true);
        signatureService = (SignatureCardService)
            card.getCardService(SignatureCardService.class, true);

        SBCHVDialog dialog = new SBCHVDialog();
        fileService.setCHVDialog(dialog);
        signatureService.setCHVDialog(dialog);
    } catch(Exception e) {
        e.printStackTrace();
    }
}
```

After a card has been inserted and the necessary card services have been obtained, data can be read from the card. This is done using the FileAccessCardService.

### Reading the file

After the user data file has been selected, standard JAVA stream functions are used to read the file.

```java
// mount file system to get access to the root directory
CardFile root = new CardFile(fileService);
// This is the file holding card holder name and e-Mail address
CardFile file = new CardFile(root, ":C009");
// Create a CardFileInputStream for file
DataInputStream dis = new DataInputStream(new CardFileInputStream(file));
// Read in the owner's name
byte[] cardHolderData = new byte[file.getLength()];
dis.read(cardHolderData);
// Explicitly close the InputStream to yield the smart card to other applications
dis.close();
```

### Generating a digital signature and terminating the Framework

Generating a digital signature is likewise a painless process.

First the private key to be used in the signing process must be selected. The private key stored on the card is represented by a PrivateKeyFile object, which is created through specification of the appropriate file ID. The signatureService.signData() method is used to pass the data to be signed to the card. Along with the data, the key file, algorithm name, and padding type to be used must be provided to the function. After successful execution, the digital signature corresponding to the data is returned to the application.

```java
// specify the key used for signing
PrivateKeyFile kf = new PrivateKeyFile (new CardFilePath("\C110"), keyNumber);
```
// Let the card generate a signature
signature = signatureService.signData(kf, JCAStandardNames.SHA1_RSA,
                                           JCAStandardNames.ZERO_PADDING, data);

Finally, when the application is finished accessing the smart card, the framework
should be terminated to allow OCF to shutdown in an orderly fashion and to avoid
wasting resources.

SmartCard.shutdown();

This concludes this brief introduction to OpenCard programming. As you can see, a
great deal can be accomplished with simple, easy-to-use functions. The code
fragments shown in this section were taken from the SignatureCard.java file
available as part of the OpenCard Framework distribution.

Conclusion

Standardized application platforms and easy-to-use frameworks are essential for the
implementation of smart card-enabled solutions and smart card-based services. The
generation of digital signatures for buying and selling stock as depicted in the
Internet Broker Demo is just one example of a host of possible smart card
applications which programmers can more easily develop with the help of the
OpenCard Framework.

In the future, reliance upon the Internet will increase. More and more consumers
will take care of their banking, shopping, and other personal business affairs over
the Internet and from a variety of locations, and the importance of network-centric
applications requiring authenticated access and secured transactions will rise as a
consequence. OCF allows smart card solution providers to develop the
necessary portable and inter-operable solutions on equipment from a variety of
different suppliers and platforms. It also enables card issuers to select from a
broader range of such solutions. OCF helps card and card terminal
manufacturers gain access to new market segments by reducing the
predominance of a single vendor. And finally, the OpenCard Framework will greatly
benefit card holders by enhancing the convenience, security, and power of their
smart cards.