RZ 2999 (#93045) 03/09/98
Computer Science/Mathematics 4 pages

Research Report

The OpenCard Framework and PC/SC

Matthias Kaiserswerth

IBM
Route 100
Somers, NY 10589
USA

E-mail: kai@zurich.ibm.com

IBM Research Division
Almaden • Austin • Beijing • Haifa • T.J. Watson • Tokyo • Zurich

The OpenCard Framework and PC/SC

Matthias Kaiserswerth

IBM, Route 100, Somers, NY 10589, USA
kai@zurich.ibm.com

Abstract

In this brief paper we would like to position the **OpenCard Framework** (OCF) visa-vis a similar effort, namely the **Interoperability Specification for ICCs and Personal Computer Systems** (PC/SC), both of which are designed to offer a standard way to integrate smart cards into computer systems. The fact that two consortia exist which roughly considered work on the same problem has led to some confusion within the industry, in the trade press, and among customers for smart card solutions. After an introduction into the genesis of these two efforts, we will attempt to compare the technical architectures, look at the value proposition of each, and explain why they in fact are largely complementary and why there is often a clear choice for one or the other or even both. The discussion presented here is based on the Version 1 of both PC/SC and OpenCard. To ease the comparison, some liberty was take to rename the components of both architectures, so they correspond better to commonly used terms when speaking about smart cards.

History

PC/SC was formed in mid 1996 to enable PC applications to exploit smart cards for security and e-commerce related applications[1]. The founding members of PC/SC came, except for Microsoft, all from the smart card industry who had realized that a standard way was needed to integrate smart cards into PC applications and to grow their over all market. Until then applications requiring smart card support were always tightly tied to a chosen smart card reader and smart card operating system. Interoperability, where one reader could be replaced with another reader or one smart card could be replaced with that of another manufacturer did not exist. The smart card industry realized that for smart cards to become widely accepted as mobile, secure tamper resistant tokens within PC applications it was necessary to define a clear standard similar to the way diskettes are now used in PCs. In PC/SC much emphasis was placed in the interoperability of smart cards and card readers and on the integration of card terminals into the Windows operating system. PC/SC takes into account the relevant international standards that already existed before, such as ISO 7816, CEN prEN 726, and EMV[2].

The OpenCard consortium[3] was started in March 1997 by a different group of companies. The founders were all computer (specifically network computer) manufacturers who saw that they would greatly benefit from a common framework[4] to support smart cards in these machines to securely authenticate users and personalize the desktop of these otherwise anonymous devices. The initial smart card application they envisioned was similar to what is done with GSM cell phones, which are activated and personalized through inserting a smart card, the so-called subscriber identification module (SIM). Being the last one to the party, OpenCard decided to pick and use whatever was already available in previous work (in particular PC/SC) and focus its efforts mainly on two areas which so far had not been addressed in PC/SC. First, through the use of Java as the specification and implementation language, OpenCard became independent of the underlying operating system. Many systems that use smart cards do not run Windows NT/95 presently and will, for various reasons, e.g., resource requirements, likely not do so in the future (e.g., a POS terminal or a public telephone). Admittedly very few of these systems currently offer Java support, however with the advent of embedded and personal Java, the likelihood of Java support on these systems increases, because of Java's **Write Once Run Anywhere** promise, which holds much value for the manufacturers of these devices. The second area that OpenCard addresses is to provide a framework in which it becomes possible to define a standard set of higher level APIs for smart card aware applications (e.g., user authentication, desktop selection, payment, cryptographic services), query which APIs are supported and offer a platform where it is possible to interchange one manufacturers smart card with that of another for as long as they both implement the same functionality.

Architecture

Architecturally PC/SC and OpenCard have similar structures, the main differences arise because of OpenCard's object-oriented design, which led to more discrete objects than can be identified in PC/SC.

---

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

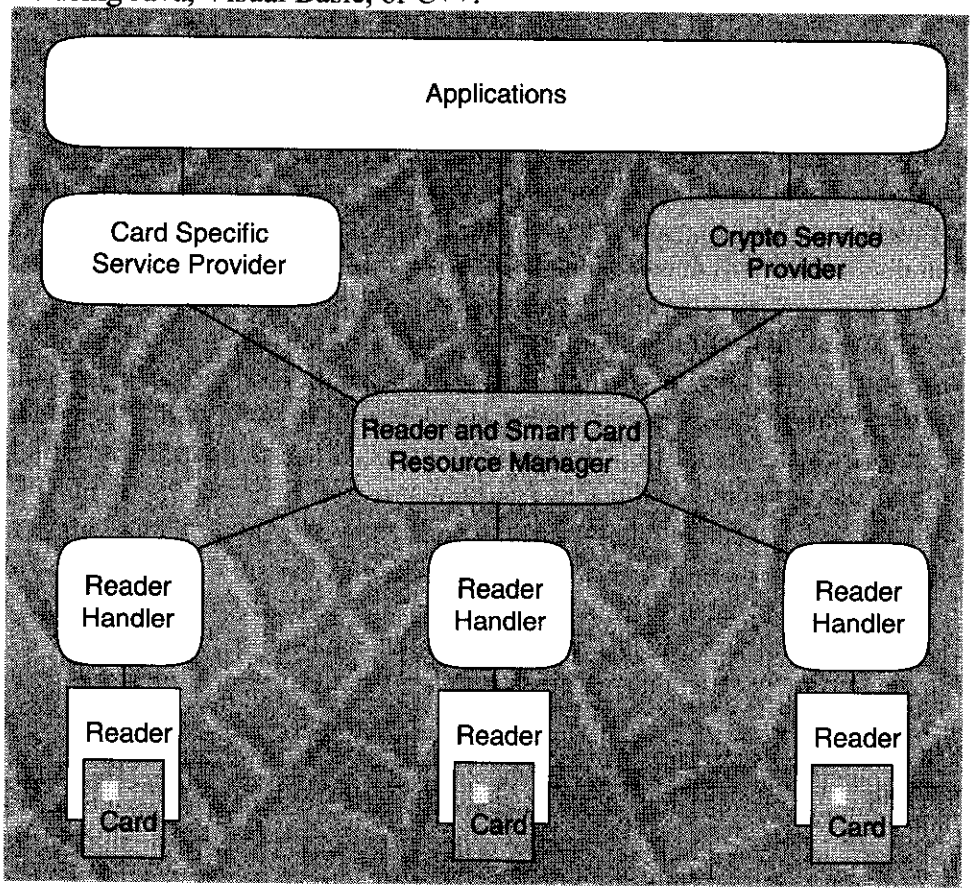
[2] Europay, MasterCard, Visa, EMV Integrated Circuit Card Specifications, http://www.visa.com/cgi-bin/vee/nt/chip/history.html?2+0#a

[3] http://www.opencard.org

[4] OpenCard Framework (OCF)

PC/SC

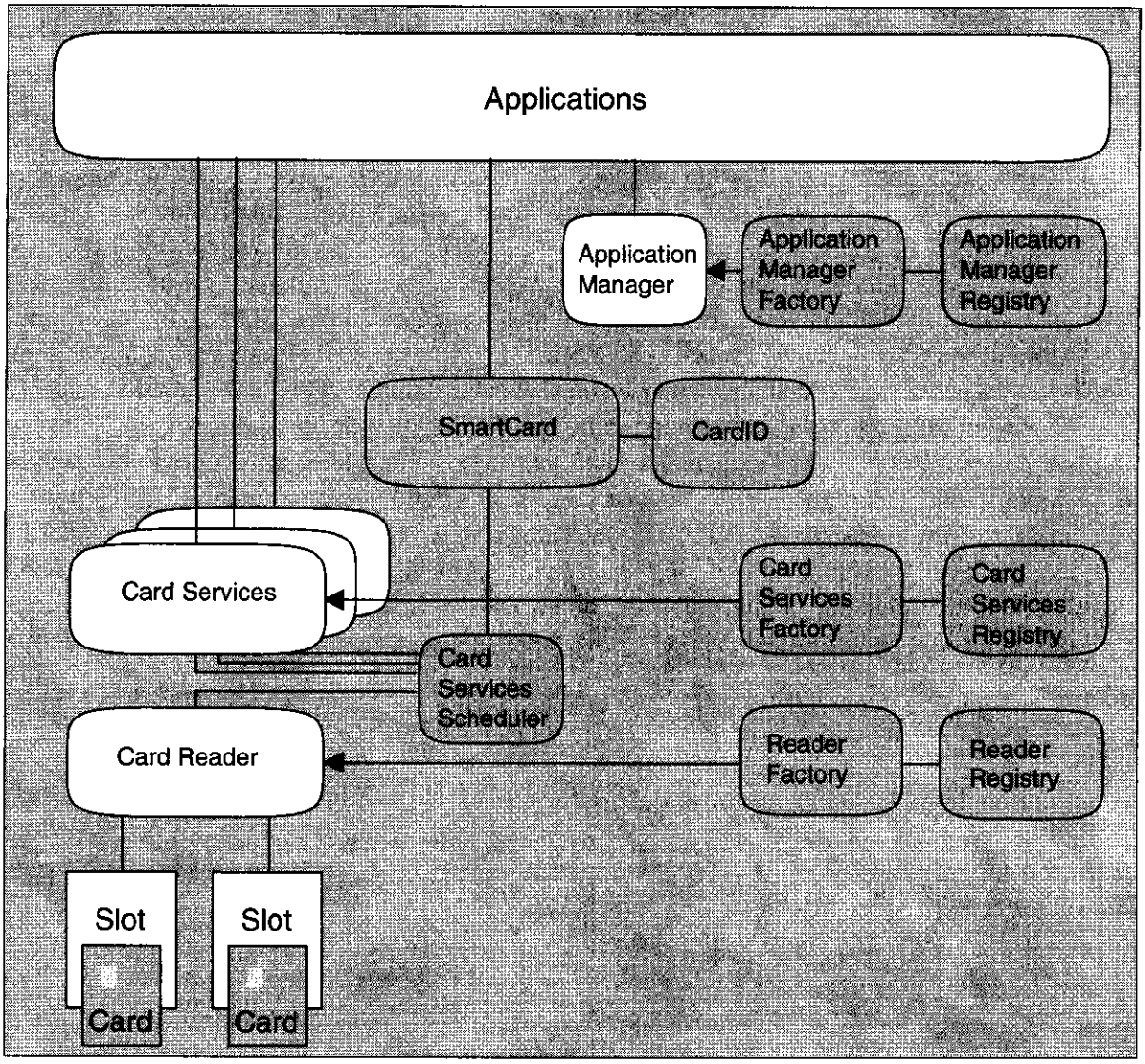
The central component of the PC/SC architecture is the *Resource Manager* which mediates all accesses to the smart card reader and to the card itself. The resource manager is a privileged component and will be integrated into future versions of the base operating system. A *Reader Handler* is a device driver for a specific reader, which maps the functionality of that reader to the native services provided by Windows and the smart card stack. The *Service Providers* expose the services of a smart card to an application through interfaces. A smart card interface consists of a predefined set of services, the protocols necessary to invoke the services, and any assumptions regarding the context of the services. A smart card can claim to support an interface through association with the interface's globally unique identifier (GUID). This binding between a card and an interface is done at the time the card is first introduced to the system, typically when the service provider is installed. Microsoft will ship several lower-level service providers to perform generic operations such as card location and command/reply APDU (Application Protocol Data Unit) maintenance. The Microsoft-supplied service providers are implemented as COM interface objects to enable software developers and card providers to develop higher-level service providers and applications using Java, Visual Basic, or C++.



OpenCard Framework

The OpenCard Framework architecture is similar in its structure, except it has split PC/SC's central resource manager into a number of registry and factory objects with responsibility for the instantiation of reader drivers, card services drivers, and application managers. The reader drivers map reader specific functions into standard APIs offered in OpenCard. Unlike PC/SC which can only handle readers with a single slot, OpenCard supports readers with multiple slots to accept a smart card. These readers are often used in financial or health care

applications, where a master smart card needs to be present in the same reader to unlock the use of the second card, which may belong to a customer or patient.
The *Card Services Scheduler* is responsible to serialize access to a given smart card and maintain state for this card. *SmartCard* and *CardID* are objects used by the application programmer to identify a given card and associate it with a slot in a reader. Common application specific APIs (e.g., PKCS #11, user authentication, purse, ...) are provided through individual *Card Services* objects. Because smart cards from different manufacturers will have different operating systems, the implementations of these classes are unique for a given card operating system. For a given smart card aware application, however, they will all offer the same API. Finally, OCF introduces a smart card application management layer, which cannot be found in PC/SC. Currently a number of different schemes exist on how to manage multiple applications on a single smart card. E.g., EMV specifies a different application selection scheme than what is used in GSM SIM cards. An *Application Manager* has all the information that is required to identify and select different applications on the same smart card.



Card Services and *Application Manager* make it possible that a single smart card aware application, e.g., user authentication can interoperate with smart cards from different manufacturers following different multi-application management schemes.

Value Propositions

PC/SC. By virtue of being supported by Microsoft, PC/SC will quickly become pervasive on all desktop systems running Windows and be fully supported by Microsoft's software development tools. Already there are downloadable device driver and software development kits available at Microsoft's web site. Many smart card reader manufacturers have already developed PC/SC compliant device drivers for their readers and smart card vendors have started to develop specific service providers for their cards. Further work will be required in PC/SC to raise the level of APIs, such that smart card aware applications can truly be programmed to be independent of the underlying smart card operating system. Unlike OpenCard, which is agnostic about application selection on multi-application smart cards, PC/SC requires a common application selection scheme and specifies concrete criteria for a smart card to be PC/SC compliant, which at present rules out many cards that have already been distributed. While PC./SC claims that it is specified in an operating system independent manner, PC/SC currently only exists for Windows 95/NT/98. The proof still need to be made that application code and PC/SC drivers can be reused on non-Windows platforms or even Windows CE.

OCF. OpenCard strives to be independent of the underlying operating system and because it is implemented in Java, all the OCF components written in Java become available on any Java enabled platform, be it a client (NC or PC), a server, or an embedded system. Compared to PC/SC, fewer OCF specific reader drivers exist. On Windows systems, however, OCF offers Java wrappers so it can transparently use PC/SC reader handlers and draw on the wide support of device drivers there. Regrettably, because of PC/SC's monolithic resource manager, it is at present not possible to use and enhance on PC/SC's card service provider. By virtue of following a strict object-oriented design and being implemented in Java, OCF is ideal for use in Java applications, applets, or servlets. Calling Java functions from other programming languages is more difficult and requires additional layers of software.

When to Use What?

The question of whether one should base a development on PC/SC or OCF will largely depend on the foreseen operating environment for the smart card aware application and depend on the question whether the multi-application management capability of OCF is deemed important or not. OCF provides transparency for the application programmer with regard to smart card operating systems, card terminals, and card issuers. PC/SC primarily addresses transparency with regard to card terminals and to a limited extent to card operating systems, but neglects the role of the card issuer. As multi-application cards are deployed, provision of functions to support installation, removal, enumeration, selection of applications on the card and of functions to perform name resolution for data files on the card becomes important. If, however, the application will exclusively be targeted for Windows, only will need to support a particular card issuer and will not be implemented in Java PC/SC will play an important role.

If, however, the application will be implemented in Java, OCF is the preferred choice because of its higher-level APIs and Java's write-once-run-anywhere promise. OCF solves the support of different multi-application cards through its card management layer. The programmer can develop a smart card aware application based on an abstract file system model, without concern about where on the card the issuer will install the smart card application. OCF architects the basic mechanisms by which the framework adapts to a particular card operating system, card terminal, and card issuer. These mechanisms are flexible enough to allow the network download and installation of missing components.