Open Smart Card Infrastructure for Europe

v2

Volume 5: **Multi-applications**

Part 3: **Basic Technologies for Multi-application Cards and Systems**

Authors: **eESC TB7 Multi-application Smart Cards**

eESC TB7 Contacts:

Chairman: Lorenzo Gaston gaston@montrouge.tt.slb.com

Secretariat : MTA brains@mta.fr

Version	Date	Contributor(s)	Main Alteration vs. previous revision
WDv1.0	7 Dec 01	L Gaston, TB7 Chairman	Creation of the document. Partially based on ISOP D1
WDv2.0	12 Nov 02	L.Gaston M. Faher (SchlumbergerSema) A. Rhélimi (SchlumbergerSema) JC. Pellicer (Trusted Logic) C. Wrathall (eURI) Tim France-Massey (MAOSCO) A. Hovsto (ERGO)	Integration of MULTOS requirements Completion of STIP Provisioning and SynchML Introduction to Guillou-Quiscater 2 Authentication Protocol Introduction to eURI
v1.0	Mar 03	L Gaston, TB7 Chairman	English Review

TABLE OF CONTENTS

1 INTRODUCTION : GENERAL OBJECTIVES

WP4 investigate Multi-Application (MA) Architectures from different business sectors in light of emerging new standards and new industrial products. The final business target is to stimulate the offer of new MA systems based on our conclusions and recommendations.

1. The first technical objective of WP4 is to reach conclusions concerning a possible « universal » MA flexible architecture, scalable down sufficiently for it to be endorsed by different card industry sectors. This Architecture enables the ***network mobility of the software***. This modular MA architecture shall allow the distribution of new value-added services for the citizen irrespective of the specific platform used (Fixed/Mobile, Private/Public). The Initial Framework Set of Requirements for this Architecture are presented MAS Spe
2. Software network mobility enables data to be delivered together with the software that knows how to manipulate or to display it. For interoperability purposes it is convenient that this software can be downloaded in terminals and cards embedded in a standard format file able to transport heterogeneous types of data. This ***"Ready-for-Execution" format file*** needs to be adapted to the reduced memory capabilities of small processing devices and , in particular; of the smart card.
3. ***Platform independence*** is a challenge, because many different types of computers and devices are usually connected to the same network. The s***ecurity model*** is also a challenge to the internal secure architecture of the card because networks represent a convenient way to transmit malicious or corrupt code. Therefore, Open Smart Card Platforms have their specific security concerns. More on this is presented in #6.
4. A second hybrid technical/marketing objective is to investigate technical solutions allowing a ***better control by the cardholder of the contents of its card***. The marketing concept under study is the perception of the card as a personal object which enables access to personalized services. This concept involves the ability of the proposed system to allow data downloading once the card has been issued (dynamic downloading of applications).
5. This ***post-issuance card personalization*** is the consequence of the need for the Service Providers to have a large base of target clients for their new services. It requires the previous agreement of the entity responsible for the security of the MA scheme. This entity can directly be the card issuer or a contractor. Post-issuance downloading of applications is not required by all the industrial sectors involved in MA projects. The TB7 generic architecture shall consider this feature as an option, so that it can be integrated either from the beginning of the Project or later on.
6. The subsystem which provides this functionality is the ***Card Management System (CMS).*** Different technical solutions for the CMS have been proposed by different specifications. The TB7 recommended CMS architecture must be in line with these existing solutions.
7. Finally, for commercial and security reasons it can make sense to store transaction related data within a ***Centralized Database***. This functionality raises specific technical, business and legal issues. To provide for design flexibility, the proposed TB7 MA architecture must be able to support this integrated subsystem information as an option.

Analogies with Micro informatics Industry

Several analogies can be made between the development of the micro-informatics industry and that of the multi-application cards and systems. As for the OS Personal Computer, the multi-application card operating system puts the resident applications at the same level in terms of offered resources, subject to the security policy of the card constraints. After selection, each application can be executed in an independent way and all the authorized card resources are allocated to the application as if the card was mono-applicative. The other resident applications are protected during the execution of a particular application by the firewall implemented by the OS of the card.

In the past, hierarchical models for the MA card were proposed, with one « core » application and other « satellite » ones. This "multi-service" but in fact mono-application approach is out of the scope of TB7/WP4 work programme.

The main qualitative difference with a PC is that, at present, cards need a terminal to work with not only for power supply reasons but because card applications are distributed. The off-card side of the application is directly resident in the terminal or in a distant server connected to the card terminal. The terminal and the card interact in a master-slave mode. After being activated, the card is in a waiting-

mode. When the card is requested by the terminal to perform some operation, the card process the command, generates the response, and then returns to the waiting mode.
In the same way that innovative computer models for the PC are in connected mode to an open network, the multi-application card takes full advantage of these possibilities when the terminal allows the card to access Internet resources. In principle, these connections could be done in two ways : Either the terminal communicates to the server, establishing a TCP/IP connection or the Terminal acts as a DNS for the card. An end-to-end channel can then be set between the card and the Web Server. The terminal becomes transparent with no modification of the application data. This requires the direct processing by the card of the Internet Protocols. Integration in the card of the TCP/IP stack is required. The card is then granted its own IP address and act as a full WEB Server and provides services over the Net.
The multi-application card is expected to be widely used to secure access to Internet resources, either from fixed or mobile terminals. The objective is giving business and individuals the ability to deliver/gain access to new services in a *secure, interactive and friendly* way. These features are supported by a System Architecture enabling network mobility of the software: Automatic secure delivery of software across the network to computers (terminals and/or smart cards) that run the software.
A final point: If the card has to make secure an e-commerce or home-banking application resident in the PC in a private environment, a business/technical issue is the interconnection of the card. At present, all PCs include a USB port which has became a de-facto standard in the micro informatics industry. Schlumberger Sema is currently proposing a USB technology card, the e-gate concept, which can be directly connected to the USB port of the PC with no need for an intermediate reader. The card is directly powered by the USB port and generates internally the clock required to communicate with the PC. This solution is currently under an ISO standardization.

2 CARD TERMINOLOGY

2.1. Multi-application related concepts

Some confusion remains concerning the following concepts. Sometimes they are used in an interchangeable way :

(1) ***Multi-application Card*** : Smart Card able to host several applications, managed by the Card Operating System in an independent way and selected by an AID (Application Identifier). Each application has its application provider which is liable for any damage to the smart card as a consequence of a wrong execution of the application (corrupted data in the smart card linked to the same/ other resident application or definitive card destruction).

(2) ***Multi-Services Card :*** Smart Card that has a single application and can be used for multiple services. An example is a National Identity Card. The successful authentication of the card to the Administration Server, enables the access of the cardholder to an e-government set of services.

(3) ***Domain of Services :*** In a multi-application card context, refers to the set of services provided by a single Application Provider. A single MA may contain several Domains of Service.

(4) ***Transaction*** : Standardized or proprietary exchange of messages between the smart-card and the terminal that, if successful, a service is rendered to the cardholder. In a multi-application context, an application is to be successfully selected by the external entity for the transaction to be run. Usually this selection is subject to previous identification and/or authentication of the selecting entity.

(5) ***Open Standard Card Platform*** : A subset of the Multi-application Card and of the MA Infrastructure in charge of the storage and execution of applications (sometimes referred to as applets) written in a language independent from the Card Operating System**.** Usually an Open Platform enables the downloading/deletion of application data and software in the card after the card has been delivered to the cardholder.

(6) ***Multi-Application Scheme: Scheme refers*** to the legal venture led by the Card Issuer and their partners in charge of issuing a multi-application card and managing the multi-application System.

2.2 Interoperability related concepts:

API (Application Programming Interface) Library of runtime classes resident in the host computer that gives the programmer a standard way to access the host resources. In this Draft, we refer to (1) The Card API , that gives the on-card side of the application the standard access to the card resources (2) The Terminal API, that gives the off-card side of the application an standard access to the Terminal Resources

Application Portability : Ability of an application to be executed by any Open Multi-application Card. It requires that the application complies with the API which has been specified by the Open Platform on which the characterizes the

Smart-Card/Terminal Interoperability, Common ability of any card and any terminal to execute jointly an application irrespective of the card and the terminal specific technologies.

Interoperability requires that:

(1) Terminal and Card comply with a standard specifying the Card/Terminal Interface.

(2) Simultaneous presence of the on-card and the off-card side of the application

Multi-application v Interoperability : A smart card may be multi-applicative but not interoperable, because it communicates with a terminal with a proprietary protocol. The card is only able to conduct a dialogue with terminals implementing such non standard protocol.

As a starting point we can consider that each application resident in the card provides access to a set of services (*Domain of services*).

In principle, each application provider can implement its own security policy for the Domain of Services under its control. Smart card technology guarantees that the execution of any application does not impact the security of the other resident ones.

Proprietary Multiplication Card : can only execute application software specific to the card operating system.

Open Multi-application Card : can execute application code independent of the card operating system due to the integration in the card of a Virtual Machine. The applications are personalized in the memory card in a specific format which is generically known as ***applet.*** After that, they can be selected and executed in an independent way: The private data of an application will not be read or modified by another application.

Virtual Machine (VM): Software layer of the Open Standard Multi-application card, which interprets the commands of an application resident in the card in the language of the card microprocessor. The Virtual Machine provides a standard API consisting of a set of instructions and library functions, called primitives, to applications that will be executed in the same way on all implementations, regardless of the underlying hardware.

2.3 Multi-application System Related Concepts

The «System»: Is the technical infrastructure made up of cards, terminals and the back-office telecommunication infrastructure, including hardware and software, necessary to support a business model.

Terminal Open Platform : Component of the Open Platform which directly communicates with the Card in master-slave mode. Resident in the Open M(BNZSA card it is based on a virtual machine which executes applets. The Card Open Platform is seen from the external world as an API. If the applet (downloadable applications) complies with the API, then the open platform can execute it and the applet is portable.

The API and the Virtual Machine have to be specified together. This is usually done by a standardization body. For example, the Java Card Forum specifies the API for the Javacard.

And MAOSCO defines the API for MULTOS. The card open platform provides the secure runtime environment.

«Open Development Environment» *usually refers to the ability of writing applets for the smart card without any previous knowledge of the card technology.* This represents major progress for the smart card industry, where applications were previously tied to the proprietary card operating system.

At first, these applets can just be personalized in the card during the manufacturing process. After that, the applets can be deactivated, but new applets cannot be added to the card. The distribution channels for new services remain blocked.

The Open Standard Multi-application Card is usually operated within a multi-application environment. Other than the Card itself, multi-application environment may also specify the system around it. This may include the Terminal, the Application Server the Card Management System and a Key Management Authority. In practice these elements allow data to be loaded securely over insecure channels to cards in the field. The secure channel provides confidentiality and authentication of the transmitted data.

The Card Management System is the Back-Office software able to deal with the complex management of the cards emitted by the MA scheme. Each card may have a different application content (different applets resident on it). Each applet resident in each card has its own life cycle . The CMS has to be able to keep track of (1) The Card Life Cycle and (2) the individual Applet Life Cycles resident in the Cards.

From the cardholder point of view the most important service provided by the CMS is the ability to download/update/delete an applet on its card through a fixed or mobile terminal. This process must be secured in order not to weaken the security status of the Card. This can be achieved either by setting a Secure Channel between the Card and the Application Server, or by using an asymmetric loading mechanism whereby code and data is encrypted before dispatch and decrypted with the card. The applet is usually signed by the card issuer or accompanied by a certificate signed by a Key Management Authority on behalf of the Issuer. The card must verify the card issuer signature or the Application Load Certificate before accepting to install the application in its internal memory. That is why CMS and PKI are closely related.

Because of the dynamic memory allocation of the smart card, to add a new applet may require the reallocation of the resident applets in the EEPROM. This operation takes significant time.

The CMS will add a minimum of one new server to the existing infrastructure. This server has to meet special security requirements and, in particular, it must be physically secured.

Another theoretical bottleneck is the limited data transmission rates between the card and the terminal and between the terminal and the server in wireless environments (mobile over the air downloading). This raises the issue of how to charge the cardholder for the allocated bandwidth. These problems are addressed by WP3 and WP5.

Closed-Systems: Is one in which the card is issued by a single entity and can be used to access this entity's own services.

Multi-services provided by the Card : The set of services provided by the card after the successful selection of one card application and subsequent transaction initialization. This concept is close to that of Domain of Services.

2.4 Services vs Applications

The service/s provided by a smart card requires the successful execution of a transaction between the terminal and the card. The transaction consist of a pre-defined sequence of messages exchanged between the "terminal-side" of the application and "the card-side" of the same application. This message exchange is supported by a mutually agreed transport protocol standardised by ISO. "Card Applications" are always distributed: Part of the application is resident in the terminal and the other part is securely stored in the card memory. The card-side of the application consists of an organized set of private data internally organized as a file tree. The terminal-side of the application can only gain access to the card-side of the application if it proves to the card that it holds the necessary rights. Otherwise the card denies the terminal access to its internal applicative data, the transaction aborts and the service is not provided.

3 THE CARD OPEN PLATFORM : THE OPEN MULTI-APPLICATION CARD AND THE SYSTEM BUILT AROUND IT

3.1 The Role of the Open Multi-application Card in the System.

The MA smart card and the supportive system built around it, complement each other in a way similar to the physics of the « vases communicantes » -- in other words, there is at least a partial trade-off between the two of them.

When the card just acts as a secure identifier, the cost of the card is lowered, but much more of the operation has to be endorsed by the system (Terminal + Back-Office). The reason is that

continuous on-line message exchange is required. The load on network resources and the communication costs are thus increased. Therefore, a realistic Business Case relies on local low-cost communications.

Conversely, if the card is a secure portable database, the transaction can be directly performed by exchange of messages between the card and the terminal with very reduced operation by the back-office system (just to capture the transaction data for billing).

The objective is to get a secure and dynamic platform on which applications can be easily developed and distributed at low cost with little effort by the cardholder: The TB7 Service Platform. A distributed Client-Server architecture can achieve this. Such an architecture provides the framework to deploy services across a wide range of card accepting devices. This approach is only actually useful to the extent that Card Terminals present the same interface to the background system. This is the objective of new standards like STIP (Small Terminal Interoperable Platform).

Finally, we want to point out that the efficient insertion of the card in the system relies on the appropriate internal organization of the card data. This organization must allow for (1) Fast acknowledge of the card services and (2) A straightforward way to select them. The next paragraph details the standard logical architectures for the smart card and some implications in the multi-application context.

Basic Principles of Card File System Organization : « File » versus « Application-oriented » organizations

1.ISO 7816 organizes the card internal structure as a file tree, in a DOS architecture. This means that any access to the card data is a two-step process : (1) The selection of the file where target data are resident , (2) The execution of the command. This organization is shown in figure below

2.The emergence in the market of cards based on MULTOS and Java Card Platforms which are in fact organized as a set of selectable applications might require some complementary approach in the traditional data organization of the smart card. For these platforms, developed in an Object-Oriented language : The Application is seen as an Object, which accepts conditional execution of commands (APDU) on it.

3.We must point out here that the mechanism of the selection of an application by its registered AID as specified by ISO 7816-5, allows the external world to access card resources with no previous knowledge of the card file structure.

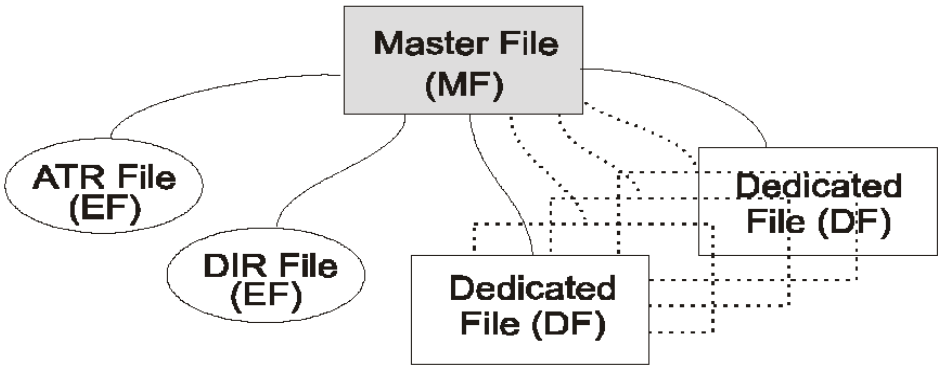
4.In addition, some ISO 7816 commands like GET DATA or PUT DATA, allow data processing in the card with no previous file selection. More recently the 7816-9 authorizes the definition of security attributes associated with data structures other than the file ones (ie at record level).

5. It can be assumed that even if the ISO 7816 include some access mechanisms in order to deal with different card architectures whilst keeping interoperability, these mechanisms require further clarification.

6.Some misunderstanding remains in the card industry concerning the apparent incompatibility between the ISO file structure and the Application-oriented Organization proposed by JAVA and MULTOS cards. Both approaches are in fact complementary. In the context of this document a Multi-application Card can support specifications from either one or both organizations.The actual storage of the data and the implementation of the functions within the card are out of the scope of this document.

This paper introduces both platforms. Further considerations concerning applet interoperability for both architectures are included in #4.

The multi-application card is integrated in the system basically to secure all the transactions. The mean reason for the card's existence is the security that it provides for the confidential data stored in it. This secure storage allows for the use of these data to cryptographically protect the exchanged messages. It does not make sense if the cost of improving the functionalities of the card is a weakening of the card security. The open characteristic of the multi-application card, involves new risks that have to be dealt with by specific design measures validated by standard certification processes.



3.2 Security Model Concept :Platform level Security & Application level Security

3.2.1 Security and Business Model

On-line revenues currently constitute a low proportion of total card revenues. Thus, current Internet business models are still much more concerned with growth, rather than security of purchases. However, as this revenue grows, security will become the key differentiator. As mentioned before, Multi-applicative cards and systems present new challenges concerning security. Encouraging the deployment of MA platforms relies on setting up clear security policies.
The implementation of a security policy is always a matter of balancing cost and inconvenience against the likelihood of the information held being compromised.
This document provides some guidance for traders when developing a MA system

3.2.2 The problems to be addressed

The Grounds:

1. The Application Provider requires a minimum level of Platform Security before deciding to download an applet on the card. Somehow the applet must assume that the OS will not be aggressive and is to be so designed This confidence is granted ***by the compliance of the Platform with a Security Policy*** which can be verified by any Application Provider..
2. In a similar way the Card Issuer must trust the Application Provider and the applets of their own. The Card Issuer is responsible to all the other Application Providers, for ensuring that a new incomer is not acting as a "Trojan Horse" (typically an applet able to identify a chain of characters which looks like a Credit Card Number and transmits it to a pre-selected URL using TCP/IP). The Application provider must therefore go through some certification process consistent with the security policy of the platform. This is just a scenario of the general problem of trusting a program that you haven't developed.
3. It is technically feasible to design a secure OS that guarantees that an unsecured applet cannot degrade the security of all the other private data in the card. But this OS may still allow an unsecured applet to be broken. This is bad for the card issuer image, which is going to require the application provider to give either certification, or proof of a minimum level of security of their applets against attacks.
4.If the new incomer is going to share confidential common data stored in the card (i.e. Keys for Signature, or an Electronic Certificate), the platform must provide mechanisms to prevent the new incomer reading these data in the card. More generally, some common resources must be « sold » to the new incomer. Yet it must be possible to prevent the reselling of these data to a third part, even with the complicity of an existing scheme participant.
5.From an economic point of view it is interesting to have the applications in the card sharing some common data and/or security services (ie, Digital Signature Services): This saves memory and allows the card issuer to bill service provides each time their application requires to produce a digital signature using one private key supported by a common certificate.

6.This cooperative approach again raises new challenges for security, which can be summarized as:

- To find the optimum balance between security and the intrinsically limited card memory constraints
- To find the balance between data sharing, security, memory allocation and improvement of cooperative schemes. Roughly, cooperation cannot jeopardize the security of the card but memory is money and cannot be wasted.
- To specify the security needs to be integrated in the applicative code and decide whether they can be provided by the off-the-shelf security kernel of the card.

7.Simple configurations where each card application has total control of its data are covered by the existing standards. But practitioners in all areas are asking for a common data area in the card, with the security model giving control on which applications get the right to read or alter which data. The next paragraph provides some insight on how to model the security issues.

3.3 Global Security Policy for the Multi-application cards

Generic Considerations for Open Multi-application Cards

1.The Card Security model is one of the key features that makes the card suitable for networked environments, which are very vulnerable to attack (Some authors consider that any networked PC is attacked on average each hour!)

2.This concern is more important if our Multi-application Architecture is an environment in which the software can be downloaded across the network and then executed in the card. In the access to the WEB it is likely to download applets from untrusted sources : How can it be guaranteed that a dynamically loaded new application does not corrupt existing ones ?

3.In the traditional security scheme, the most direct filtering method is to prevent any code that the user does not trust being loaded in your local machine. As mentioned, this is difficult to achieve in a WEB environment : The class files for an applet are automatically downloaded when browsing. This is why Security Mechanisms have been developed early on for Java code.

4. In the Java environment (not only for the Java Card) scenario the Sandbox Security Concept has been developed in order to address the security problems of the mobile code. One of the greatest advantages of the Sandbox is that it can be customized. Just one word of caution : The Sandbox is commonly used to refer to the Virtual Machine in a wide sense. The applet is executed within a secure execution context called the Sandbox with basically three security mechanisms :

- The Class File(Bytecode) Verifier
- The Class Loader
- The Security Manager

which are analyzed in #4.3.1

5. The Java Card technology inherits the security properties of the Java Programming Language, but adds specific concerns which arise from the nature of the card, and, in particular, of the limited memory resources. The Java Card Virtual Machine is divided into two separate modules, resident respectively in the Terminal and in the Card. The Terminal side of the VM preprocessed the Applet and packaged it into the CAP-file format, which is specific to the Java Card technology. MULTOS on the other hand utilizes on-card security mechanisms implemented by the Operating System combined with a Key Management Infrastructure to ensure that applications are genuine and they cannot be affected by other applications on the card.

6. Whilst Java Card security against malicious applets has long been discussed , it is necessary to define a global security policy for multi-application smart cards. The main reason is because of the need for secure information sharing between the resident applets. Existing Java-based security policies have not succeeded in providing either full isolation between applications or allowing secure sharing of data between applications without disclosing any information to a third " spy application" resident in the card.

Objectives of the Security Policy: Platform Security and Application Security

In our context, the Security Policy is a Multilevel focused approach to solve some of the Security Problems **specific to the Open Multi-application card,** for which secrecy and integrity of the data must be guaranteed, despite the fact that:

1. The card can host applications (Data + Code) developed by organizations that do not necessarily trust each other
2. It is assumed that the end-user may be able to download unsecure applets into the card, despite Card Issuer Control
3. Some of the applications must be willing to exchange data in a confidential way regardless of the eventual presence of malicious code in the card. Adding communication channels must not jeopardize the security of the communicating parts and this communication cannot impact the security (code and data integrity and secrecy) of the other resident applications

This Security Policy is achieved by defining a set of requirements for the Platform (OS and VM) hosting the applications and for the applications themselves. The degree of achievement of these security objectives is proved by the certification process of the card in relation to a pre-defined Security Evaluation Profile (eEurope SCC/TB3).

(1) ***The Security requirements for the OS*** ("At platform level") managing multiple applications, which is under the direct responsibility of the Card Issuer (We focus here on the most common scenario of one Card Issuer, the only authority responsible for the card and for the definition of its Security Policy)**. The main objectives of the Platform Security are:**

- To ensure that information is not disclosed from one application to another, even in the presence of a "Trojan Horse". The Operating System of the Card will prevent unauthorized interference between the applets even if mutually hostile. This means that the OS does not make any assumption concerning the applet.
- To protect against unauthorized tampering or sabotage of the application data stored in the card (Data Integrity). Their code/data must neither be observable nor alterable by other code resident in the card. However, some application providers may decide to classify their applications (code and data) into different levels of integrity and confidentiality
- To protect against denial of service

The Security at this level is intended to:

1. Guarantee the effective firewalling between applications
2. Guarantee the level of quality of the Verification Security Services of the Platform (#4.3.1) as well as the cryptographic services provided by the card.

(2) ***The Security Requirements for the Applet*** ("At application level").

As mentioned, the applet must undergo some certification process before being authorized to be resident in the common platform. The important point here is that this certification must be done, even if the OS is so well designed that a poor applet cannot jeopardize the security of the platform. But if this application can be broken, not because of the OS failure, but for its own weakness, the image of the card issuer will be damaged, and the security of the card will appear suspicious.

(3) The Security Requirements for Object Sharing between Applications ("At logical level")

If two application providers agree, communication between their respective applications must be authorized by the OS, which sets a communication channel between application AID1 and application AID2. The channel can read the content of a file from AID1, to write it in an intermediate file that only AID2 can read.

This communication channel is secure, meaning that

1. It allows the exchange of data only in the way that the communicating parts have agreed on.

2. The communication cannot be observed or manipulated by other programs. This is probably the most technically hard objective to comply with. See next paragraph.

The Transitive Communication

If application AID1 sends some information to application AID2, AID1 can no longer control how AID1 uses the information. AID2 can then send them to AID3, potentially hostile to AID1.
If AID1 trusts AID2 and AID2 trusts AID3, it is untrue that AID1 necessarily trusts AID3. Because it is difficult to avoid this problem, it is expected that AID1 and AID2 will sign some type of NDA agreement.
Some models based on previous work by Denning (1977), Bell/LaPadula (1976) and Biba (1977) have been suggested to deal with the Transitive Problem in the multi-application card context. These Security Policies are still in their feasibility phase of development. Basically the OS must have some control of ***any*** data exchange between resident applications, and to prevent indirect transfer of data between applications that haven not explicitly so decided.
1.The Card OS must guarantee this selective access procedure, by for example preventing that an applet AID2 having access to data applet AID1 could transfer these data to another non-authorized AID3 . (Applet AID3 may have legal access to any data held by applet B by agreement and /or contract)
2.The above example highlights that the OS must have control of the data flow between applets beyond the individual policy/collateral agreements of each/between application provider/s. A "logical level policy" for the Platform is then required (Card Sharing Mechanism). With the current business models only the Card Issuer is able to guarantee this to the Application Providers of its scheme.
3.In Multi-Application schemes the obvious case to be avoided is the following : A and B are competitors. Both have set co-branding schemes with a common partner C, in two different MA schemes. C could pass data from competitor A using the scheme 1 A to competitor B or vice versa.
Another case is an applet B reading confidential data from applet A (personal medical record).
Attempts have been made in the past to standardize, in ISO 7816, a card internal structure supporting a Security Policy. The proposed solution was based on the definition of matrices with crossed Access Rights between the resident applications. These matrices were to be upgraded by the OS after each new applet downloading or deleting. The proposal was finally refused because of the memory resources required in the card and the complexity of its management.

However, this initial analysis requires some refinement and shall be completed in a new version of the document. The interest for the TB7 is that a Global Security Model for the Multi-applicative card may help to create the Shared Liability model identified by WP2.

After this introduction to the Open Multi-application Card security issues, the next chapter focuses on the practical platform solutions currently available in the market: Java Card and MULTOS cards. Chapter 5 will deal with the respective Card Management Systems designed around these specific platforms and the efforts produced towards a common interoperable standard.

4 JAVA and MULTOS CARDS

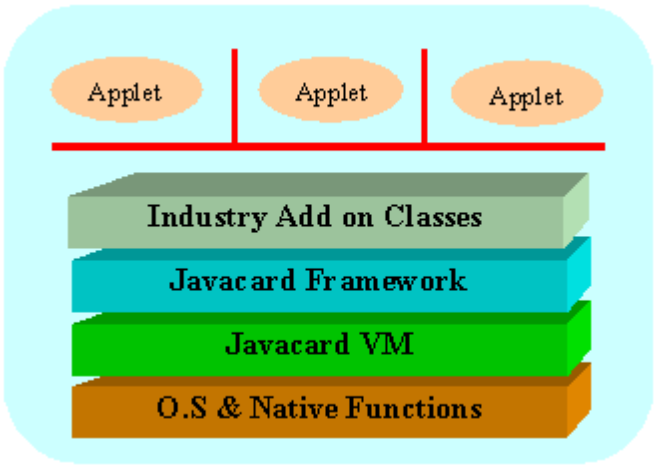
4.1 PRODUCT DESCRIPTION

4.1.1 JAVA CARD

Roughly, a Java Card platform consists of a Java Card Virtual Machine build on top of an operating system. The later usually resides in the ROM of an Integrated Circuit and is implemented in native code. The OS basically offers cryptography, I/O, memory access and management services.
The JcVM consists of a byte code interpreter as well as some management information. The JcVM is responsible for the applet selection and deselection, byte code execution, APDU dispatching, runtime verification, memory management, etc.

To help developers to write Java Card applications a set of Application Programming Interfaces is defined within the Java Card Framework. These API's offers basic services and are industry-independent. Standardization bodies such as Java Card Forum or ETSI define API's which are industry-specific (banking, GSM, Etc.).

The applets are converted Java applications that are uniquely identified by an AID. The applet code is executed by the JcVM which maintains a firewall and control communication between different applets. The applet can access the OS and native functions services through the appropriate API's. Figure 1 shows an abstract view of a Java Card platform.



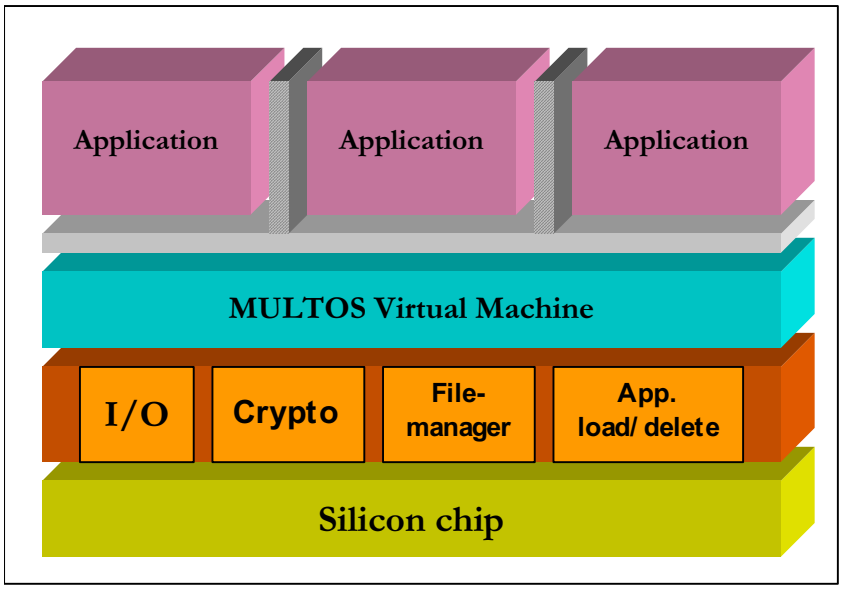
4.1.2 MULTOS CARD

MULTOS define both the Card Operating System and the Virtual Machine for interpreting the application code. Because of its origin as a high-security platform for financial and identity applications MULTOS has a comprehensive security architecture behind it.
Applications are typically written in C or Java and compiled into the Multos Executable Language (MEL) , which is a low level RISC (Reduced Instruction Set Computer) language specific to MULTOS and optimized for the card. MEL applications are therefore small and fast.
The Platform Independence of the MULTOS applications is achieved through the implementation in the card of a Virtual Machine. As with Java Card the application is run on top of it. There is also a standard API between the applications and the OS. This enables applications from different vendors which comply with different standards (EMV, BO') to coexist on the same card. It also enables developers to write a MULTOS application once and run it on top of different MULTOS hardware platforms without modification.

The key differential elements of the MULTOS card are:

1. A highly secure architecture (evaluated to ITSEC High 6)
2. A mature asymmetric infrastructure for dynamic loading of applications allowing a great flexibility for card issuers to share their card infrastructure with third parties

3. Proven Interoperability on three levels:
   Byte code execution
   Application Loading mechanism
   Security enforcing functions



4.2 SECURITY ARCHITECTURES

4.2.1 JAVA CARD SECURITY FRAMEWORK

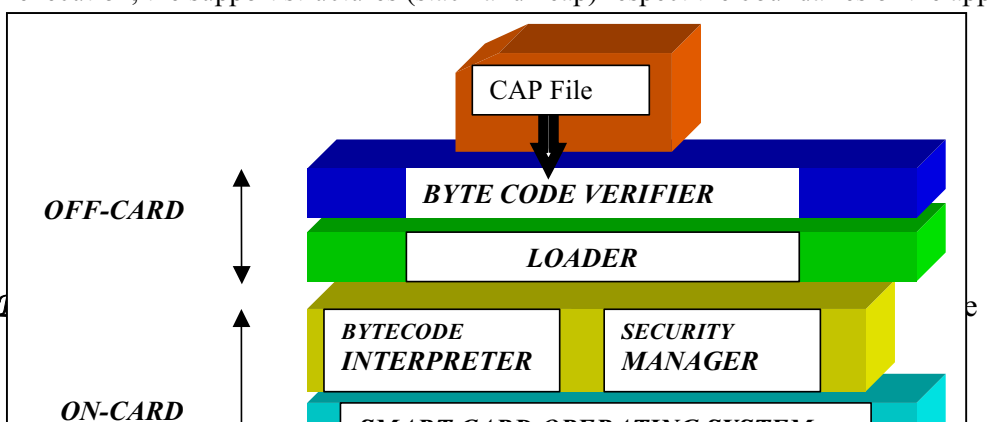
The Java VM consist of four functional elements: The Bytecode Verifier, the Loader, the Bytecode Interpreter and the Security Manager, organized as shown. Due to the small amount of resources usually available on hardware where the Java Card platform is implemented, the Java Card specification enables on-card verification of CAP files, but does not mandate it. The Bytecode Verifier and the loader are usually implemented in the off-card side of the VM. Each one ensures a basic security function:

1.The Bytecode Verifier is the first filter which checks bytecodes of the class files of the applet (constants pool, syntactic correctness, object inheritance hierarchies, state of the variables) to verify that they are safe to be executed by the Virtual Machine

2. ***The Loader*** processes the checked data and packages them into the CAP format. Then the CAP format is sent to the on-card Virtual Machine. This is a critical point for the Security Policy of the Card. This applet in the CAP format file must be preserved against manipulation and some crytpographic function to ensure its integrity must be provided by the loader (digital signature).

3. The Bytecode Interpreter fetches, reads each bytecode along with its input argument, interprets them to the specific machine code of the microcontroller of the card and ensures they are executed.

4.The Security Mechanisms are completed by the ***Security Manager*** which has the authority to stop at any time the processing (interpretation and subsequent execution) of the bytecodes by emitting an exception handler. The Security Manager verifies that during the applet execution, the support structures (stack and heap) respect the boundaries of the applet.



The JCVM, which ensures the above basic security mechanisms is the core of the Java Card Runtime Environment (JCRE).

JCRE (Java Card Virtual Machine, the Java Card Application Programming Interface classes and support services) implements a firewall during the applet selection and execution according to the Security Policy at the Platform level decided by the Card Issuer (see #3.3).

Applet firewall and Sharing

In addition to the CAP File Verification, the Java Card Runtime Environment enforces applet isolation by implementing a firewall between applets. All the instance method invocations are trapped by the firewall that verifies the validity of the invocation. However, the firewall does not impose a complete confinement of the applets, but enables a controlled communication through shareable interfaces.

However the access to public static fields or methods is not restricted by the firewall.

Memory management and transactions

As is the case for the Java Programming Language, there is no memory management or pointer arithmetic at the applet level. And the array bounds are checked at runtime by the JcVM to avoid buffer overflow vulnerabilities.

In addition, the JCRE offers a transactional mechanism to guarantee the atomicity on persistent objects. On the other hand, the JCRE provides services to create and manage transient objects (arrays, actually), which are objects whose content does not persist from one CAD session to another. These transient objects are never stored in persistent memory technology and are cleared at the occurrence of certain predefined events.

The JCRE also manages the APDU buffer, which is a transient object owned by the JCRE. In particular, this buffer is cleared before each applet selection and its reference cannot be stored in class variables or instance variables or array components.

4.3.2 MULTOS CARD SECURITY FRAMEWORK

MULTOS mandates, that all implementations are ITSEC E6 approved and hardware tamper resistance evaluated. This is currently the highest level of security approval.
The six main Security Objectives of the MULTOS Platform are :

1. Applications are only to be loaded onto a card with the permission of the Card Issuer
2. The application load process must be able to guarantee the authenticity and confidentiality of the application code and data
3. Loading an application must have no effect on the code and data of existing applications
4. Removal of an application and consequent reuse of the application space are only to be performed with the authorisation of the Card Issuer
5. Removing an application must have no effect on the code and data of the remaining applications
6. Applications are to be segregated from other applications - an application may not read from or write to another application's code or data

Application verification

An application has to be loaded using certificates specific for each card or for a batch of cards of the same version and implementor, the application load certificates (ALC). The ALC contains a declaration of the memory required by the application (for creating the security firewalls), the application id and a flag which constrains the use of cryptography called from the MULTOS operating system by the application. The latter is required to address regulatory restrictions associated with the export of high strength cryptography.

The verification of these certificates is based on an RSA encrypted signature. The used global key certification keys are only known to the MULTOS CA, which generates the certificates on behalf of the issuer of the card.

After loading an application, a checksum is calculated over the code of the application. After each selection of the application, this checksum is recalculated and checked against the stored value. If these checksums differ, the card aborts.

Memory management and transactions

The MULTOS OS guarantees the integrity of the static memory of each application on the card. For each application loaded on the card, a separated area is reserved. The application does not get to know the real location of this memory for it always uses virtual memory addresses. Each command of the VM is checked against the boundaries of the memory. If the application tries to use an illegal memory area, the card aborts.

The private volatile memory area containing the stack and session data is used by all applications. It is cleared after selecting a new application to keep the application data secret.

However access to the volatile public data area is not restricted by the firewall. The public memory area is mainly intended for external communication. It can also be used for communication between different applications on the card.

The MULTOS OS also allows the applications to perform transaction management, meaning that multiple EEPROM writing can be bound together in a transaction.

> The OS manages all system resources and controls their access. The VM relies on the OS services to achieve firewalling.
> The security mechanisms are available to the applications through an API:

Each application has to manage the security attributes and status via the API. The same principle applies for the security attributes linked to the commands specific to each application.
Security of the MULTOS Platform is probably the major strength of the technology because it inherits:

-Complete definition of the Security Model

-High Security Level Achieved: The security target for the MULTOS Platform is ITSEC E6 High

-Complete OS Specification, including dynamic load and delete of applications, Issuer control procedures for applet management, Virtual Machine Runtime Environment and common Security Model

-The global key certification keys are only known to the MULTOS Key Management Authority (KMA), which generates the certificates on behalf of the issuer of the card.

Table : Summary of Security features of Java Card and Multos

	JAVA CARD	***MULTOS***
Language specific verification features	Strongly typed Initialization of variables Access control to classes, methods and fields	On card MEL level verification guarantees integrity through load mechanism. Authoring language and target platform Independent
Types of memory area	Persistent Transient	Private static Private volatile Public volatile
Memory management	No pointer arithmetic Array bounds (at runtime) Management of transient object (arrays)	At runtime, verification of access to the application code and to private static and volatile mem. Areas (in case of violation the card aborts) Note : private volatile areas are reset before each new selection.
Atomicity	X	X
Transaction	X	X
Abend of transactions	If the applet returns during a transaction, the JCRE must abort the transaction and recover the data	If there is a delegation during a transaction, the transaction is aborted. If the application returns during a transaction, the transaction is aborted and the data recovered.
Code sharing	JCRE (access to everything). Objects are either shared or not. JCRE Entry Point object (only public methods) Shareable interface object (SIO) Public static methods (not restricted by the firewall)	Delegation (via APDU) Codelet. No common memory space code sharing or direct interaction. Code integrity guaranteed.
Memory management during code sharing	For JCRE entry point object, there is a context switch to the JCRE context. For SIO, there is a switch context. For public static methods, the current context is used.	For delegation, there is a switch context. Private volatile areas are not reset (notion of "session"). For codelet, the current context is used.
Data exchange	Global arrays (owned by the JCRE) Shareable interface object Public static fields (not restricted by the firewall)	Use of the public volatile mem. area. Protection against application data corruption by other applications through use of firewalls.
Native	Access to native methods is possible but	Export/Import Data specified for native

Methods calls	vendor dependant.	method calls (version 5)
Applet/ Application Loading and Deleting	Applets do not have to use a defined load mechanism and the card manager is implementation specific.	Applications must be loaded through the MULTOS load mechanism and verification mechanism using card and issuer specific certificates

4.3 APPLET AND CARD LIFE CYCLES MANAGEMENT

4.3.1 JAVA CARD MANAGEMENT

4.3.1.1 .Applet Loading

The processes of loading a CAP file representation onto card memory is referred to in the Java Card specification as the installation process. The Applet Installer is an optional part of the JCRE specifications. From the CAD point of view, the Installer behaves like any other applet. Its selection is based on its AID and it processes and responds to the APDU's it receives.

Although an Installer may be implemented as an applet, an Installer will typically require access to features that are not available to "other" applets. For example, depending on the JCRE implementor's implementation, the Installer will need to:

- Read and write directly to memory, bypassing the object system and/or standard security.
- Access objects owned by other applets or by the JCRE.
- Invoke non-entry point methods of the JCRE.
- Be able to invoke the install method of a newly installed applet.

The JCRE specification does not mandate any specific Installer implementation and loading protocols, such as Visa OP protocol, may be validly implemented.

4.3.1.2 Applet life cycle Management

In the JCRE specifications, the applet's lifetime begins at the point that it has been correctly loaded into card memory, linked, and otherwise prepared for execution. Applets registered with the register method exist for the lifetime of the card. The JCRE interacts with the applet via the applet's public methods install, select, deselect, and process. More detailed applet life cycles can be defined at the applet level or at the JCRE level as long as they are compatible with JCRE/Applet interaction model. In particular, Visa OP and ETSI GSM 3.19 documents.

The method of installing

The main task of the install method within the applet is to create an instance of the Applet subclass using its constructor, and to register the instance using the register method. Usually all other objects that the applet will need during its lifetime can be created at this stage. Any other preparations necessary for the applet to be selected and accessed by a CAD can also be done. The install method obtains initialization parameters from the contents of the incoming byte array parameter.

The method of selecting

Applets remain in a suspended state until they are explicitly selected. Selection occurs when the JCRE receives a SELECT APDU in which the name data matches the AID of the applet. Selection causes an applet to become the currently selected applet.

Prior to calling select, the JCRE shall deselect the previously selected applet. The JCRE indicates this to the applet by invoking the applet's deselect method.

The JCRE informs the applet of selection by invoking its select method. The applet may decline to be selected by returning false from the call to the select method or by throwing an exception. If the applet returns true, the actual SELECT APDU command is supplied to the applet in the subsequent call to its process method, so that the applet can examine the APDU contents.

The method of processing

All APDU's received by the JCRE are passed to the process method of the currently selected applet with an APDU class instance as argument.

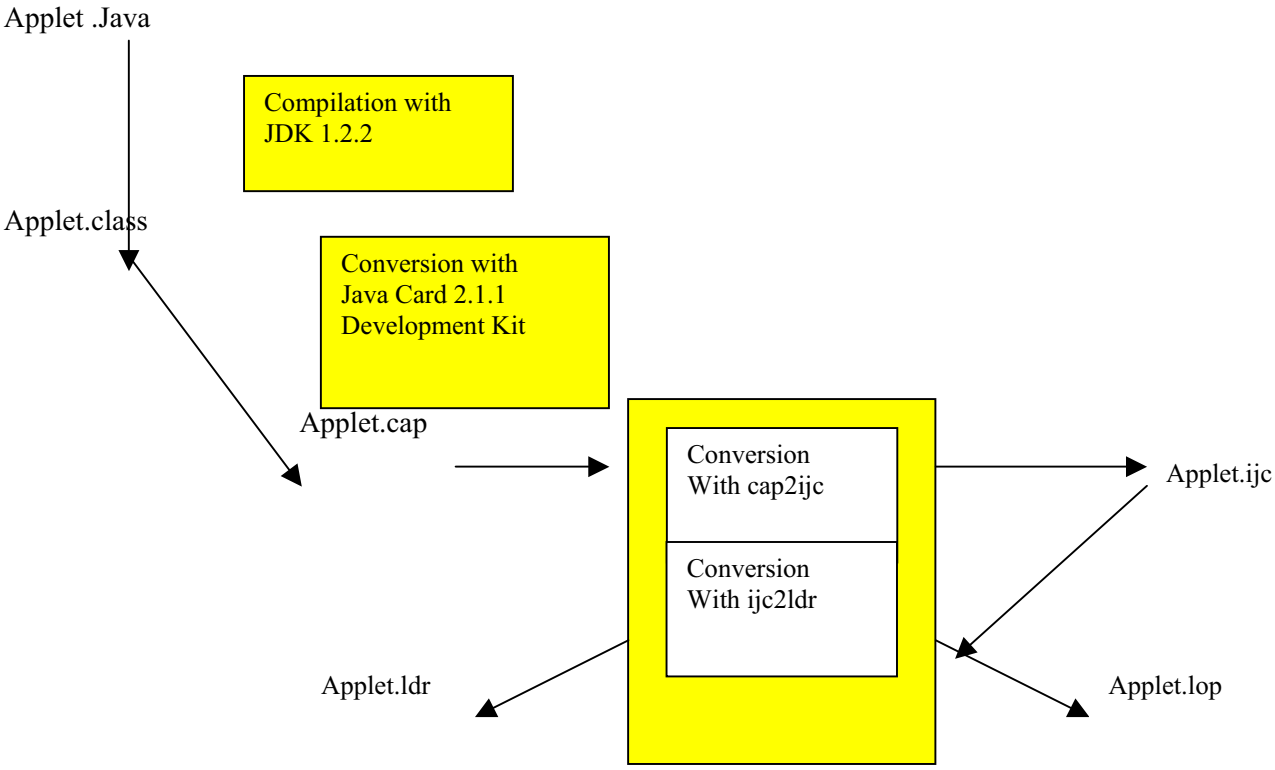
The method of deselecting

When the JCRE receives a SELECT APDU command in which the name matches the AID of a registered applet, the JCRE calls the deselect method of the currently selected applet. This allows the applet to perform any cleanup operations that may be required in order to allow some other applet to execute.

4.3.1.3 Card life cycle Management

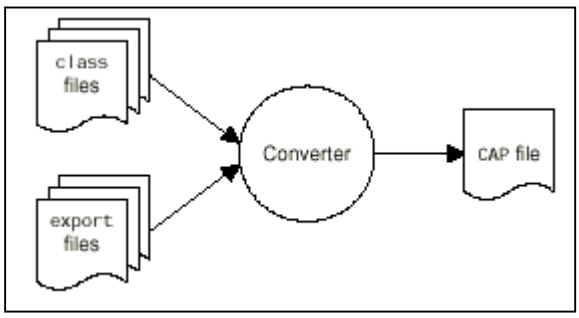
Java Card specifications do not mandate any card management policy. The Java Card Virtual Machine life time is the life time of the card, which is defined by the card manager. The Java Card VM appears to run forever. When power is removed, the VM only stops temporarily. When the card is next reset, the VM starts up again and recovers its previous object heap from persistent storage. It is for the JCRE implementors to define a specific card management policy. This policy may be enforced by a Java Card applet (with special privileges) or by a native application. Some industry-specific or inter-industry card management frameworks have been defined or are under construction. The Visa OP card management framework exemplifies the trend in this area.

Example :The Java Applet Interoperability



4.3.1.4 Building a Java Applet

1.The first thing the programmer does is to generate a Java Source code using a text editor.
2.Then the applet is built using a standard Java compiler used to transform the Source code into Java classes, in bytecode format
3. After that, the class files are transferred to the off-card side of the JVM, which performs some checking and verification procedures as explained in #4.2.1. If these controls are successful, the converter (core of the off-side JVM) converts the class files into a CAP file as shown in the figure : The converter takes as argument the Class files, the Export files (if any) and produces the corresponding CAP file.



NOTE An Export file contains name and link information for the contents of other packages that are imported by the classes being converted. When an applet or library package is converted, the converter can also produce an export file for that package.

4. Finally, the applet is loaded into the smart card in the CAP file format, and can be run after being selected by its own AID. The on-card JVM tests and interprets the bytecode line by line, generating machine instructions for the smart card microcontroller.

In order to reduce the time for the programming of card applications there are four packages that provide an API. Only one, the ***javacard.framework*** is mandatory for the JCVM. A JavaCard applet must inherit from the javacard.framework.Applet class. This class offers the basic methods that are needed to implement a new applet.
The other three packages are :

The javacardx.framework is an optional extension which implements a classical ISO 7816-4 compliant data tree structure

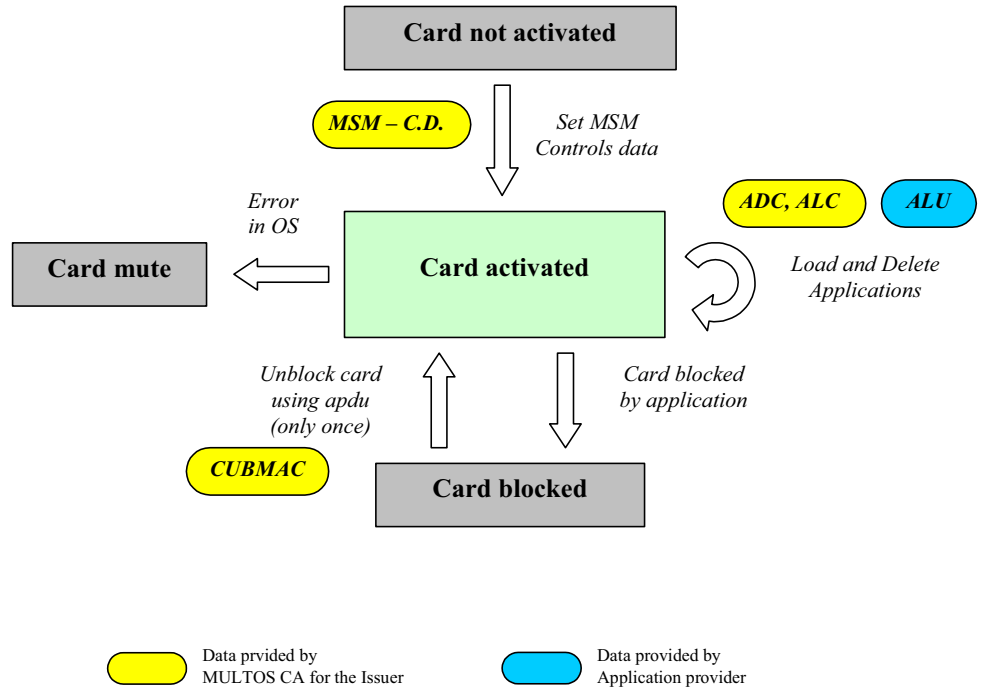
The javacardx.crypto contains cryptographic functions, which can be subject to exportation restrictions.

The javacardx.cryptoEnc includes the methods to provide encryption services to the applets

4.3.2 MULTOS APPLET IMPLEMENTATION AND MANAGEMENT

4.3.2.1.Card Life Cycle

The diagram below shows an overview over the card life cycle:



All data can be sent to the card using an insecure network and is or can be made specific to exact one MULTOS card. The issuer of the card – assigned by the MSM controls data – controls the card during the whole of its life.

The card life terminates either by going permanently mute (due to an error found by the MULTOS OS implementation) or after the second blocking of an application. MULTOS does not mandate applications to use the card blocking mechanism.

The application load process is controlled by a counter of unsuccessful load operations. If this counter reaches a limit, the card will always respond with “function not supported” on load commands, so that no new applications can be loaded to the card. Applications already existing on the card can still be used until they are deleted.

Activation of a MULTOS Card

The issuer or the card bureau (on behalf of the issuer) receives the chip with the complete MULTOS OS from the chip provider. The MULTOS Card with this chip has only to be activated by sending the

MSM-Controls data (MSM: MULTOS Security Management Controls Data, which the issuer obtains from the MULTOS CA. These data include implementor's specific data, the issuer identification, the activation date and other data. Beginning from this point, the card is specific to the issuer.

There is a limited number of wrong set MSM-Controls data commands that can be sent to the card. If the counter has reached its limit, the card will always respond with "function not supported". The card is not useable any more.

Card blocking/unblocking

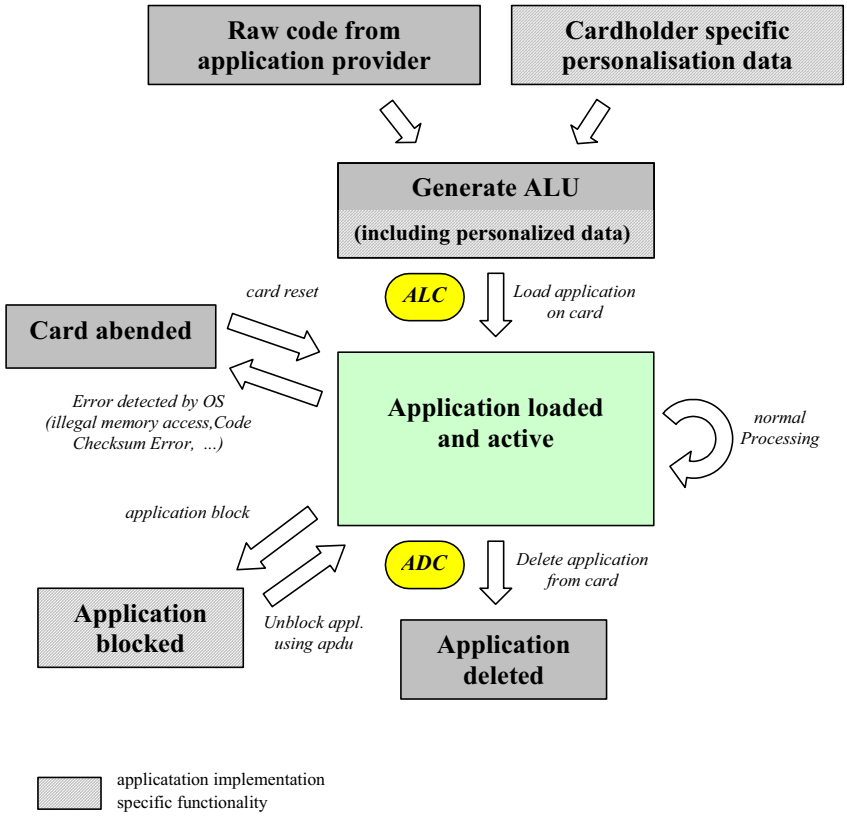
A MULTOS card can be blocked by an application, using a specific primitive. This could be done e.g. after detecting severe EEPROM degradation. Then the card can only be unblocked using a specific Card Unblock MAC (CUBMAC), which is specific to each MULTOS card and can be obtained by the issuer of the card from the MULTOS CA.

This cycle is limited in that the unblock command can only be sent once. That is a MULTOS Card that has been blocked may be unblocked; but should that MULTOS Card be blocked again then it cannot be unblocked a second time.

4.3.2.2 Application Life Cycle

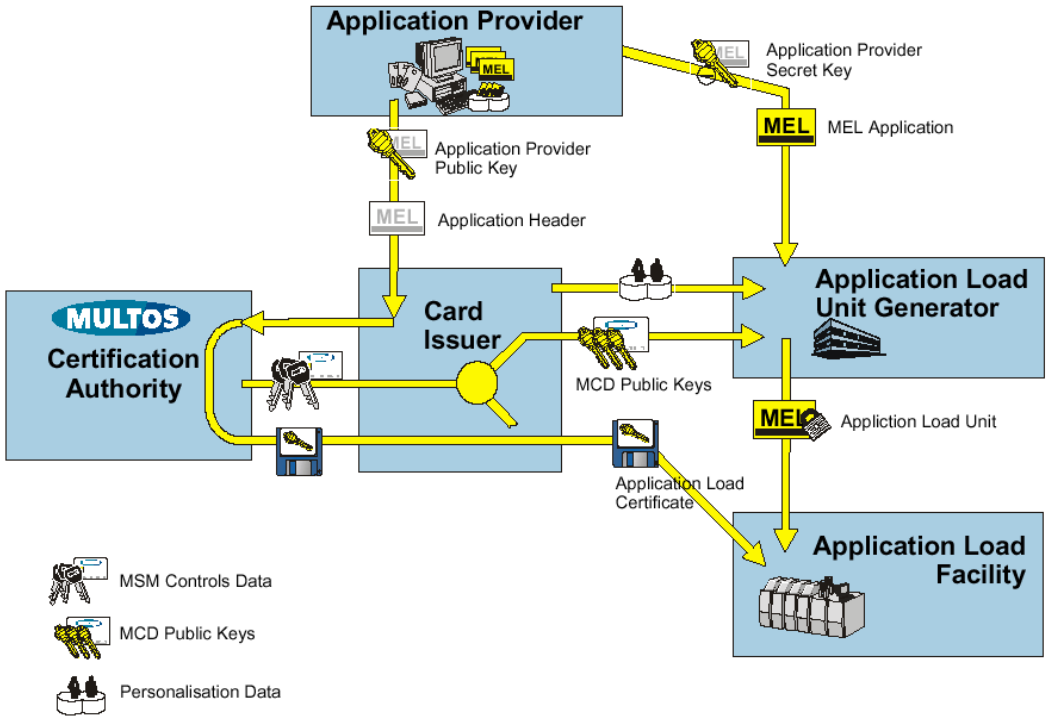
Each application on a MULTOS Card is controlled by the issuer. He can load and delete an application at any time. Further control –such as application blocking or unblocking, checking of expiry date – is up to the application itself. An integrity check of the code is performed by the MULTOS OS each time before selecting an application. If this check fails, the card goes mute.

The following diagram shows an example of an application's life cycle. It assumes the application to be personalised before loading to the card (see chapter 0) and that the application has implemented ablocking mechanism for itself (this is not supported by MULTOS). If the application does not block itself and if it is not deleted (using the ADC – Application Delete Certificate – of the issuer) it exists for the whole lifetime of the card.



Application Load and Delete Overview

This chapter gives an overview of the application loading and deleting process.
MULTOS application loading and deleting allows MULTOS card Issuers to create a card that is uniquely tailored to an individual customer or lifestyle. Card Issuers can add, modify or delete applications at any time during the card's life, even after the card has been issued. Applications can be downloaded anywhere, using insecure networks (e.g. the Internet, telephony or radio networks) without compromising security. For loading, the application is packaged in a protected file format, called an Application Load Unit (ALU). ALUs can optionally provide a mechanism to protect confidential application data, and a digital signature to allow the MULTOS card to detect a corrupt ALU during loading. These integral security features mean ALUs can be stored on any platform and transmitted over any network, whether or not it is secure.
An application can only be loaded or deleted from a MULTOS card with the permission of the card Issuer. This permission takes the form of an Application Load Certificate (ALC) and Application Delete Certificate (ADC). The MULTOS Certification Authority (CA) is responsible for providing the cryptographic services for the MULTOS scheme, including the generation of ALCs and ADCs for live cards. Only the MULTOS card Issuer can request ALCs and ADCs for its card base.
The diagram below shows an overview of the process of Loading and Deleting Applications:



- Makes use of Asymmetric Key Cryptography and digital certificates– such as in a Public Key Infrastructure.
- The benefits of asymmetric key mechanisms are:
  - that an application provider can send encrypted application code to the card, without having to submit application code in clear or symmetric keys to the card issuer.
  - the card issuer can allow applications from third parties to be loaded to the card without the need to share secret keys with the application provider.
  - * Symmetric key loading mechanisms mean that trust must be established between card issuer and other application providers. With asymmetric mechanisms this is not necessary.
- Access to operational and proven MULTOS Key Management Authority, which acts as the trusted third party which generates the digital certificates at the request of the card issuer.
- The process by which certificates can be generated is entirely under the control of the card issuer.

Loading mechanism security claims form part of the MULTOS ITSEC E6 High security rating

Application Load Unit

To load an application onto a card, the Application Code must first be formatted into an Application Load Unit. The ALU consist of

- Code record
- Data record
- File control information (FCI) record
- Directory (DIR) record
- Application signature (protected and confidential ALU only)
- Key Transformation Unit (KTU) (confidential ALU only)

From that there are three types of ALU:

Unprotected ALU: The ALU contains neither a signature nor is any of the segments encrypted. The unprotected KTU should only be used in a secure environment.

Protected ALU: The ALU contains an additional signature of the application. This signature is generated with the application providers private key. The public key is included in the ALC. The protected ALU can be checked against changes by the card and can therefore be used in an insecure

environment, if the application provider does not want to keep the code secret and if the data contains no secret data.
Confidential ALU: The ALU contains the application signature as the protected ALU. Additionally one or both of the code and data records are enciphered (using one or more DES or Triple DES Keys). The keys are stored in the KTU, which is enciphered with the card's public key. Thus confidential ALU are specific to one single card. The confidential ALU must be used if the data contains secret information after personalization and if it is sent over insecure networks.
Applications can be loaded in one of two modes, Standard or Shell. This gives application providers the flexibility to create MULTOS applications that require an ISO or non-ISO file structure. These are explained below:
1 Standard Mode. The MULTOS card allows one or more applications to be loaded in Standard Mode and this represents the "normal" mode of operation. The Master File (MF) is implicitly selected at reset and applications selected according to [7816-4].

2 Shell Mode. In Shell Mode, the shell application (not MULTOS) is responsible for processing the **Select File** command and the shell application is implicitly selected when the card is reset. Therefore, an application that requires a non-ISO file structure (which the application can control since it is responsible for processing all **Select File** commands) and/or must be selected when the MULTOS card is reset, should be loaded in Shell Mode. To load an application in Shell Mode, there must be no other applications loaded. Only one shell application is permitted. It is up to the application provider to determine which mode their application requires and to provide this information to the Card Issuer as part of the application registration process. See the MULTOS Developers Guide [MDG] for more details on shell and standard applications.

From Version 5 there will be a third mode, the default mode. The card behaves in default mode just as the standard mode, with the difference that the default application is selected after reset.

Application Load History

The application load history is a list of applications that have been loaded onto the MULTOS Card during its lifetime. Applications are only added to the list if the random seed within the Application Load Certificate is non-zero.

The application load history contains the application_id (application identifier) and a random seed value. The MULTOS Card will not allow an application to be loaded where the application_id and random seed value from the ALC already appear in the history. This has the effect of making certificates with a non-zero random seed single use per MULTOS Card. They may be used to load an application once but cannot be used to reload the application. If an Application Load Certificate with a non-zero random seed is used then the application may only be deleted using a delete certificate which has a random seed value equal to the random seed value used when loading the application.

Application Customization/Personalization

MULTOS does not provide a specific method to personalize the application, that is how to get the specific data of the cardholder (e.g. name, account number, specific keys ...) in the static memory space of the application. It is up to the application provider and the issuer to implement one of the 2 fundamentally different methods:

1. The personalization specific data is written in the data segment of the ALU prior to loading the application to the MULTOS Card. This must be done in a secure environment usually by the issuer. The ALU can then be encrypted and be sent over an insecure network to the load terminal. This load terminal needs no specific customization modules and no secure environment as the ALU will be decrypted on the MULTOS card.
2. The personalization specific data is written to the card using application specific commands after loading the application on the card. Usually, this will be done by the Card Bureau in a secure environment.

5 THE CARD MANAGEMENT SYSTEM

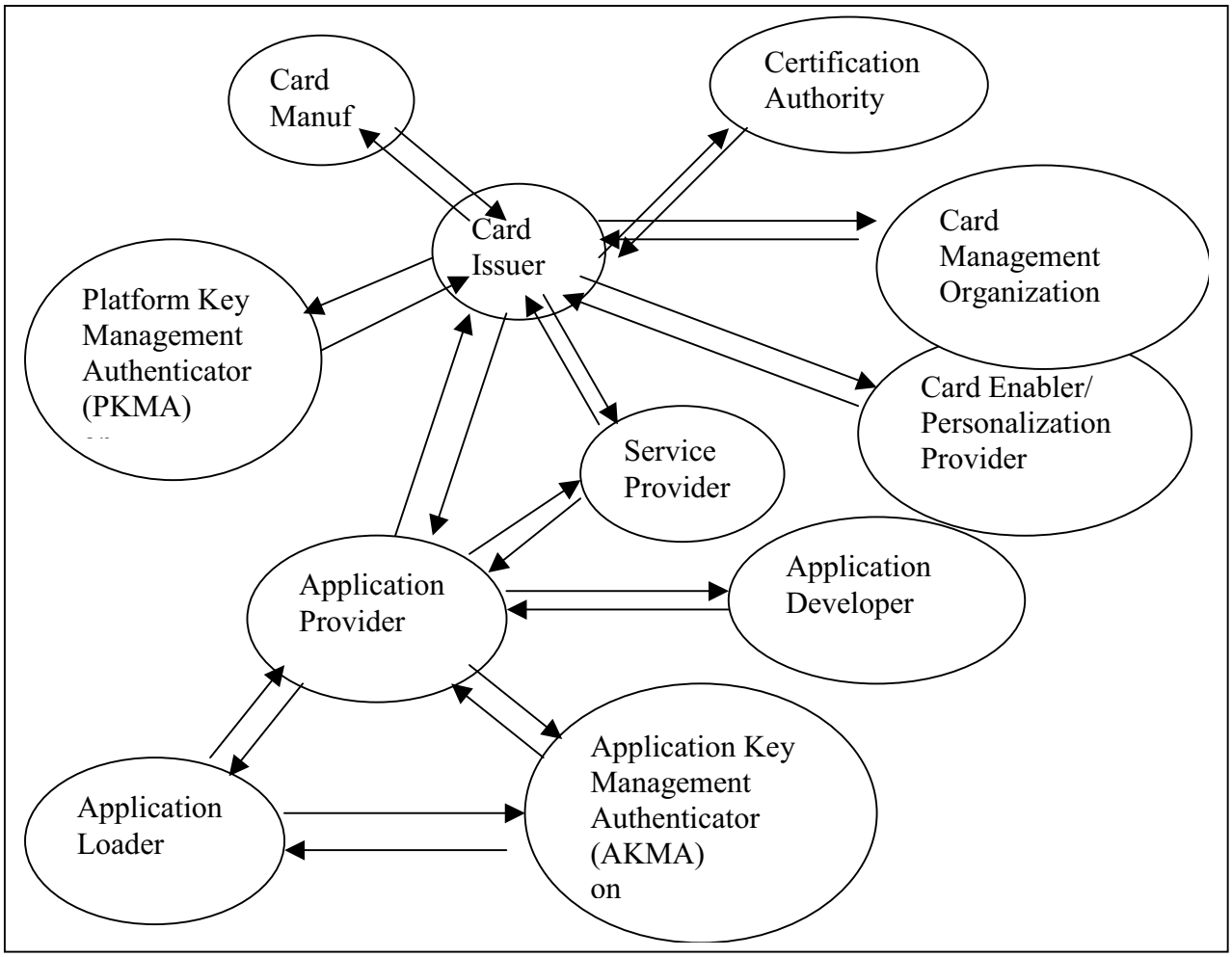
5.1 INTRODUCTION TO THE CMS

The CMS can be considered as the control panel of the Multi-application System. It contains all the necessary information about the status of each card (card life cycle status, applets status, memory available for applet downloading). The CMS communicates with the Terminals under their control to update, modify or delete the card's contents on end-user demand or because of security needs. At the same time, the CMS is the distribution channel that allows service providers to distribute their applications .

The functionalities of the Card Management System include:

- Application loading
- Card Production and Personalization
- Card and Application Maintenance
- Data Sharing
- Informative support for Card Renewal
- Certificate Status (recovery, expiry date, revocation)
- Blacklist management
- Billing, Reporting, Logging and Archiving

5.2 THE ROLES FOR THE CARD MANAGEMENT SYSTEM



The existing architectures for Card Management Systems (OP, MAOSCO, NICSS) have a similar functional architecture around the central role of the Card Issuer. The same organization can play different roles simultaneously. The de-centralized approach of the figure is consistent with the trend to improve the possible business case for the stakeholders.
This chapter will be completed in the next version of this document.

5.3 INTRODUCTION TO OPEN PLATFORM AND MAOSCO

Both architectures can be modeled as a flow of information/product interchange between 14 interrelated « roles ». A particular organization may play several roles within the MA scheme. A role supports a well-defined function within the architecture : eg Card Manufacturer or Card Issuer.
Some of these roles need to interact to perform well-defined functions within the scheme. This appendix roughly tries to identify some separated paths in order to get a better understanding of the MA architecture and to identify possible business agreements.

GlobalPlatform is an international, cross-industry forum, founded in September 1999 to focus on the development, management and promotion of specifications for multiple application smart cards, smart card applications, and enabling devices. With support from its global member organizations, which totalled 50 in March 2001, GlobalPlatform promotes a standard framework facilitating the implementation of smart card programs in any industry around the world. GlobalPlatform allows flexibility in the choice of technologies and vendors through an emphasis on open standards for cards, terminals and support infrastructure.

The Open Platform card and terminal specifications developed by Visa are the first open standards adopted by GlobalPlatform and will provide a solid foundation from which the organization will define the future of multiple application smart cards. The Visa Open Platform (VOP) specifications to fill the interoperability gaps in the Java Card Specifications, addressing terminal and card personalization issues. In October 99 Visa handed over the Open Platform (OP) specifications to the GlobalPlatform consortium in order to have OP endorsed
by other industry sectors outside the financial market.

5.4 OPEN PLATFORM OVERVIEW

The presentation of Open Platform is included here because it is often associated with the Java Card platform

Open Platform is based on the paradigm that there is one single Card Issuer for a card, but it is designed with the idea that the card issuer will have an ever-changing array of business partners who may want to share application space on the card Issuer's cards (The ultimate control always rests with the card issuer).

Practicalities for the Open Global Platform

Because the solution is compatible with existing Open Platform technology, backward compatibility with existing Open Platform cards can be ensured. A small application is added, pre-issuance at the personalisation stage, or even post-issuance in the field, to handle the Card Management Identification template. The card management system should be able to select such application and retrieve the relevant information. If an existing Open Platform card does not contain the Card Management Identification template, the card management system should be able to fall back to existing trial-and–error mechanisms – in other words try to select Card Manager and retrieve the (more limited) management information. With the issuance of Open Platform cards compliant to release 2.1 of the Open Platform Card specification, such a situation will no longer occur and smart cards will respond appropriately to card management system queries.

Global Platform Requirements
Will be completed in a next version of the document

5.5 THE MAOSCO PLATFORM OVERVIEW

The MULTi-application Operating System was originally developed by Mondex International, but is now controlled by MAOSCO.
MAOSCO is a consortium of thirteen of the world's most innovative smart card companies and was formed in 1997 to drive the adoption and manage the on-going development of MULTOS, the most highly secure open multi-application standard platform for smart cards. MULTOS consists of an operating system designed specifically for smart cards, an application development language, specific silicon chip families, and the supporting key management infrastructure. MULTOS is licensed on an open, non-discriminatory, royalty-free basis and is a standard for multi-application smartcards across all industry sectors. Third party software developers can develop MULTOS applications or custom design Value Added Services for the card issuer using both low level and high level languages and publicly available API's.

PRACTICALITIES FOR MAOSCO

The MULTOS release 5 has been targeted for inclusion in the common card management ID framework. A primitive in the OS responds to the card request identifying the card configuration. Backwards compatibility is not targeted for this feature as terminals will, in any case, need to support both mechanisms in situations where issued cards are part of the scheme. A non-compliant card would need to be identified and then an application loaded to support the framework for no practical net benefit.

The need for interoperability - for devices that coexist and that can collaborate where appropriate - is fundamental for mass adoption. The goal for Global Platform and MAOSCO is to remove the barriers to successful interoperability whilst maintaining the pace of innovation and diversity in services and products that the emergent smart card industry needs. Our member organisations will help to define those barriers and we will jointly marshall the technical resources necessary to remove them. We envision a convergent future where GlobalPlatform and MULTOS compliant cards deliver differentiated solutions within a common framework.

5.6 SIMILARITIES OF BOTH APPROACHES

A document has been recently draft with a comparative study of similarities and differences between both systems.

The "Card and Application Management Systems" (CAMS) document has been jointly developed by MasterCard/MXI/Visa to define a common terminology for systems and interfaces used in smart card systems, donated to Global Platform.

This document has been provided to the Secretary of the TB7 by MAOSCO, and is available on request.

The common framework developed by GlobalPlatform and MAOSCO defines a Card Management Identification mechanism that enables smart card systems to identify the on-card technology, whether it is MULTOS, Open Platform or any other standard. It allows a card management system, or any other system, to handle various smart card platform technologies, using common commands to automatically determine the card platform type, after which appropriate platform and management processes can be performed as required.

This Card Management Identification mechanism makes use of two ISO 7816-4 commands that can be issued to the smart card at any time. These commands give access to information in the card which ISO 7816-6 calls 'card data'. This data can provide a range of information, for example:

- The type of platform and its version, indicating to the card management system what management functions the card supports and how they can be invoked.
- The card identification scheme ensuring that the chip identifier can be made globally unique, so that the card management system can unambiguously identify the smart card.

Specifically, the two ISO 7816-4 commands are:

- A special form of the 'SELECT' (or 'SELECT FILE') command, providing unambiguous treatment of the second command.
- The 'GET DATA' command for retrieving the 'card data' information from the card.

'Card data' includes a template containing the following data elements:

- The Card Management Identification data object, containing an ISO object identifier, identifying the card platform organisation, the platform type, and its version.
- Optionally, the Card Identification Scheme data object, containing an ISO object identifier, identifying the card identification organisation, the identification scheme, and the unique smart card identifier.
- The ISO object identifier of the 'Tag Allocation Authority'. This is the organisation that assigns tag values to the various data elements of the template.

The template is ASN.1 BER encoded and uses a 'compatible tag allocation scheme', according to ISO 7816-6. With its tagging approach, this template can be easily extended to cover other platform configuration information that may be defined in the future. Using a tag allocation scheme approach, and object identifiers, allows the card management information to be universal, while the future addition of data elements and their associated tags remains possible.

By leveraging the widely used ISO 7816-4 commands and a minimum set of data, this framework causes minimal overheads in memory, input/output and processing to any existing and future smart card platform. It does not compromise privacy requirements, because it primarily provides platform technology and management information. It simply identifies the smart card, but not the individual.

Targeted for card and application management activities, this framework is generic and can be extended to smart card platforms other than Open Platform and Multos. It allows continuing platform differentiation and competition and has no impact on the applications themselves and their interaction with users: cardholders, merchants, or acquiring systems and devices. It is backward compatible with existing MULTOS and Open Platform cards and can be implemented today. The solution is integrated into release 5 of Multos specifications and release 2.1 of Open Platform card specifications.

5.7 CARD MANAGEMENT SYSTEM WITH PKI

Transactions require a certain level of security managed by the smart cards. The Card should interface to the BackOffice Systems in a secure/transparent way.

MA Card Issuers must improve their security posture by ensuring the integrity and confidentiality of their data, validating all users who wish to access data, and by providing a means for digital signatures that cannot be repudiated at a later date. For example, digital signature provides an audit trail which allows one to determine which user performed a specific action, and under whose authority that action was performed.
The security improvements realized through the use of digital signature provide Card Issuers and their stakeholders with greater confidence in the integrity of their systems and the accuracy of their data. Further, Card Issuers will be confident that their data is being used as intended. Without these improvements in security posture, Card Issuers/System Operators will not be able to become true competitors in the new e-commerce economy by participating in e-government initiatives. The use of PKI raises the issue of the legal recognition of the Digital Signature. Recently USA and European initiatives (European Directive on Electronic Signature) have made it possible to define under which

conditions a Digital Signature may have the same legal value as a hand-written one.
Even if this issue is out of the scope of TB7, this document will provide some information on this promising evolution in the USA, which could leverage the multi-application smart card market.

The use of PKI/smart cards is being prompted by recently enacted legislative, executive, and agency policies. For example, PKI certificates can be used to comply with the Government Paperwork Elimination Act (GPEA, Public Law 105-277). This act requires Federal agencies to accept electronic signatures, including digital signatures. Further, GPEA asserted that electronic signatures would not be denied legal validity simply because they are in electronic form (and this point was reinforced through the enactment of the Electronic Signatures in Global and National Commerce Act in June 2000, covering transactions between private parties, such as businesses and consumers). GPEA required agencies to submit plans to OMB by October 2000 as to how they would comply with the act. Using PKI is one way agencies can comply with GPEA. Subsequent presidential administration directives reinforced the need for acceptance of electronic forms with electronic signatures. OMB has issued guidance (Federal Register May 2000: Volume 65, Number 85, page 25508) for the use of electronic signatures to facilitate adoption by Federal agencies.

Electronic Certificates

Digital certificates provide a means for authenticating transacting parties over the Internet, and thus conducting business with confidence over the Internet. Public key technology enables digital signature functionality that provides authentication of electronic data for a wide variety of applications. The use of digital signature without public key technology may compromise authentication and lack nonrepudiation capability. Further, a single infrastructure provided by PKI supports both digital signatures and confidentiality (preferably using two different key pairs and certificates). As a result, many agencies are inclined to use public key technology as a solution. Although the deadline for compliance with the mandates of GPEA is three years away, the passing of the act and the OMB requirement for an implementation plan have encouraged Federal agencies to consider seriously digital signatures.

Furthermore, agencies must also comply with the provisions of Presidential Decision Directive (PDD) 63, which mandates that critical infrastructures of the United States must be protected against terrorist attack. Certain information systems are identified as a critical infrastructure, as is the continuity of government operations. Information assurance is often a key component of an agency's critical infrastructure plan. The encryption of data via PKI/smart cards is one way agencies can comply with PDD-63 requirements to protect critical infrastructure against terrorist attack.

Some agencies are also issuing guidance that is driving promulgation of PKI/smart cards. In the DoD, for example, a memo from Dr. John Hamre requires that PKI-enabled via a common access card be issued to all active duty military, reservists, and civilians and contractors employed at the DoD by the end of 2002.

Accomplish Mission

- Providing a means for completing forms over the Internet
- Ensuring the integrity of the data provided to customers
- Guaranteeing that confidential data will not be compromised
- Validating that users attempting to gain access to systems are authorized users.

Many agencies responsible for national security, including the DoD, can accomplish their mission more effectively by implementing technologies that will greatly enhance their security posture. For example, it is imperative for agencies with national security missions to guarantee the confidentiality of classified data, protect engineering secrets, and prohibit unauthorized users from gaining access to data. However, all agencies can meet mission objectives more effectively by providing a means to

complete transactions securely over the Internet, ensuring the integrity and confidentiality of the data, and properly authenticating all users.

The use of PKI technology facilitates “many to many” relationships. PKI enables the use of the same technology for a wide range of applications. PKI creates a trustworthy environment for e-commerce transactions and secure communications over the Internet for both individuals and organizations. The standards-based directory structure can grow as the user base grows.

Benefits of Utilizing Smart Cards

PKI certificates can be stored on smart card tokens. Smart cards have become widely accepted due to the high level of security the card provides compared with PKI certificates stored on a hard drive. Additionally, smart card applications are developed based on standards and using advanced and proven technology. Like PKI, smart card technology also offers significant benefits in its interoperability and scalability, but unlike PKI, also offers portability.

6 THE STIP ARCHITECTURE AND JEFF FILE FORMAT

6.1 INTRODUCTION

WP4 is intended to address Interoperability issues for the Card System. This task involves to precisely define the term « interoperability ». As outlined above, interoperability means different things from the point of view of each of the card holder, the application provider and the system operator. Roughly, interoperability from the card holder perspective is similar to access to services « anytime-anywhere ». For an application provider, interoperability matches with software portability. For the system operator, interoperability refers to sub-system functional interchangeability enabling integration of standard physical/logical devices.
Once an agreement on the concepts is reached, WP4 has to succeed in providing interoperable solutions both in terms of service and infrastructure. Several approaches can be taken.

NOTE: Even if Terminal and Card Accepting Devices are different concepts, in the following they will be used interchangeably.

Whereas the Terminal/Card Interface has been widely standardized in the past years, for both fixed and mobile terminals, very little has been done for the Terminal/Back Office interface. The main reason is that current operational networks remain largely closed.
Secure transactions in general and payment applications in particular are moving from the traditional POS environment to mobile phones, vending machines, PDA’s and other devices. These card accepting terminals (1) have to be interfaced to open networks (2) have limited hardware resources (3) become of generalized used. Therefore the need for interoperability becomes of first importance. The idea therefore, is to specify a software interface (API) which could be implementable in most of the Card Accepting Devices with the following properties :

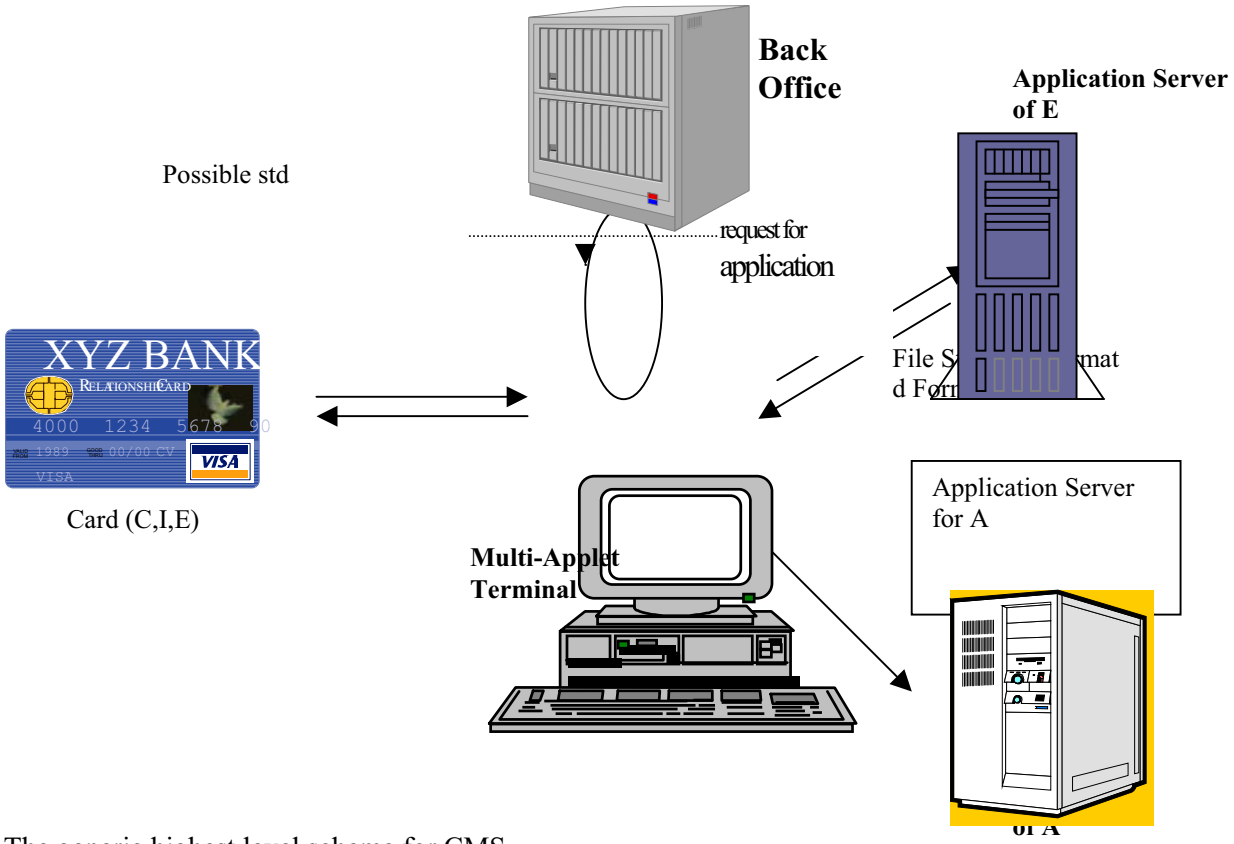
- Support multiple secure applications on the Terminal side
- Provide interoperability such that applications may run on a wide range of device types
- Provide Platform and Application life cycle management
- Can be implemented on small card accepting devices with limited resources

We are going to review in the following example, with a simple protocol, how these problems can be addressed. For simplicity purposes, the security aspects of the protocol have not been included (authentication of the partners and of the sensitive messages).
After that, the STIP Architecture is presented as a framework which allows support for this first model. The idea is that the STIP supporting the JEFF File Format may constitute a Generic Value Added Services Multi-application Platform that the TB7 is looking for.

6.2 GENERIC FUNCTIONAL MODEL FOR ANYTIME- ANYWHERE APPROACH

The Problem
Case 1 : Any CAD (Card Acceptance Device) is able to proceed to the capture and download of an Application A, not-resident in the card

Case 2 : The Card contains an application which is not resident in the Terminal. The terminal is able to download the Terminal side of the application to proceed to the execution of the application on legitimate card holder request



The generic highest level scheme for CMS

Case 1

1. The Card (C,I,E) is inserted into the Terminal
2. The Terminal proceeds to the activation sequence
3. The Card Holder wants to execute an application which is not available in the Card
4. The Terminal request its own Back-Office in order to download application A in the Card
5. The Terminal requests the Emitter E of the Card for authorization to download A in the Card (C,I,E).
6. The Terminal contacts the Server for A and downloads A
7. The Terminal downloads A into the Card and informs E
8. The Card and Terminal executes A
9. The Terminal contacts Back-Office of A to inform that the transaction has taken place with Card (C,I,E). Security Risks : The Terminal can invent false transactions with Card (C,I,E). To avoid this any transaction with a protected application must be signed and dated by the card itself. The Card Issuer trusts its card, eventually the AP does, but it is unclear whether AP and CI trust a terminal. B and E agree on a clearing scheme

Case 2

1. The Terminal does not host the application, the card holder wants to run it and the application A is available on the card
2. The Card tells the Terminal the URL where it can proceed to the application downloading
3. The Terminal requests authorization from its Back-Office to download application A
4. The Terminal requests URL (Server application A) to download At

5. The Card and Terminal executes A
6. The Terminal A informs Back-Office of A that the transaction has taken place with Card (C,I,E).

Case 3 : White Card directly issued by the Card Manufacturer

The Card contains a public certificate and an electronic signature function. This signature function is the only service provided by the card after card issuance

1. This certificate has been produced by a CA on request of the card manufacturer. The pair of keys has directly been generated by the card itself. The public key is certified whereas the private key never leaves the card. The pair of keys can be used on the restrictions declared by the certificate. The certificate indicates the profile of the card.
2. These keys+ certificate allows a secure channel to be set up between an application provider server and the card itself. This secure channel can allow later downloading of an applet on card holder request.
3. The Card produces a request for download applet message. This message is signed with the private key of the card. The certificate corresponding to the public key of the private key of signature is also transmitted.
4. This message is relayed by the terminal and transmitted to the server application
5. After signature verification, the server application proceeds to the applet transmission to the terminal, which relies on and proceeds
6. To the applet installation into the card.
7. Once the applet is installed and initiated in the card, card and terminal proceed to the application execution if requested by the card holder

9. The Terminal sends the transaction information to the Back-Office

NOTE: Furthermore the certificate hosted by the card may be used to execute a secure protocol over an Open Network, type SSL, TLS or WTLS. This certificate allows for strong non-repudiation as far as it proves that a secure signature creation device generates the signature.

6.3 THE STIP SPECIFICATION

As mentioned, the main problem in order to get global interoperability for the system comes from the lack of standardization at the terminal, both in terms of the distributed application and of the interconnection of the Terminal with the Back-Office system.
Somehow, STIP tries to reproduce the Java Card Scheme at Terminal Level : By defining a Virtual Machine to be integrated in all the STIP terminals, the portability of any application compliant with the STIP API is granted.
This way, any new application just needs to be certified as being compliant with the STIP by the System responsible with no need to re-write the code.

STIP Approach

The following scenario illustrates the vision of STIP:
Imagine four devices:

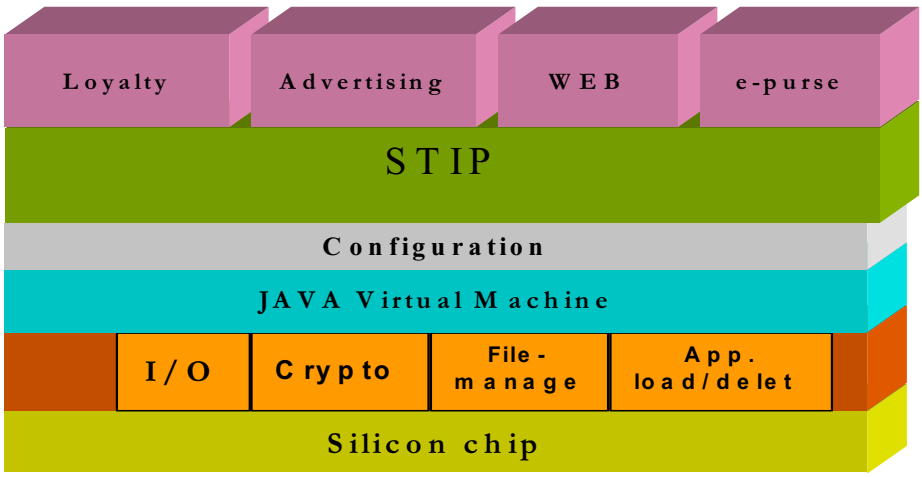
1.A vending machine
2.A low-end desktop POS terminal
3.A high-end hand-held portable terminal with large touch-screen and a display
4.PC (with attached peripherals as required)

And three STIP applications:

1.An e-purse application
2.A loyalty application
3.A security/ID application

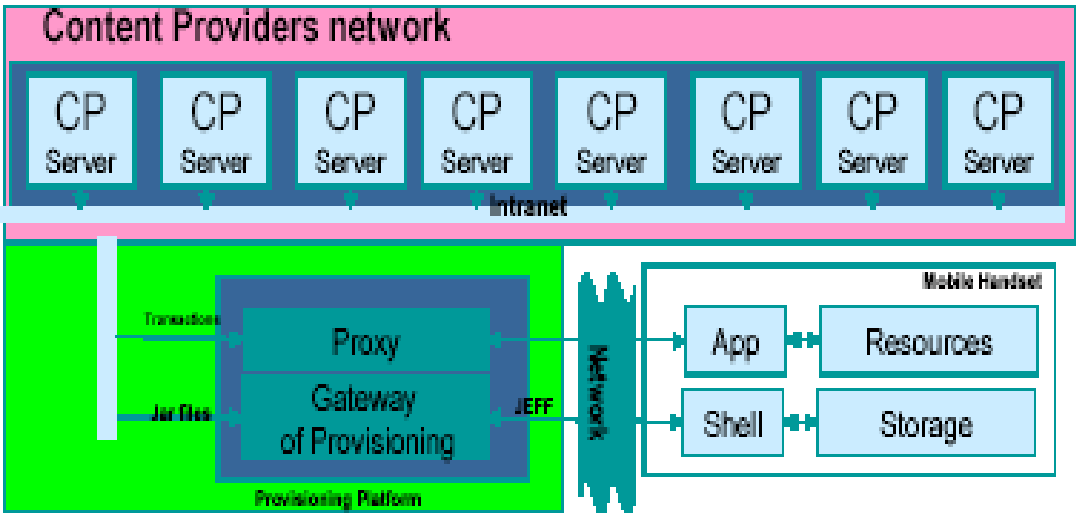
Also imagine the following:
1.Each application was written by a different developer and certified once by the organization responsible for the security of the overall card based system.

2.You are able to load and run any combination of the 3 applications on any of the 4 devices without modification or re-certification. You are able to do this using the existing download system for the particular device, the desktop and portable terminals have legacy credit/debit applications already installed which continue to run the applications automatically adapted to the different display, keypad, and printer configurations.
3.Further, the applications only provide the functionality appropriate for the type of device. For example, in a vending machine, the e-purse application might only do a simple debit based on the cost of the item dispensed, whereas the application may provide re-value on a desktop terminal.



6.4 STIP ARCHITECTURE FOR DATA PROVISIONING

This figure gives an overview of the components of the STIP generic Architecture, intended for the provision of services to a STIP compliant terminal through an External Network (Wired or Wireless).
The STIP architecture consists of the Service Provider Infrastructure (Under the Control of the Service Provider § 6.4.1) , the Network (managed by the Network Operator) and the STIP Terminal (§ 6.4.2). The Service Provider and the Network Operators roles may be ensured by the same organization. The Content Provider (CP)



6.4.1 The Service Provider Infrastructure

A Device (eg Mobile Handset) interconnects with the Service Provider Infrastructure (SPI) through a Network (Public or Private).

The Service Provider Infrastructure (SPI) supports the **Provisioning of Services Platform** (STIPlet download) as well as the **Administrative Services Platform (ASP)** (billing relation between the SP and the NO to manage the Content Provider and Subscriber accounts as well as the billing subsystem.)

The SPI communicates with the Network through an interface **The Provisioning of Services Platform (PSP),** which consist of the **Access Interface** to the PSP and a **Back-Office** Infrastructure interconnected through an IP Backbone.

The Access Interface is made up of the **Proxy Server** and the **Gateway of Provisioning**

The Proxy is a firewall between the *Intranet and the Extranet*

The Gateway ensures the content provisioning for the Terminal, the Smart Card or both of them. Its tasks include:

- The secure downloading of Data
- The uploading of the device profile
- The checking of the device identity
- The checking the service/content provider identity
- The translation of the JAR file into the JEFF format
- The encoding of binary data (MP3, icons) for heterogeneous loading
- The bytecode verificatioon of the STIPlets
- The signing of the JEFF file

A Gateway architecture is built around a common bus to which are connected different subsystems cooperating in order to download a STIPlet into a JEFF file to a STIP Terminal.

These Susbsystems provide the Gateway with the required services to secure the download operation:

- PKI in charge of signing the downloaded files and to Authenticate the *External* Devices (Terminals, Network, Content Provider Server) requesting access to the Gateway Services.
- Certification Services for the Digital Content Multimedia, Data , Card Application) provided by the Content Providers.
- Synchronisation between the Terminal Data Base and the Internal Server Data Base.

- File System Servers storing the software modules required to install the required application into the STIP terminal.
- Digital Right Management services when the Content Provider requires the secured distribution and /or Storage though the external network to the STIP terminal.

A Gateway architecture is built around a Provisioning Processor (PP), Expert System which manages the architecture and initiates the synchronization phase of the provisioning. The PP actions are detailed in § 6.4.3.

Gateway Compatibility with PKI

As mentioned, the gateway can use a PKI Infrastructure to provide Digital Signature and Certification Verification purposes. The Authentication of the negotiated parties and of the Software to be downloaded is a requirement of most of the cryptoprotocol algorithms over the Net. This authentication is most used to negotiate a common session transport key for confidentiality of either very sensitive information (Credit Card) or to protect Multimedia (MP3) information against Piracy.

6.4.2 The Device (ie Mobile Handset)

The device (ie a Portable Handset Terminal, like a PDA or a Mobile Set),is made up of a **Shell** implementing the User Interface, the **Applications** contained in the STIPlets and the **Device Resources** managing the control of the operation by the end user as well as the execution environment for the management of the applications (downloading, execution and deletion).

The Shell , provides a homogeneous Human Machine Interface for all devices managed by the Network Operator, their main tasks are:

- Allows end-user navigation within a Service Web Portal
- Selection of a Service Provider
- Purchase and download the associated JEFF file with the content
- Store the JEFF file locally
- Install the service or content in the terminal and register the new service
- Register the new service. The Shell makes the device aware of the new application.

The Shell is a STIP downloadable application and deals with the browsing of service within a private network of Content Provider and the Service Life Cycle Management (downloading, registration, initialization, installation).

An advantage of this scheme relies on the Homogeneous Human Interface available for all devices managed by the network operator

As mentioned the Applications in the STIP architecture are contained in the STIPlet.

6.4.3 The Synchronisation during the Provisioning of Services

The Synchronisation using a standard protocol like SYNCml provides a fine solution for the management and upgrade of the software resident in the STIP Terminal.

The objective of the Synchronisation is the consistency between the current configuration of the STIP Terminal and the Configuration required for the provisioning and installation of a new application in the Terminal. These process can be summarizes as follows :

1. The Terminal Request for the provisioning of a new application (STIPlet)
2. The Gateway retrieves the Terminal Configuration, after Terminal Authentication and sends it to the Provisioning Processor (PP)

3. The PP proceeds to the analysis of the differences, if any, between the current configuration of the Terminal and the required one for the new application. The identified gap (Software Modules) is transmitted to the Gateway Server Database which keeps track of the current configuration of the Customer Park of Terminals. These Software Modules shall be downloaded to the Terminal along with the Application . They are available in the File Server of the Gateway
4. Once the Server Database is upgraded the Synchronisation process starts, between the Server Database and the Requester Terminal using SYNCml. When the Synchronisation process is over, the Terminal is aware of the required files for the new application.
5. The Terminal proceeds to the download of these files using a negotiated protocol with the Gateway (FTP, HTTP ..)

In this Scheme, the SYNCml technology provides several specifications to:

Initiate the Communication between a Server (in the Gateway) and a Client (The Terminal. SYNCml Device Management Bootstrap)
Manage the Device (see SYNCml Device Management Protocol)
Define and Manage the representation of standard objects in the device (SYNCml Device Management Standardised Objects)
Ensure the security of the Synchronisation Process (SYNCml Device Management Security)

Along with the STIP platform, it is necessary to specify a file standard format in order to transport heterogeneous data from the distant Servers to the Stip Compliant Terminals. **STIP recommends the use of JEFF like container to distribute the STIPlets and binary data (icons, pictures, MP3, …).**

6.5 THE JEFF FILE FORMAT

JEFF is a standard format defined by ISO 20970, able to be managed by any platform based on an Object Oriented Language (Java, Brew, C…)

JEFF has been specified by the J-Consortium (www.j-consortium.org). JEFF is a byte code format which is well suited for the next version of Java Cards, i.e. 3.0.

This format is designed to download and store classes on a platform. JEFF is a translation of the Java class file format but which is not still ready for execution by the VM. The goal of JEFF is to provide a ready-for-execution format allowing Java programs to be executed directly from static memory, thus avoiding the necessity to recopy classes into dynamic runtime memory for execution. A virtual machine can use it efficiently

A JEFF file can contain several classes from several packages. The content can be the complete application , parts of it, or just one class. The binary content of a JEFF file is adapted to be efficiently read by a wide range of processors, and, in particular with STIP compliant Terminals.

6.6 STANDARDIZATION FOR THE PLATFORM STIP+JEFF

The TB7 is intended to support Technical Solutions which are:

1.Open, not proprietary

2.Supported by existing standards as far as possible

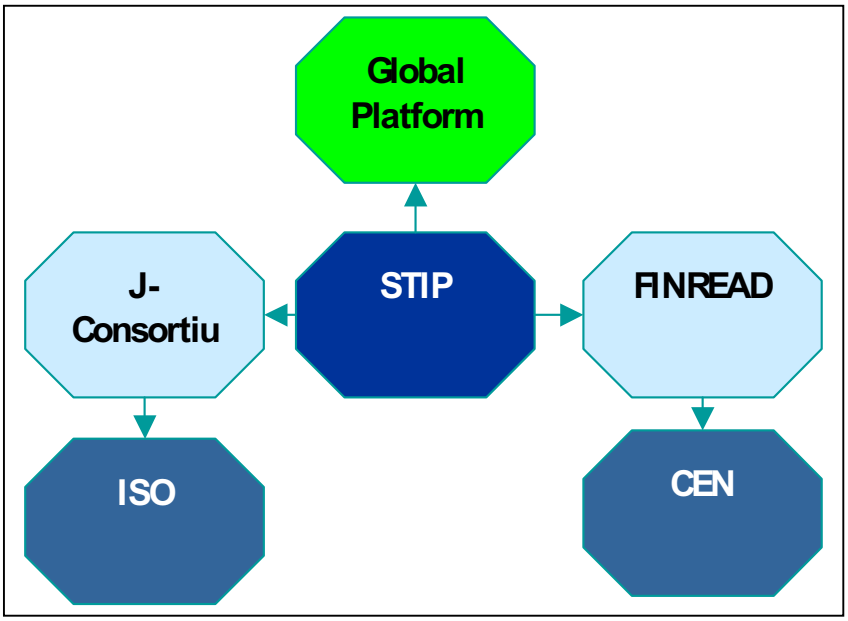
3.Compatible with solutions proposed by other TB's

The solution proposed in this Draft largely complies with the above requirements:

- STIP is an Open Industrial Standard jointly developed by major terminal manufacturers.

- Their specifications are freely available
- STIP is to be endorsed by Open Terminal Platform and become FINREAD compatible (TB4 compliant)
- STIP is a Java Distributed Architecture in line with other Industrial Sectorial Specifications (DVB-MHP, Java TV)
- JEFF has been posted in fast track ISO in March 2001. JEFF was in DIS in September 2001(JTC1/SC22) and has just been officially approved as a new ISO standards: ISO 20970. This might prove to be a milestone event in smart card history. For the first time a Java related standard has been recognized by ISO.

The above scheme shows the core position of STIP in relation to other major standardization initiatives:



7 IMDES, INTEROPERABLE MULTI-APPLICATION DATA EXPLOITATION SYSTEM

7.1 When, How and Why does it makes sense to federate proprietary back-office systems to a common Data warehouse system?

A careful backup strategy may be required to cater for multi-application card loss. Guarantee must be provided that any central database information is kept in step with the data in the card.
When launching a MA scheme, a decision to be made is (1) whether or not to place a central system in the infrastructure (2) If it is allowed to read sensitive information on the card for back-up purposes and (3) If it is allowed to store security sensitive data, eventually impacting the individuals privacy rights.

In addition, a satisfactory agreement between the parties about card loss and card withdrawal must be reached. In the case of multifunction cards, this may involve reloading the card with back-up data possibly coming from different sources. From the cardholder side, it would be interesting to do this process with a single operation (WP2, User Interface Requirement).

At the end, the issue that the IMDES addresses is a more fundamental one:
Should the multi-application card rather federate or making the back-offices independent?

From the cardholder point of view, the main advantage in using multi-application smart cards is ergonomic: the card allows access to different services possibly operated by different providers, which in principle are independent of each other. The card federated what was originally and remains independent. At first sight, the MA card does not make necessary the connection of back-offices.

However some business, technical and security reasons may lead to the interconnection of independent back-office systems to a common data capture and storage system that we generically refer as « data warehousing »

7.2 A proposal for the Data Warehouse Concept

The definition we suggest is that a data warehouse is a collection of data which is :

- Individual-oriented
- Integrated
- Time-variant
- Non-Volatile
- Protected against unauthorized access

These data are organized in databases built using relational technology and hosted in servers under the control of a partner of the Multi Application scheme.
In our context, these data are related to the use by the cardholders of the services provided by their personalized multi-application card. The Data warehouse support multiple data sources corresponding to the data captured by the terminals once a transaction with a card has taken place.

7.3 Exploitation of the Data Warehouse

An important aspect of the data warehouse context is the metadata. Simply stated, metadata is data about the data. Metadata keep track of what is where and acts somewhere as an operational system of the database. In principle, metadata specification is out of scope of the TB7.

In the multi-applicative context Metadata acts as an information repository of the Data mart. Data mart refers to data specific to a particular application. Data mart is to be directly exploited by the application provider. A summary of the Data mart can then be sent to the central Data warehouse for Consolidation/Clearing purposes.
To support the MA system, the Data Warehouse provides :

1. Immediate access to a large number of financial and non-financial information linked to the scheme operation, including real-time operations (excepted for off-line pure transactions like e-purse debit)
2. Automated transmittal of data into financial and information systems of the scheme partners so that the data associated with the transactions, authorizations and denials related to their respective applications are transmitted and recorded
3. Automated reconciliation of data transactions, which enables streamlined, timely financial settlement, reconciliation and generation of activity statements and eventually of real-time audit trails

7.4 Multi-application Card decentralized approach

But for the system operator, the main reason for the multi-application smart card to be developed is to lower the access cost to a card service. In principle, if no dynamic download of new applications is offered, the multi-application scheme does not require to the modification of any existing « mono-application »-oriented system. Such a card simply allows the holder to have access to a number of different proprietary systems which are not required to communicate between them. Because the card federates the access possibilities, the terminal and back-office subsystems may remain independent. From this point of view, the multi-application card seems to support a decentralized approach of the informatics.

The most obvious response is that it depends on the relationships set up between the card issuer and the card scheme participants.
In principle an application provider only has to establish a one-to-one relationship with the card issuer.

But the Application Provider must know who are the other participants in the scheme : He may dislike sharing resources with a competitor for security and/or branding reasons. Even more, he may well require that the card issuer cannot download anything in the card that they consider bad for their image. Once again the problem of the need for a centralized authority comes up.

This means that the AP trusts the card issuer : It guarantees that the card OS firewalls the applet and that the Platform Security Policy is compatible with its own AP security policy.

Case 1 : The Card Issuer requires the AP to put the application data available in a common datawarehouse.
The Card Issuer is liable to the card user. If the Card is lost or stolen , the Card Issuer may decide to be able to restore the existing data in a new card. This requires the use of a common back-ground interface.

Case 2 : The OS is under total control of the card issuer. How the Card Issuer can guarantee an AP that itself is not going to proceed to modify some application data**.** *Rule 1 :The AP cannot distrust the Card Issuer*

7.5 Privacy Concerns : Data Warehousing Legal Issues

Is the consumer aware that he is multi/cross profiled ?
Is the retail store aware that he is also profiled and that his customer file is stored outside his company (Internet bannering)?

Caution: will the data producers keep benefits from the data they contribute to generate? all/some/none…or disadvantages ?

Incorporate access right as a functionality of the multi-application smartcard

- E.g : what and where are my personal data
- Who is using or allowed my data and why
- Allow the data subject to defend himself or to negotiate

The balance of these requirements may raise complex management issues with multi-applicative cards which makes the generation of positive business cases difficult.

7.6 Requirements for IMDES Service Providers

Keeping track of system transactions requires the careful design and secure operation of a Data Warehouse sub-system. The responsible organization must meet the privacy and consumer legislation and national (when available) principles for fair handling of personal information.
These legal principles may include some kind of accreditation for those companies wishing to play this role. This accreditation certifies that basic codes of practice are respected and is intended to raise the business objective of achieving trust and confidence.
The requirements shall typically include the nature of the data to be processed, as well as organizational and managerial procedures. They will clearly be specified in the contract linking the card issuer with the IMDES service provider. The identification of the sensitive information to be manipulated is under the responsibility of the card issuer. The card issuer shall clearly specify what we generically can call Information Privacy Principles including :

- Collection-what data may be collected about individuals
- Notification-who should be notified of any collected data
- Storage-under what conditions collected data must be stored
- Access and Correction-the limits to use and disclosure of stored data
- Access conditions for the Application Providers-ie AP shall only have access to information on the card necessary to carry out transactions with this AP

Based on this initial analysis the card issuer shall proceed to the selection of the security policy applicable to the data related to the exploitation of the system.

7.7 Protection of Digital Assets

- Confidentiality of sensitive information : Encryption procedures and limitation of the transmission of confidential information
- Integrity of valuable information
- Prevention of unauthorised copying and use
- Availability of Critical Information
- Management of risks to critical information

7.8 Dependability of Services and Systems

- Availability, reliability and integrity of the infrastructure
- Prevention of unauthorised use of infrastructure
- Guaranteed level of services
- Management of risks to critical infrastructure

7.9 Example of standard requirements for IMDES/ Cardholder contractual terms

1. IMDES service provider shall implement procedures to ensure that the information in the storage system is accurate, current and the minimum necessary for the purpose.

2. When no longer required, information shall be purged from the card and associated systems.

3. The contractor shall ensure that information given for one purpose is not used for any other purpose, or passed to any third party without the subject's informed consent.

4. The contractor shall ensure that the cardholder is informed of what information is being held, the purpose for which it is being held and when it is being passed on.

5. More specifically about Cardholder rights to access and verification of the information :

6. The contractor shall provide for cardholders to obtain access to all information held about them on the card and any associated system, in accordance with the requirements of data protection legislation at the time of the request.

The cardholder shall not have to make multiple requests to multiple service providers and pay multiple fees to access the totality of the information.

APPENDIX I: GUILLOU-QUISCATER 2 AUTHENTICATION PROTOCOL

Introduction

The MAS prerequisites Deliverable (D2) strongly relies on Device Authentication to secure applet management during the Multi-application system operation. Mutual Authentication between Card and Terminal is then required for off-line operation. To reinforce the system security, this Authentication should be dynamic. The Claimant must proof its knowledge of a key, by being able to calculate a cryptogram by using this key to cipher a challenge (a random number that the Claimant cannot anticipate) which has been generated and sent to the Claimant by the Verifier.
D2 relies on RSA for electronic signature purposes. RSA can also be used for Dynamic Authentication.
Both the RSA and the Zero-Knowledge (ZK) based cryptoprotocols are asymmetric: the Verifier knows a Public Key corresponding to a Private Key which is a Claimant Secret.
In RSA, the Claimant produces RSA signatures using its Private Key as a response to a challenge sent by the Verifier. The RSA Authentication is therefore a Challenge/Response algorithme. The ZK protocols is a Commitment/Challenge/Response one. This fact involves that both the execution sequence of dynamic authentication schemes, and the workload (in terms of calculation) are different. The next paragraphs clarify and compare both Dynamic Authentication Algorithmes.

Asymmetric Authentication based on integer factorization

The integer factorization problem:

A public modulus n is the product of secret large primes, at least two of them being distinct. As the modulus n is the product of k f primes: p_1 to p_K , the ring of the integers modulo n is composed of k Gallois fields GF (p_1) and GF (p_K).

Workload for Authentication based on integer factorization

The following apply:

1. For multiplication and square modulo n, multiplying the modulus length by m, multiplies the workload by a factor m^2. It is therefore interesting to perform multiplication and square modulo in a ring of the integers of value less than n (n/2, n/3, n/4 …), because it allows a reduction in the number of multiplications of modulus n of respectively (1/4, 1/9, 1/16…) in relation with working into the ring of the integers modulo n.

2. A square modulo n represents approximately ¾ of the workload of a multiplication modulus n.
3. A multiplication and a reduction modulus n represent approximately the same workload

With these assumptions, the workload unit is the multiplication modulo n and depends on the modulus size.

Dynamic Authentication using RSA

The RSA Algorithm is based on the Secret Factorization of a Public Modulus.

The RSA scheme uses two large primes p_1 and p_2 and a public exponent e that must be co-prime with ($p_1 – 1$) and ($p_2 – 1$). The ring of the integers modulo n is the product of two Galois fields GF (p1) and GF (p_2).

The private exponent d is the inverse of e modulo the maximum order modulo n, ie, the Carmichael function λ (n) of modulus n, denoted lcm ($p_1 –1$, $p_2 –1$).

$n = p_1 \times p_2$ dxe≡ 1 modulo ½ lcm $(p_1 - 1, p_2 - 1)$

The Public Key is denoted as ⟨e,n⟩ , meaning "e-th power modulo n", and permutes 1 (RSA permutation) each Galois Field GF (p) and therefore the ring of the integers modulo n (numbers 0 to n-1).
The Private Key is denoted as ⟨d,n⟩ , meaning "d-th power modulo n", an inverts the RSA permutation.

Workload Estimation for Dynamic Authentication using RSA

For the Claimant

With this mechanism the RSA Verifier Workload depends on the public exponent e, whereas the RSA claimant workload depends on the number of prime factors and on the use of the Chinese Remainder Technique (CRT). The RSA Claimant produces RSA Signatures. The straightforward computation $\Sigma \equiv X^S \pmod n$ consist of $\log_2 n$ squares modulo n plus an average of (1/2) $\log_2 n$ multiplications module n (for multiplications and squares modulo n multiplying the modulus length by x multiplies the workload). As it knows the prime factors, the RSA claimant may dramatically reduce the workload by signing in mod p_1 , the mod p_2 , finally the signature according the CRT. The workload in this case (n as a product of just 2 prime numbers) is divided by 3.95, according to the rules previously defined. Therefore the RSA Claimant workload depends on the number of prime factors and on the use of the Chinese Remainder Technique (CRT).

For the Verifier

It verifies the Claimant's Signature with the Claimant's Public Key retrieved from the Claimant's Certificate after Certificate Verification. The RSA claimant workload depends on the public exponent e. The numbers 3, 17 and 65537 offer practical advantages: With two prime numbers the number of modular multiplications is limited respectively to 1,75, 3 and 13.

The Zero-Knowledge Algorithms (ZK)

The ZK algorithms are based on the following:
A Client (The Claimer herein) wish to proof a Service Provider (Verifier herein) that he holds a secret without revealing it. At the end of the execution of the ZK protocol, the Verifier knows nothing about the Secret of the Claimant except that the Claimant knows it.

Many ZK involve a triplet denoted Commitment, Challenge and Response represented as { Cmt, Ch, Rsp }. Both the Claimant and the Verifier calculate separately a ZK-triplet.
The *Claimant ZK-triplet is the Private triplet* because it is calculated from the Claimant Secret (one or several private numbers)
The *Verifier ZK-triplet is the Public triplet* calculated using only Public Numbers.
But when a ZK-triplet is captured at the interface there is no way to know whether it's Public or Private. However, the chronology for both triplets is different:

The Claimant first calculates the **Commitment (Private Commitment)** and anticipates a Response to any Challenge than can be transmitted form the Verifier side.

The Verifier calculates first **a** Response (the Verifier cannot anticipate **the** Response, because it knows nothing about the Secret using to calculate the Response by the Claimant to its challenge) **then a Commitment (Public Commitment)**

The **Challenge** is a string of kxm bits (k bits per basic number) organized in m numbers denoted Ch_{i+} to Ch_m. Ch_i is a k-bit number, from the most significant bit $Ch_{i,,1}$ to the least significant bit $Ch_{i,,K}$.

A verification formula converts any challenge-response pair into a reconstructed commitment that must be non-zero and identical to the received commitment.

Step1: A "commitment" from the Claimant to the Verifier
Step2: A "challenge" from the Verifier to the Claimant
Step 3: A "response" from the Claimant to the Verifier

The Guillou-Quiscater 2 (GQ2)

For GQ2 is based on a ZK proof of knowledge of a modulus decomposition held by a Claimant. This means that:

(1) the Claimant knows the Decomposition of the Modulus and needs to convince the Verifier without disclosing the specific Decomposition.
(2) The Verifier checks that the Decomposition is present without knowing it (the Claimant reveals nothing)

GQ2 allows for mutual authentication between two entities, the Claimant and the Verifier.

The consideration here of the GQ2 is justified by the cross-sectorial approach of TB7/WG4. As mentioned, GQ2 is very fast. Public Transport Multi-application cards supporting an ISO standards Contactless Interface (Transaction time limited to 200 ms) may take advantage of GQ2.

Concerning the coexistence of RSA and GQ2 it should be noted that any modulus n used for RSA signature with v=3 or $v=2^{16} + 1$ when a digital signature is needed by the application should be used in GQ2 mode with $v=2^K$ for current authentication.

Workload Estimation for GQ2

As mentioned the number of modular multiplications increases as the square of the modulus length n. The RSA claimant workload increases with the cube of the modulus length. But for the RSA verifier and for both the GQ2 claimant and verifier, the workload only increases as the square of the modulus length. As a consequence:

In the RSA the Claimant Workload is a factor of twice larger than the Verifier Workloasd following the modulus length n (EX: With n=1024, with two prime numbers, the Claimant proceeds to 324 multiplication modules whilst the Verifier only performs 1,75, 4 or 13 following e.

In GQ2 both the Claimant and Verifier have a balanced Workload which is equivalent to the Workload of the Verifier using RSA.

The Dynamic Authentication process with GQ2 is therefore faster than the RSA one.

Main Advantages of the Guillou-Quiscater 2 for Dynamic Authentication

1. Efficient Authentication Algorithm
2. Supported by any computer with cryptographic resources including the simplest one (smart card) with limited memory and processing capabilities
3. Compatibility with RSA based signature scheme
4. Under ISO Standardization Process: ISO/IEC 9798-5, **Entity Authentication, Mechanisms using zero-knowledge techniques** is presently under revision and the new version includes GQ2.

The main problem for GQ2 is that EMV Specification only supports Dynamic Data Authentication (DDA) using RSA. TB7/WG4 believes that future release of EMV Specification might accommodate GQ2.

APPENDIX 2: Contribution of the CEN/ISSS eURI Workshop

Aim and scope of the Workshop

The URI specification was initially developed as CEN Workshop Agreement CWA 13987 : 2000.

URI stands for User Related Information, in this case stored on a smart card. The original URI project took as the scope of the data:

- such information as the user might otherwise be expected to type in at the terminal when using ICT services – typically this includes name, address, date of birth, name and/or address of the service that the user wishes to select (but *not* credit/debit card information)
- configuration information for the terminal, to be used by the terminal to adapt its user interface for users with special needs, such as requirements for large print on the display or for a longer period before the terminal times out when the user is requested to respond to a prompt
- optional additional information to be defined by the card scheme or by more than one card scheme acting in partnership

The terminal should read at least the configuration information as soon as the card is initialised, and immediately act on it.

Implementation of the URI, defined as an interface specification between card and terminal ('card edge'), was not part of the earlier CWA, and the Extended URI Workshop is developing the interface specification for contact cards conforming to ISO/IEC 7816 or for contactless cards conforming to ISO/IEC 14443 part 4, for use in a multi-application environment. The full title of the extended URI CWA (known as eURI, pronounced 'e U R I') is 'User Related Information Extended to Multi-application Smart Cards'. The CWA includes the card edge specification for the dataset, an access method, security methodology, and implementation information and guidelines.

The security level required for the URI is low, as release of the information is always controlled by the card holder (user). Thus the URI dataset should only be used to hold information which can be viewed by the user in a form which is readily understood.

The CWA consists of three Parts:

- CWA 13987-1 Smart Card Systems: Interoperable Citizen Services: User Related Information: Definition of User Related Information and Implementation
- CWA 13987-2 Smart Card Systems: Interoperable Citizen Services: User Related Information: Implementation Guidelines
- CWA 13987-3 Smart Card Systems: Interoperable Citizen Services: User Related Information: Guidelines to Creating, Operating and Maintaining an Interoperable Network

The specifications given in Part 1 are prescriptive, whereas Parts 2 and 3 are informative.

Publication of the new CWA is planned for early 2003.

Mandatory and optional data fields

The eURI definition contained in Part 1 of the CWA 13987 specification includes both mandatory and optional items. The mandatory *data* is the minimum set that is required properly to support required eURI functionality. However, many implementations produced by eURI Issuers, with Card Holder approval where necessary, will include optional data. Such optional data will extend the usefulness of the eURI in providing inclusive operation and in making available a wider range of candidate interoperable applications at the terminal.

Implementation method

The proposed implementation is as an application or applet in the smart card, with an Application Identifer (AID) as defined in ISO/IEC 7816, and selected by means of a 'SELECT' APDU using the AID as parameter. There will be one universal eURI AID, registered with ISO.

Data elements are encoded into data objects using TLV format (Tag, Length, Value), and the resulting data objects are combined into BER-TLV constructed data objects using tags and formats compliant with ISO/IEC 7816.

Where possible, the eURI uses existing standards to define the data elements, particularly EN 1332-4 for the special needs parameters to configure terminals.

In addition:

- a format is proposed for transfer of data elements from the eURI across a network, using 7816-compliant methods for carrying identification of the source of the data element definition along with the data itself;
- the dataset specified in CWA 13987 : 2000 is being updated; and
- support for legacy datasets (CWA 13987 : 2000 and DISTINCT ID) continues until at least 2005.

Using the eURI to build interoperable networks

The case for interoperable networks

Smart card interoperability is a key requirement of the developing Information Society, especially as citizen mobility around Europe increases. This requirement is strongly supported by two factors:

- The growing use of smart cards as enabling tokens for products and services emanating from the Information Society, i.e. products and services based on Information and Communications Technologies (ICT)
- The usefulness of smart cards in the creation of an inclusive society by informing ICT applications and services of the requirements profile of the Card Holder.
-

An organisation wishing to deliver services by means of a smart card system may choose to invest in the issuing of its own smart cards and in the installation of its own equipment and provider network, or it may choose to share resources with other organisations. The effective and efficient sharing of resources among different organisations for service provision to the end user is what is intended in the eURI CWA by the term interoperability. The resources shared in an interoperable network using smart cards may include:

- the cards
- terminals supporting card readers
- data networks and network nodes used by Service Providers
- a Service Provider's customer base
- data and information.

The collective use of these resources for improving service delivery requires collaboration between Service Providers, Card Issuers, Card Acceptors and often also administrations or other regulatory bodies. Collaboration of this kind may be sought because the services offered by different providers are complementary or, indeed, they may be in direct competition. In any case, the benefits of participating in an interoperable network of this kind are considerable.

Actors

Deployment of the eURI is expected to benefit the various actors in the ICT service delivery value chain. The main actors identified in Part 3 of the CWA are:

- Card Holder
- Service Provider (both on and off card, including Application Loaders)
- Card Acceptor
- Card Issuer

- Certification and Registration Authorities
- Card Community Managers
- Regulatory Bodies

Card Communities

CWA 13987 provides specifications and guidelines that can enable the establishment of an interoperable network of actors. These organisations may form Card Communities (CCs) in order to manage the eURI, provide authority for loading it onto a smart card, and controlling its format and contents.

The nature of a Card Community and its role are discussed in the informative parts (Parts 2 and 3) of the CWA specification. It is recognised that several Card Communities may exist side by side covering different geographical or other jurisdictional areas. It is also noted that Card Communities may co-operate with one another to widen yet further the scope of service interoperability.

Levels of deployment

The full benefits of eURI deployment will be achieved through the delivery of ICT services within an interoperable environment of ICT actors (as listed in section 0.4). The eURI specification is a key enabler for the establishment of this environment. However, it is recognised:

i) that eURI implementations may be deployed at different levels, depending on the agreements made between members of the interoperable network and the technical restrictions of system terminals and

ii) that other Card Community schemes other that those described in Part 3 of the CWA (an informative part) may be set up and that these Card Communities could nonetheless implement and make use of the eURI.

CWA 13987 provides guidelines for deployment up to the maximum level of interoperability, i.e. for service delivery involving on-line connection with a remote application. In practice, the following limitations to deployment at this level may apply:

i) a Card Acceptor may choose to implement the eURI with restricted functionality, i.e. without providing access to on-line services

ii) the system terminal accepting the smart card may be restricted in its ability to adapt the human machine interface

iii) the system terminal accepting the smart card may not allow an on-line connection to a remote application

iv) the system terminal accepting the smart card may only be capable of offering a single service.

Where these restrictions apply, it is clear that many of the benefits of eURI, and the interoperable environment it supports, cannot be achieved. However, it is likely that some organisations may opt for phased deployment of the eURI, starting with a system offering restricted functionality and then gradually introducing additional on-line services at its terminals. Such organisations (who are likely to be both Card Issuers and Card Acceptors) are encouraged to use the CWA 13987 specification, even though their current terminals offer restricted functionality. Such organisations are also encouraged to form Card Communities (CCs) (or join existing CCs), making use of the guidelines set out in other parts of the CWA, if they wish.

As an illustration of a possible implementation of the eURI in a Card Community with an organisational architecture that is slightly different from that proposed by the eURI Workshop, we could consider the hypothetical implementation of the eURI in a Card Community set up to meet the MAS Requirements proposed by TB7/WG4.

The following diagram illustrates how this implementation could be achieved.

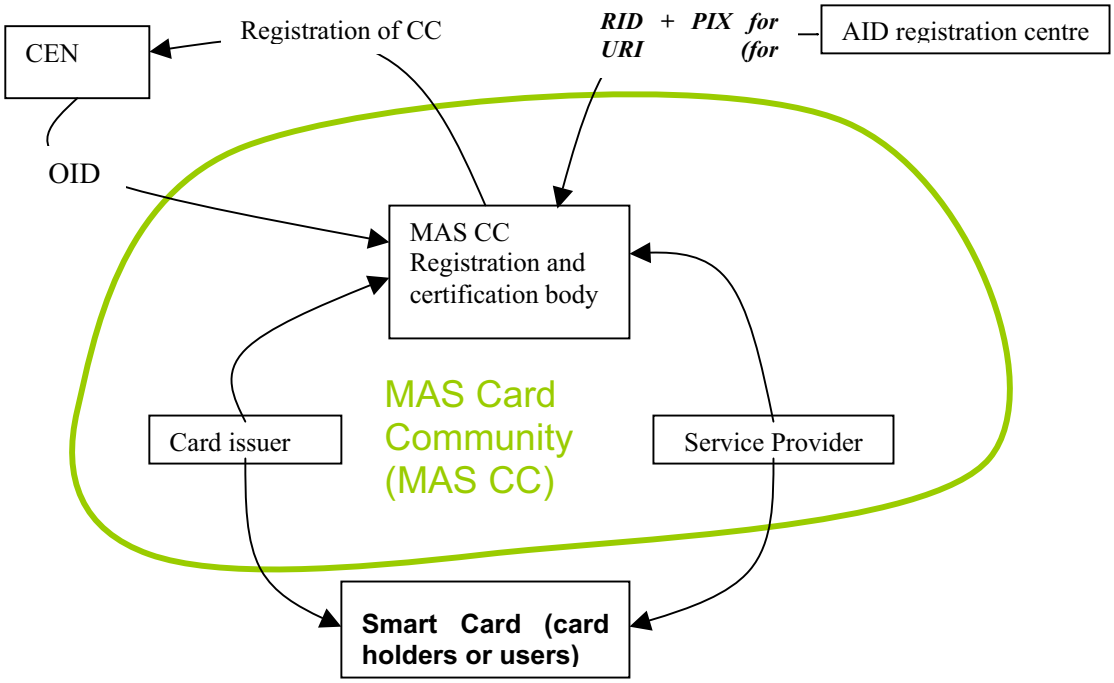


Figure A2-1 Hypothetical example of registration of the eURI as an application of a MAS Card Community

Once additional data elements not defined in the CWA are added to the eURI on the smart card, for interoperability the service provider or providers responsible for the definition of the additional data elements must be identifiable when accessing the smart card. For this purpose, compliance with the CWA requires those service providers, who together define and use an additional set of data elements, to form a Card Community (CC), register that CC with CEN, and obtain from CEN, and use on the smart card, a unique identifier for the CC. This identifier is formatted as an Object Identifier (OID) as defined in ISO/IEC 7816-6:1996 and ISO 8825-1.

At a lower level of deployment, the following parts of the eURI may be included on a smart card, and used, without the involvement of a Card Community:

- the mandatory component of the eURI, known as the Universal Card Holder Information or UCI
- the elements of the Personal Profile defined in this CWA
- management information (including pointers) supporting the above items

The UCI provides for the smart card to hold personal preference information, such as language preference and other terminal configuration preferences (e.g. large print).

At the lowest level of deployment, a Service Provider (who may well also be the Card Issuer) may load an eURI application and the UCI data structure without having any responsibility for the data held in the UCI. At a suitably configured terminal, the Card Holder may then load and manage personal preference information.

Relationship with existing standards

The eURI CWA makes use of existing standards where appropriate and in particular conforms to ISO/IEC 7816 and to EN 1332-4:1999, *Identification card systems – Man-Machine Interface – Part 4: Coding of user requirements for people with special needs*. References to standards such as EN1332-4

which define coding of eURI data elements and objects indicate that those standards have priority over the CWA.

EN 1332-4 is of particular interest to citizens with special requirements and it also provides support for EU directives concerning Special Needs and Design For All which have been enacted by member countries. However, while EN 1332-4 specifies the data structures necessary to support the directive, it does not specify how this data may be used and accessed.

The eURI CWA provides a method to identify the presence of and method of access to a specific data set on the citizen's smart card, including EN 1332-4 data structures where appropriate. In addition, the eURI provides a single, common data set for all uses of the smart card and all ICT services where access is provided by the smart card. The solution proposed will permit the Card Holder to perceive a common interface to ICT services tailored specifically to their wishes and requirements.

Moreover, the eURI CWA also supports and conforms to EN 1332-1:1999, *Identification card systems – Man-Machine Interface – Part 1: Design principles for the user interface*. In addition to the principles laid down in this standard, the CWA provides advice on some particular aspects of design for an interoperable, multi-application environment.

Privacy and data transparency

The eURI CWA takes account of the requirements of EN1332-4 and ANEC98/ICT/007 with regard to the privacy of personal information, data security and data transparency. It also permits Card Communities to comply with applicable data protection legislation.

Summary of content of the eURI CWA

The CWA 13987:200x specification includes:

- a normative definition of the eURI data set (Part 1);
- a card edge specification for access to the eURI (Part 1);
- supporting requirements, including logical message formatting for requesting and exchanging data elements (Part 1);
- informative guidelines on how the eURI may be implemented and deployed within an interoperable environment (Part 2); and
- an indication of the supporting organisational infrastructure necessary to maintain the ongoing operation of such an environment (Part 3).

The three parts of the CWA are mutually supportive. Hence, although Part 1 provides details of the fundamental requirements for implementing the open scheme, the reader will not gain a true understanding of the eURI's role as a mechanism for interoperability without reading the other two parts as well.

Current progress

A full draft of Part 1 (Normative) of the new CWA has been produced, and is available, along with other documents, at the Workshop web site http://www.uninfo.polito.it/WS_URI/default.htm.

Full drafts of Parts 2 and 3 (Informative) are currently in preparation.