OpenCard and PC/SC - Two New Industry Initiatives for Smart Cards

Abstract

With **OpenCard** and **PC/SC,** we see two new industry initiatives in the chip card world of today, which are currently developing into defacto standards and might eventually evolve to formal international standards. This document tries to give an overview of both initiatives and to explain their relationship and positioning to each other.

About the Author

Frank Seliger's current responsibility is software architect in IBM's Global Smart Card Solutions (GSCS) unit. IBM GSCS is an active member of both, the OpenCard and the PC/SC initiative. Frank Seliger is working on the architecture of the OpenCard Framework and has contributed to the PC/SC specification.

Introduction

"*The nice thing about standards is that you have so many to chose from*"

Andrew Tanenbaum

The smart card industry is currently in a transition from the gold rush atmosphere of a new technology towards a mature technology with more established products and processes. Increasingly we see industry initiatives for standardized protocols and solutions replacing the diversity and proliferation of isolated solutions. This trend makes it less of an adventure and thus more attractive to invest into solutions that use and exploit smart cards.

A strong force for standardization and interoperability arises from the belief, that almost all new smart card solutions will have several applications on one card and several possible card types carrying one kind of application. This is commonly called "**multi-application card**" **[Cart97].**

The physical level and the lower layer protocol-levels have been governed by the formal **ISO 7816** standard for several years now **[ISO-97].** This standard leaves enough room for variation and interpretation that components from different manufacturers are often not compatible with one another although they conform to the ISO 7816 standard. The higher layers of programming, which an application programmer using a smart card is preferably dealing with, are barely regulated by ISO 7816.

Besides ISO 7816 there are several specifications that emerged from specific application domains. Examples are **EN 726** from the telecommunication domain **[EN726]**, **EMV'96 3.0** from the finance domain **[EMV-96]**, **EU/G7 WG7** from the health care domain **[G7-97]**, just to mention a few.

OpenCard and **PC/SC**, which we will look at more closely in the following, are both not business domain specific. The two initiatives have a different scope and focus.

OpenCard as a specification is concerned with an object oriented framework located in the computer to which the smart card is attached. The reference implementation has been developed using the **Java** programming language and execution system and lends itself to be used for application development in Java. For regulation of the smart card itself, OpenCard relies on ISO 7816. Therefore, OpenCard can be used together with any smart card.

PC/SC as a specification is focussing on the Personal Computer and the card terminal

attached to it. The central component of specification is the **Resource Manager**, which should be integrated into the computer's operating system. The implementations available today are offered for use on Personal Computers running a 32 bit Microsoft Windows operating system. For regulation of the smart card itself, PC/SC relies on ISO 7816. Therefore, PC/SC can be used together with any smart card.

Technical Challenges of Providing a Multi-Application Card

Smart card applications essentially comprise two parts: the application part resident on the card, henceforth called *card-resident application* and the application part running in the computer henceforth called the *card-using application*. In the case of a freely programmable terminal this device itself can be considered "the computer" here. The card-resident application comprises the data elements kept on the card in the context of the application, such as a purse balance, and possibly application-specific functionality in the card, such as the secure increasing or decreasing of a purse balance. The card-using application comprises the program code in the attached computer (or terminal) that interacts with the card-resident data and functions.

Parties involved in deploying a smart card application

Smart card applications are deployed by a cooperation of a **Card Issuer** and an **Application/Service Provider** (see Figure 1: Parties involved in developing the software of a smart card solution). In the past, these two roles have commonly been owned by the same organization. With multi-application cards, this has to change. An application developer and service provider may want to deploy his applications on smart cards issued by different issuers. Issuers may want to deploy smart card applications from various application developers and service providers on their cards.

Furthermore, issuers may want the freedom to choose different smart card types with different **Card Operating Systems** to deploy the same application. For instance, for a simple multi-application card combining airfare collection and a loyalty application, the issuer may want to choose an inexpensive card. For the executive card, which in addition combines these two applications with credit/debit and electronic cash applications, he would use a more expensive card with tighter security mechanisms.

Finally, access to these applications should be globally accessible and therefore should work with the equipment and hardware drivers provided by any **Card Terminal Provider**.

The technical challenge is to let these different players vary without redoing the entire solution with every change of a single component. The different parts contributed from the different players must be independent and interoperable. This is the primary goal of the two industry initiatives described in the following.

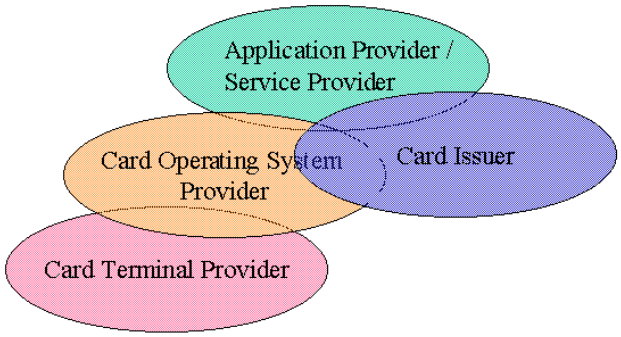

Figure 1: Parties involved in developing the software of a smart card solution

OpenCard

The OpenCard initiative was started in early 1997 by the network computer industry. Most of the future network computers will have an integrated smart card terminal. Therefore, the middleware to drive smart cards was specified as part of the **Network Computer Reference Profile** **[NCRP97]**. The companies driving OpenCard as part of the Network Computer Reference Profile were IBM, Netscape, NCI and Sun.

Later more technology providers as well as smart card users joined. In October 1998 the OpenCard Consortium included 3-G International Inc., Bull Personal Transaction Systems, Dallas Semiconductor Corporation, First Access, Gemplus, IBM, Intellect, Network Computer Inc., Newcom Technologies, Schlumberger Smart Cards & Terminals, SCM Microsystems, Siemens Microelectronics, Sun Microsystems, UbiQ and Visa International.

Figure 2: The members of the OpenCard Consortium

The technical focus of OpenCard lies at providing an object oriented framework (**OpenCard Framework**, or **OCF** for short) to the application developer and at decoupling the different parties involved in a smart card solution (see Figure 1). The reference implementation of OCF has been developed as a set of Java packages and classes **[OCF97]**.

The service provider or application programmer is writing his services to the interface provided by the OpenCard Framework itself and of the CardServices, which are plugged into the framework. This ensures that differences or changes in the card operating system, in the card terminal or in the application management scheme used by the card issuer do not impact the application code. **Figure 3** shows the highlevel architecture of OCF. Together with each component, the provider of it is indicated (horizontal layers).

The application developer provides the application program (shown as the highest layer).

The card issuer is responsible for the application management layer (second from top). This layer manages the execution and coexistence of the different application functions and application data on the same card. Examples of application management schemes are found in ISO 7816-4,5 and in EMV'96 3.0.

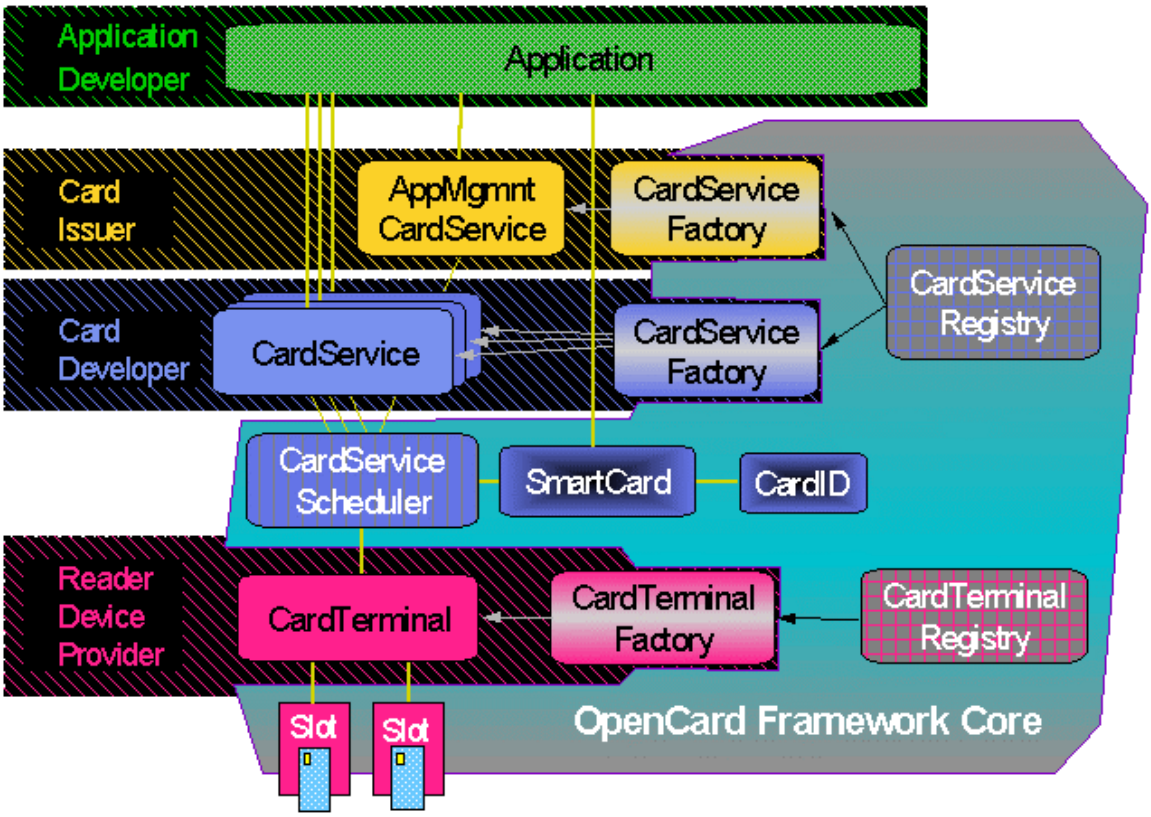

Figure 3: Components of the OpenCard Framework and their providers

The application program will most likely use application data on the card. In addition, it might use card-resident application-specific services. If the card is a JavaCard, these services and the data they operate on are implemented as a card-resident applet. For traditional smart cards, the application-specific function on the card is usually provided by the developer of the card operating system. The encapsulation of these dependencies in OCF is called **`CardService`** (third layer from top). A typical `CardService` would be a `PurseCardService` providing function to increase of decrease the purse balance by a specified amount. The application management, which we have seen in the previous layer, is as well offered as a `CardService`.

The provider of the card reader device must also supply a **`CardTerminal`** object to drive the actual hardware (lowest layer).

The OpenCard Framework core is covering the right side of the picture. It contains objects like `CardServiceScheduler`, `SmartCard`, `CardID` and two registries.

In the following, we will look at OCF and its classes in more detail. Before we do so we briefly revisit a basic principle of object oriented software development: An **object** is an **instance** of a **class**. A class can be considered the blueprint for all objects of the same kind.

The **`class SmartCard`** is the central abstraction. A `SmartCard` has a **`CardID`** that contains the information uniquely identifying the card type. This information is gained after the first communication with the card has been established in the *Answer To Reset (ATR).* The `SmartCard` is the primary object used by the application program. For application specific functions, the application turns to the respective **`CardService`** objects.

A CardService has all information about the card operating system that it supports. CardServices can use function from other CardServices. For example, a PurseCardService, that needs to access the files on the card, should access these through the FileSystemCardService.

All knowledge about a family of card services is encapsulated in a **CardServiceFactory**. The use of such factories is a standard object oriented design pattern. The CardServiceFactory with a set of CardServices is usually provided by the same developer.

When a CardService with a particular interface is requested from the OpenCard Framework, the **CardServiceRegistry** calls every CardServiceFactory registered for the CardID of the SmartCard object until an appropriate CardService has been created. The new CardService object is now connected to the SmartCard object with which it will be used.

While we can have many CardService and CardServiceFactory objects in a system, there is always just a single CardServiceRegistry.

If cryptographic functions are needed to access the card, they are called by the CardService. Again, the CardService knows best about the security mechanisms and protocols needed for the specific card operating system.

A CardService communicates with the CardTerminal through the **CardServiceScheduler**. As the SmartCard and the CardID classes, the CardServiceScheduler class is provided by the OpenCard Framework core.

The CardServiceScheduler takes care of synchronization in situations with concurrent access from different applications on the same card. Consequently, we have for every smart card exactly one CardServiceScheduler object.

The main abstraction in the reader device layer is the **CardTerminal** class. For every hardware device, the associated CardTerminal must be provided. A CardTerminal has one or more objects of type **Slot**. This allows exact modeling of the situation of card terminals with more than one slot for card insertion, for example to accept a patient data card together with the authorization card of the doctor. A CardTerminal can also have a keyboard and a display (not shown in Figure 3).

As we have previously seen in the card service layer, a CardService is created from a CardServiceFactory and all CardService and CardServiceFactory objects are controlled by the CardServiceRegistry. Now we find the same mechanisms in the reader device layer: A CardTerminal object is created from a **CardTerminalFactory,** and to keep track of all known CardTerminal types is the responsibility of the **CardTerminalRegistry**.

The application management layer contains the application management CardService and the associated CardServiceFactory. An application management CardService has all information on the application management scheme used. This scheme is usually controlled by the card issuer. Industry initiatives to agree on the application management scheme are under way (for example EMV'96). Until the world has agreed on a single scheme, exchangeable application management CardService objects are protecting the application writers from differences and changes in that layer.

OCF offers a separate component for the file system of a smart card, as specified in ISO 7816-4. This function is not part of the core framework `opencard.core`. Rather it is contained in the optional packages `opencard.opt.` In the `opencard.opt.iso.fs` package, we find a **`CardFile`** with attributes like **`TRANSPARENT`**, **`LINEAR_FIXED`**, **`LINEAR_VARIABLE`** and **`CYCLIC_FIXED`** and a **`CardFilePath`**, **`CardFileInputStream`**, **`CardFileOutputStream`**, **`CardFileReader`** and more.

The Java reference implementation of the OpenCard Framework is publicly available on http://www.opencard.org at no cost. Due to the US export restrictions on cryptographic software, the distribution is split into a part without cryptographic capability and into a part that contains cryptographic software which is underlying the export restrictions.

PC/SC

PC/SC, with full name "***Interoperability Specification for ICCs and Personal Computer Systems 1.0***" **[PCSC97]**, covers the use of smart cards with Personal Computers. In eight parts, the specification is spanning the entire range from the physical characteristics required from smart cards and readers up to the layer of application programming. The specification is based on and is compatible with ISO 7816. The focus is on the interoperability of smart card (called **`Integrated Circuit Card`**, or **`ICC`**) and card terminal (called **`Interface Device`**, or **`IFD`**) and on the cooperation between the card terminal and the PC operating system.

The PC/SC Workgroup was formed in May 1996. The companies that drove the PC/SC specification 1.0 are: CP8 Transac (Bull), Gemplus, Hewlett-Packard Company, IBM, Microsoft Corporation, Schlumberger SA, Siemens Nixdorf Informationssysteme AG, Sun Microsystems, Toshiba and Verifone. Most of these companies also have announced PC/SC compliant products, particularly smart card readers of all kinds.

The eight specification parts of PC/SC 1.0 are shown in the Figure 4 and explained below.

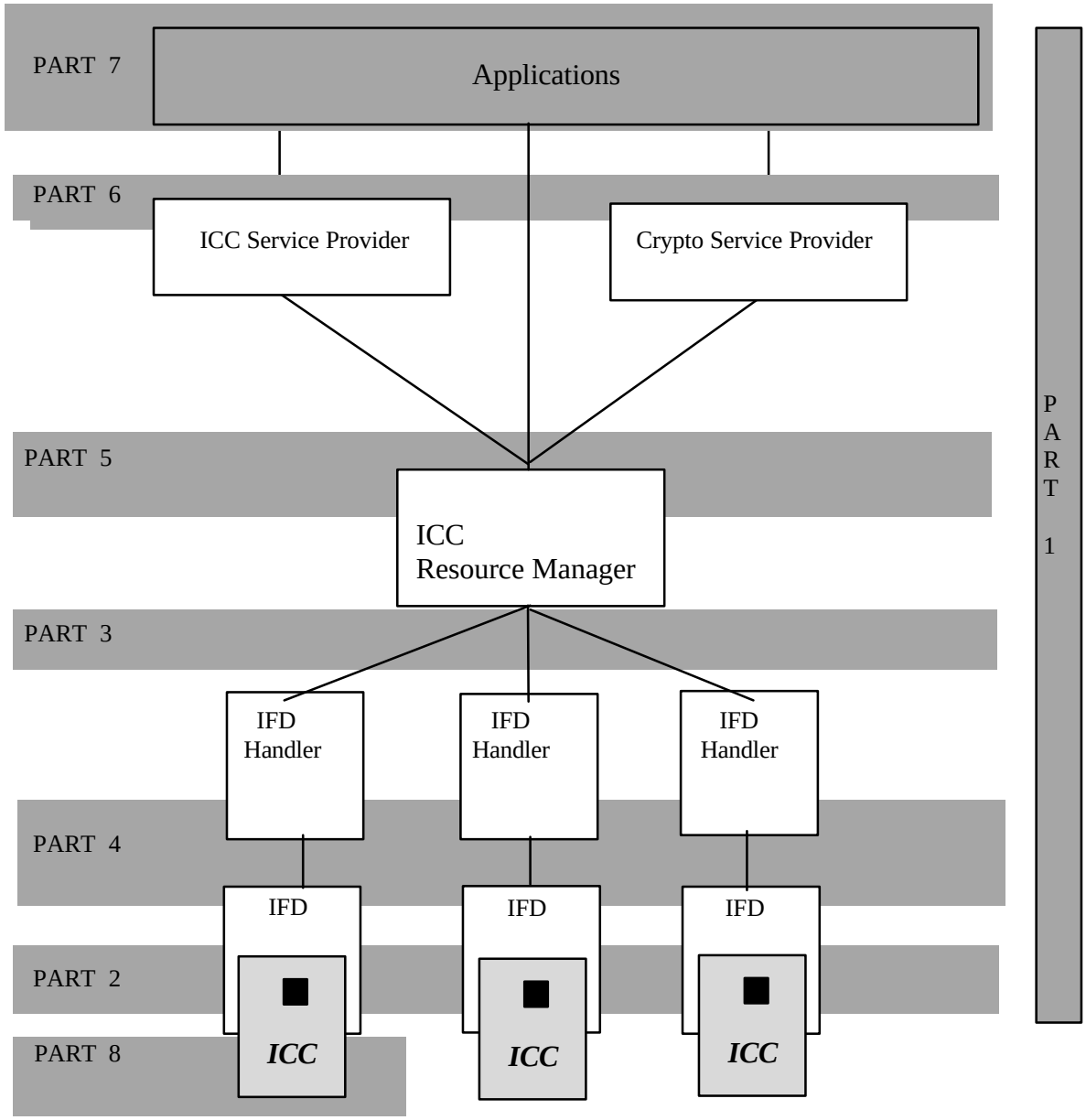

Figure 4: Architecture layers and specification parts of PC/SC 1.0

1. Introduction and Architecture Overview

2. Interface Requirements for Compatible IC Cards and Interface Devices

   Part 2 specifies physical characteristics of the card and the reader hardware, as for example voltages. The lower transport protocol levels including the protocol error handling rules are also specified here. Further, we find here the specification for the expected Answer To Reset (ATR).

3. Requirements for PC-Connected Interface Devices

   Part 3 specifies the characteristics of the Interface Device Subsystem. To let the higher layers of PC/SC communicate with the **Interface Device (IFD)** in a device independent way, an **IFD Handler** is adapting the IFD to a common

programming interface.

The IFD must provide basic function like sending commands to the card or checking for card insertion. Optionally it can have various other capabilities, such as powering up or down the card inserted, or user authentication with a PIN pad, keyboard, fingerprint scanner, retina scanner etc.

4. IFD Design Considerations and Reference Design Information

Part 4 gives help and guidelines for the design of the Interface Device Subsystem's inner protocols.

5. ICC Resource Manager Definition

The **ICC Resource Manager** is the central component of a PC/SC system. It controls all IFD's, as well as the off-card resources providing the card's application-programming interface (called ICC Service Provider) and the cryptographic function (the Crypto Service Provider). The ICC Resource Manager is a privileged component. It should offer the same degree of protection and security as the base operating system. In future versions of the Microsoft Windows operating system, the ICC Resource Manager will be an integral part of the operating system.

6. ICC Service Provider Interface Definition

Part 6 specifies the ICC Service Provider and the Crypto Service Provider interfaces. These two interfaces are the primary interfaces the application developer will use.

The **ICC Service Provider (ICCSP)** must offer a mandatory **class SCARD** that maintains the context of the communication with the smart card. Optionally the ICCSP may offer classes for access of the files on the smart card and for the authorization functions needed to access the card.

The **Crypto Service Provider** is optional. If cryptographic function is needed for the access of the card, that function is localized in the Crypto Service Provider. Such function often falls under export restrictions for cryptography components. Keeping it localized allows for better control. For the Crypto Service Provider the interfaces are specified for key generation, key management, key import/export, digital signature, hashing and bulk encryption services.

The ICC Service Providers need to be registered together with the card and interfaces they support. This enables the system to identify and fetch the appropriate ICC SPs for a particular card. For the identification of the card, the ATR or parts of it are used.

7. Application Domain/Developer Design Considerations

Part 7 contains advice for the application developer on how to use the interface provided by the ICC Resource Manager to obtain information on the card readers installed, to wait for insertion of a particular card and other common tasks.

8. Recommendation for Implementation of Security and Privacy ICC Devices

Part 8 contains advice on how to handle identification, authentication, and secure storage and how to achieve information integrity, traceability and confidentiality in a smart card solution. In the section on Cryptographic Services, it also offers an

excellent overview of the current cryptographic methods.

Microsoft has provided implementations of the PC/SC 1.0 specifications on the Windows 95 and Windows NT 4.0 platforms and has released this implementation to the public Web. For Windows NT 5.0 the resource manager will be part of the operating system, for Windows 98 it is shipped as part of the installation package.

The Relation of OpenCard Framework and PC/SC

OpenCard Framework and PC/SC are both providing for the access to smart cards from computers of different kind. We can expect that they have concepts and mechanisms in common. Comparing the individual components of both, we find that this is quite true.

Commonality in the Architectures of OCF and PC/SC

The OpenCard and the PC/SC initiatives both have the goal of supporting the interoperability of the different elements of a smart card solution, of card, card reader and application software. How do the architectures of the OpenCard Framework and of PC/SC address this goal? Figure 5 is showing the correspondence of the architecture elements of OCF and PC/SC.

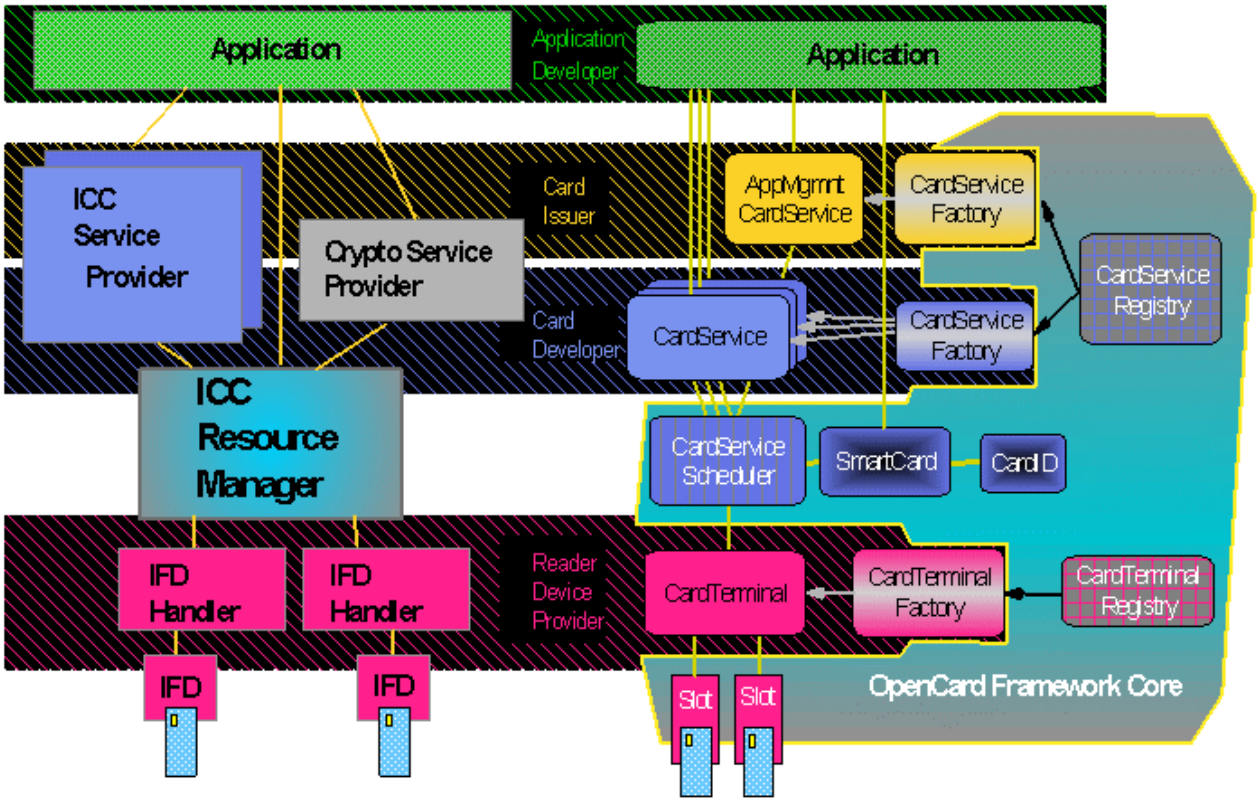

Figure 5: Comparison of the Architecture components of OCF and PC/SC

The application program uses mainly application specific services and application management services. In PC/SC the ICC Service Providers offer these services, in OCF the CardServices provide them. The CardTerminal in OCF finds its correspondence in the Interface Device Subsystem in PC/SC. Therefore, on comparing the major components OCF and PC/SC have a lot in common.

Differences in the Architecture of OCF and PC/SC

Architectural differences appear when we zoom closer into the components. While OpenCard does not make any further specifications of the internal workings of the CardTerminal, PC/SC provides very detailed guidance for the Interface Device Subsystem. PC/SC structures the Interface Device Subsystem into the IFD and the IFD Handler (see Figure 5) and gives very detailed specifications for both. The absence of detailed specifications for the inner structure of an OpenCard CardTerminal has a desirable consequence for the interoperability between OCF and PC/SC. It was possible to create an OpenCard class PCSC10CardTerminal that allows the use of any PC/SC compliant IFD Subsystem as OpenCard CardTerminal.

Thus, OCF and PC/SC do hardly overlap in the area of the CardTerminal or IFD. The rich specification of PC/SC rather complements OCF.

More architectural differences become apparent, when we examine the layers for the card itself more closely.

Table 1 lists the corresponding classes of OCF and PC/SC and their responsibilities. The first difference between OCF's and PC/SC's handling of the smart card layers is a couple of unmatched abstractions in OCF, CardID and SmartCard.

OCF class	PC/SC class	Responsibility of the class
CardID	(use basic String)	ATR and identification of card type
SmartCard	(no direct correspondence)	An application's view of a smart card
(no direct correspondence)	SCARD	Smart card communication context
CardFile	FILEACCESS	An application's view of a smart card file
CardFileInfo	(no direct correspondence)	Access permissions for a smart card file
(no direct correspondence)	CARDAUTH	Authentication card-to-app. + app.-to-card
CHVDialog	CHVVERIFICATION	User Dialog for Card Holder Verification
CardFilePath	(no direct correspondence)	Path of a smart card file
Credential	CRYPTKEY	Cryptographic Key
(no direct correspondence)	CRYPTHASH	Hashing (message digest) for crypto
CredentialStore	(no direct correspondence)	Credentials (keys) for one application and OS
CredentialBag	(no direct correspondence)	Bag of all credentials (keys) for one application on all supported OS
CardService	(no direct correspondence)	Application (domain) specific service
ApplicationManage-	(no direct correspondence)	Application selection and data management

`mentCardService`		

Table 1: Classes for abstraction of the smart card in OCF and PC/SC

The next difference is more significant. OCF and PC/SC took a completely different cut at the organization of cryptographic algorithms and the associated credentials. This is hardly surprising. This area would best be addressed by a separate cryptographic framework. Unfortunately, the choice of cryptographic standards and frameworks is all but clear. PC/SC specified another generic crypto interface with the interface of the `Crypto Service Provider`. OCF treats the interface of the crypto functions needed as an internal decision of the CardAgent. If the industry reaches some more convergence in that area, we can expect OCF to use a standard crypto interface. We can speculate that PC/SC will try to adapt to such a standard interface, if this is possible in a backwards compatible way.

The most significant difference in the smart card layers of OCF and PC/SC is following structural division found in OCF but not specified in PC/SC. OCF enforces a clear separation between application specific services (OpenCard `CardService`) and application management function (OpenCard `ApplicationManagementCardService`).

Differences in the Scope of OCF and PC/SC:

As we have already discussed, PC/SC concentrates on the PC platform. PC/SC concentrates more on the reader device than OCF. OCF has more emphasis on specifying a separation of application-specific card services and application management.

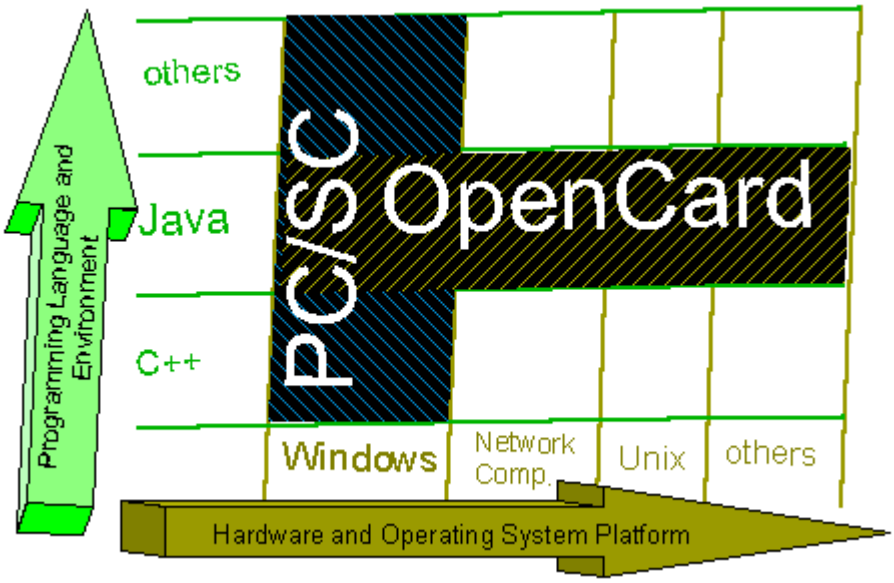

Figure 6: PC/SC and OpenCard have a different scope and overlap only where Java is used on Windows

Besides these obvious differences, there are a few more subtle distinctions. OCF has been architected and designed by Java programmers for Java programmers. It is centered in the world of Java and the net. Java programmers will feel at home with OCF's style, for example with the smart card file I/O that is closely modeled after the standard I/O of native Java.

Is there a conflict on the PC between OCF and PC/SC?

Since there will be applications on the PC using PC/SC and others using OCF, will we be in the next problem of incompatibility, just at a higher level? The members of the PS/SC initiative and the members of the OpenCard initiative work together to avoid this problem. OpenCard supports existing PC/SC interface devices as OpenCard CardTerminals. OpenCard's clear separation of card operating system, application-specific card services and application management can be added to PC/SC as a backwards compatible extension. This is addressed by the technical workgroups of OpenCard and PC/SC.

What can we predict for platforms other than the PC and NC?

OpenCard will very likely be available not just on PC's and Network Computers, but also on other platforms like POS terminals, PDA's, smart phones and set top boxes. We can expect to see OCF implementations that adapt to the limited Java environments of smaller devices. These versions will exploit and help exploiting the portability of Java code.

For PC/SC we can expect an implementation on the future versions of Window CE. For PC/SC it is an open question if we will see it on additional non-Windows platforms. A subset of PC/SC function has been implemented by IBM for the Windows 3.11 and OS/2 operating systems. For Linux there exists an Alpha release of a Smart Card Resource Manager offered by the MUSCLE (Movement for the Use of Smart Cards in a Linux Environment) **[Musc98]**. Microsoft stated that it is not interested to provide PC/SC on other platforms than the new versions of Microsoft Windows, but that it welcomes other companies, should they plan to provide this. The implementation of the resource manager function for another operating system does not seem a small task.

Relation to **Java ™ Card** and **Windows for SmartCard ™**

Sometimes the names OpenCard and Java Card seem to be a source of confusion. We attribute the difficulties to the fact, that the OpenCard reference implementation is using Java. Nevertheless, the difference between OpenCard and Java Card is simple and fundamental: Java Card **[JCrd97]** is Java in the card itself, while OpenCard and PC/SC are driving the card from the outside.

A Java Card will typically be used in conjunction with OpenCard and/or PC/SC. Particulary OpenCard as a Java framework can offer a seamless Java environment together with JavaCards. In the same way the Windows for SmartCard cards, announced by Microsoft in October 98, can be used in conjunction with OpenCard and/or PC/SC.

References

[Cart97] Several papers on multi-application smart cards and trends can be found in the Conference Proceedings of the "Cartes 97", Oct 15-17 in Paris.

[EMV-96] EMV'96 3.0 Integrated circuit {card, card terminal, card application} specification for payment systems

[EN726] EN 726 Telecommunications integrated circuit(s) cards and terminals

[G7-97] EU/G7 WG7-3 Interoperability of Healthcard Systems

[ISO-97] ISO 7816 Identification cards – Integrated circuit(s) cards with contacts

[JCrd97] http://www.javasoft.com/products/javacard (Java ™ Card is a trademark of Sun.)

[JSec97] http://java.sun.com/products/jdk/1.1/docs/guide/security

[Musc98] http://www.linuxnet.com/smartcard/index.html

[NCRP97] http://www.opengroup.org/pubs/catalog/x975.htm

[OCF-97] http://www.opencard.org

[PCSC97] http://www.smartcardsys.com

[SCWin98] http://www.microsoft.com/windowsce/smartcard/default.asp (Smart Card for Windows™ is a trademark of Microsoft.)