NFC Handset APIs & Requirements
Version 2.0
November 2011

Security Classification – NON CONFIDENTIAL

Copyright Notice

Antitrust Notice

The information contain herein is in full compliance with the GSM Association's antitrust compliance policy.

Table of Contents

1 Introduction

1.1 Overview

With the increasing activity to deploy commercial Near Field Communication (NFC) services in a number of markets around the world, it is important to embrace common standards to promote the global interoperability of services, while maintaining the momentum to meet time-to-market requirements in certain markets.

This document, starting from the Pay-Buy-Mobile specifications, focuses on requirements for handsets to support UICC-based NFC services. It sets out a common framework of requirements, selecting options among those allowed by existing standards to ensure interoperability.

The body of this document sets out requirements that are agreed globally, according to the GSMA's processes for consulting its members.

In addition, this document contains Annexes containing 'elective requirements' that are supported by certain (named) operators as being applicable in their market. Within a market, the global requirements, and the applicable elective requirements, together comprise the total requirements for that market.

Furthermore, there are a number of areas where the global standards organisations (SDOs) have yet to complete their standardisation activities. To enable deployments to proceed prior to complete standardisation, a number of interim requirements have been defined in the elective requirements Annexes.

Elective requirements have been identified in Annexes 5, 6 and 7, including the operators who support them in their deployments.

It should be noted that this document is expected to evolve by:

- Embracing new standards as and when they are approved by the relevant SDOs;
- Adding further market-specific Elective Requirement Annexes if such a need is identified in a collection of markets;
- Migrating requirements into the main body of the documents, from Annexes, in the event that they become globally agreed.

The GSMA is defining the appropriate implementation for NFC based services within open Operating Systems (OS) and the device hardware which leverage the incumbent features of the OSs. Overall, the aim is to:

- Provide transferable solutions between different mobile device OSs and mobile devices;
- Provide the ecosystem with a quicker and simpler method for service deployment.

These ambitions will be fulfilled by adoption of the key NFC enablers, thereby facilitating a quicker time-to-market by providing clear and unambiguous handset requirements. This document is primarily targeted towards mobile device manufacturers.

1.2 Scope

The document scope is restricted to software platforms/application frameworks that are supported by multiple Original Equipment Manufacturers (OEMs) and defines at a high level the application architecture required to fulfil the *secured* NFC use cases. It further expands upon this, by detailing the particular handset Application Programming Interfaces (APIs) per Operating System (OS) to enable a *secured* service use case and the atomic requirements necessary to fulfil the NFC software architecture.

Additionally it builds upon the GSMA Pay-Buy-Mobile Specifications, by providing further details for specific use cases.

1.3 Use Cases/Services

The intended use cases for NFC can be grouped into *secured* and *non-secured* services. This document only targets the *secured* service use cases, and can include the following, but is not limited to:

- Plastic credit/debit card replacement
- Travel vouchers
- Business to Business transactions
- Secure access
- Mobile health
- IT system, e.g. RSA
- Touch and Pay
- Event ticketing.

For *secured* services, the device and Universal Integrated Circuit Card (UICC) must provide for a *secured* environment, i.e. an environment which satisfies the security needs of Service Providers', MNOs' and consumers' data.

1.4 Definition of Terms

Term	Description
AC	Access Control
API	Application Programming Interface
APDU	Application Protocol Data Unit
BIP	Bearer Independent Protocol
CLF	Contactless Frontend
JCP	Java Community Process
JVM	Java Virtual Machine
HCI	Host Controller Interface
JSR	Java Specification Request
MIDP	Mobile Information Device Profile
MNO	Mobile Network Operator
NFC	Near Field Communication
OS	Operating System
ODM	Original Device Manufacturer
OEM	Original Equipment Manufacturer
PoS	Point of sale
RIL	Radio Interface Layer
SCWS	Smart Card Web Server
SE	Secure Element
SIM	Subscriber Identity Module
SP	Service Provider
SWP	Single Wire Protocol
UI	User Interface
UICC	Universal Integrated Circuit Card (USIM)

2 References

Term	Description
3GPP Specifications	3GPP TS 31.111 (V7.16.0) Universal Subscriber Identity Module (USIM) Application Toolkit (USAT) 3GPP TS 31.116 (V7.1.0) Remote APDU Structure for (Universal) Subscriber Identity Module (U)SIM Toolkit applications
ETSI SCP	ETSI TS 102 613 (V7.8.0): UICC – Contactless Front End (Physical and

Specifications	data link layer characteristic) ETSI TS 102 622 (V7.8.0): UICC – Contactless Front End, (HCI) ETSI TS 102 223 (V7.16.0): Card Application Toolkit ETSI TS 102 226 (V.7.8.1): Remote APDU structure for UICC based applications
ISO Specifications	ISO/IEC 14443 Identification cards -- Contactless integrated circuit
JSR Specifications	JCP – JSR-000118: Mobile Information device Profile -Trusted MIDlet suites using X509 PKI JCP – JSR-000177: Security and Trust Services API for J2ME JCP – JSR-000257: Contactless Communication API
NFC Forum Specifications	NFCForum-TS-Type-1-Tag_1.0 NFCForum-TS-Type-2-Tag_1.0 NFCForum-TS-Type-4-Tag_1.0 NFCForum-SmartPoster_RTD_1.0 NFCForum-TS-GenericControlRTD_1.0 NFCForum-TS-RTD_1.0 NFCForum-TS-RTD_Text_1.0 NFCForum-TS-RTD_URI_1.0 TC-SEC-TS-RTD_Signature-1.0_draft_10
SIM Alliance	SIM Alliance Open Mobile API Specification v1.01
GSMA Requirements	GSMA Requirements for SWP NFC handsets V 4.0
GSMA Requirements	NFC UICC Requirement Specification, Version 2.0
AFSCM Specification	090929 - AFSCM TECH - PROC - NFC Mobile Handset High Level Requirements - V3.2

3 Terminology

As per IETF Requirements terminology, reference RFC 2119, the following terms have the following meaning.

Term	Description
SHALL	Denotes a mandatory requirement
SHOULD	Denotes a recommendation
M	Denotes Mandatory
O	Denotes Optional

4 Generic Device Architecture

4.1 Dual Application architecture

We promote the following application architecture (below) to pragmatically support the key use case of *secured* NFC services.

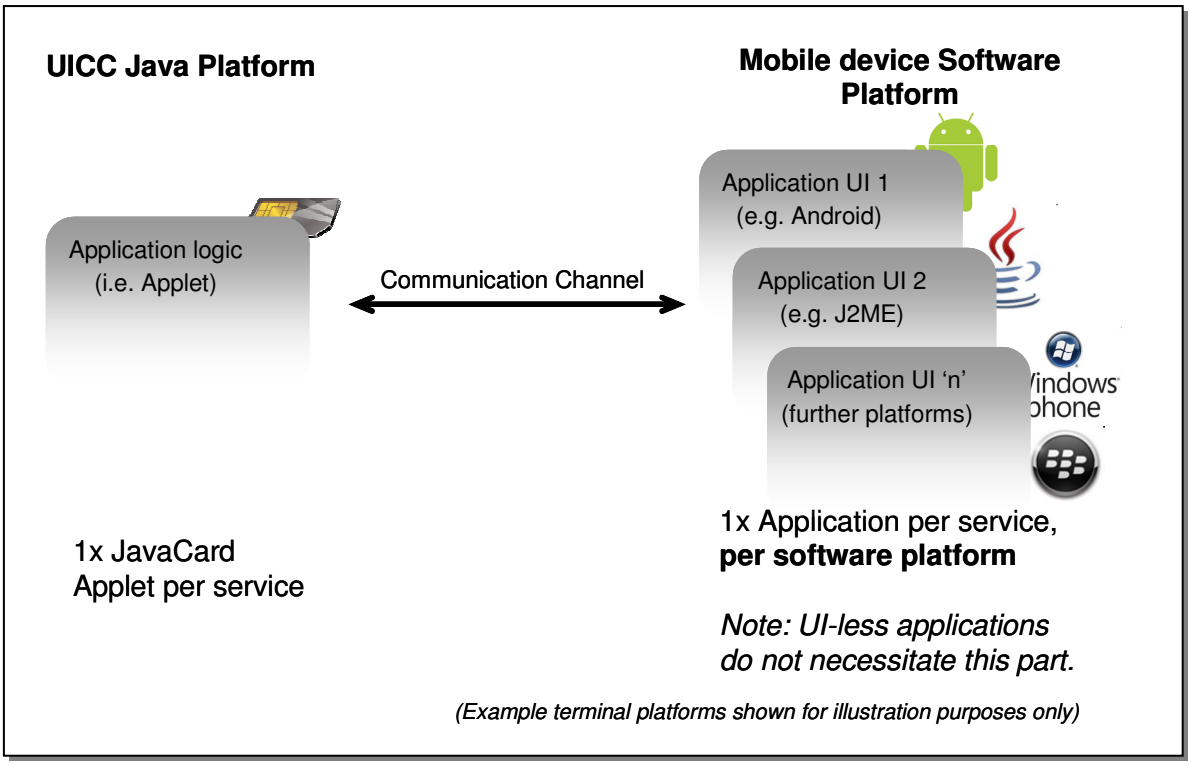

Figure 1: Dual application architecture

The mobile device User Interface (UI) application executes on the device OS, and is the consumer facing component. In this example, the UI application allows the customer to interact with the service functionalities, e.g. PoS (point of sale) in the financial services use case or a physical ticketing barrier in the case of an e-ticketing application. However this component is not seen as mandatory for all use cases, where the Service Provider (SP) could decide to have a UI-less service.

The *applet* component resides within the UICC, and works in tandem with the UI application when applicable. It performs actions such as holding *secure authentication keys* or *time-stamped transaction data* to provide for transaction resolution, history and fraud prevention etc.

Within this dual-application architecture for secured services, there is need for a consistent *communication channel* between these two applications; its use could be restricted to status information passed from the UICC to the UI for notifying the user on NFC events. As the communication channel is effectively accessing a *secured* storage space on the UICC, this communication channel itself must have attributes which allow it to be accessed by authorised UI applications.

The mandated method of communication between these two applications is ***APDU*** (Application Protocol Data Unit)**.**

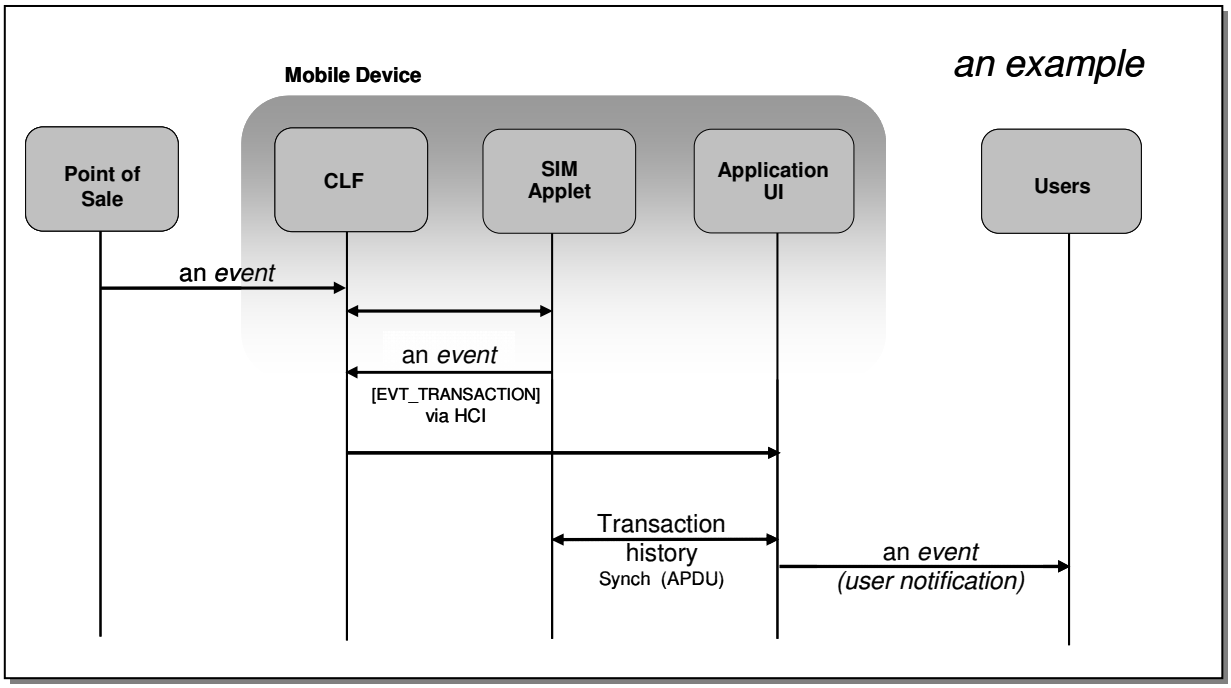

Figure 2: NFC Event routing, an example, for card-emulation mode

Figure 2 depicts the routing that the *event* will need to follow. The event is the trigger from the PoS to the user which indicates an activity in the NFC service. From this can be determined the nature of this *event* between the various components, for example where the *event* needs to be protected and has attributes which will allow for, or not allow for, any modification.

4.2 Security

For the *secured* services use case it is imperative for MNOs (Mobile Network Operators) and SPs to provide secured and trusted communication along the end-to-end chain of the various components necessary to provide for the *secured services* use cases.

Two key areas where security is important are the UICC and the privileges available to communicate with the UICC NFC service applet.

The UICC will securely hold protected information (within the Secure Domain), and provide a controlled access path to relevant parts of its internal memory.[1]

[1] Further information can be obtained from the *NFC UICC requirement specification*

4.3 Generic Device APIs

The following illustration (Figure 3) gives an overview of the software components within a device as related to NFC. It includes key components required to satisfy the dual application architecture proposal, which delivers key use cases for NFC.

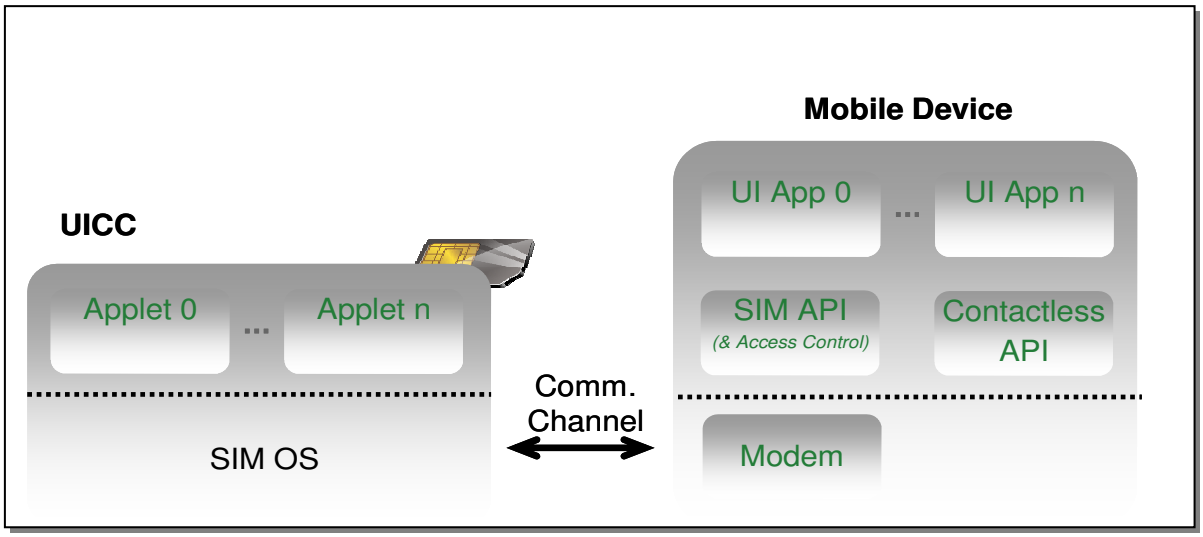

Figure 3: Handset API generic software stack

For Open OSs where a software solution does not exist, the open and endorsed solution in Section 5.2 “NFC Architecture” for Android can be implemented: this will allow for easier transferability of MNO and SP services solutions.

4.4 Handling Multiple Secure Elements

This document sets out requirements for handsets to support UICC-based NFC services. In some markets additional secure elements may coexist, in which case the following requirements apply.

When several Secure Elements, (the UICC SE and others) exist within the mobile device, simply providing access is not enough; the OS also needs to provide a way of managing them.

At a high level, the device should be able to provide for the following operations:

- Select a new Secure Element for future transactions (UICC being the default);
- Get the active Secure Element;
- Receive “Intent” when a card transaction event is detected by the active Secure Element.

The operation of defaulting the UICC SE directly implies that the transaction events are routed from the CLF (Contactless Frontend) to the UICC.

4.5 Mobile Wallet

The *Mobile Wallet* is intended to facilitate the user experience, and allow the MNO or SP to differentiate by providing targeted and convenient access to the NFC Services within the mobile device and UICC. The wallet application, for example, can typically list all SP services loaded into the mobile device or UICC and displays their current status. Additionally, this application may also allow the user to manage the NFC settings of their mobile device.

5 Android

5.1 Introduction

Android does not differ from other platforms regarding the NFC ecosystem. It is likely that Android will provide, as standard components, an NFC controller and one or more Secure Elements (SEs). Access to NFC has already been introduced with Gingerbread *(Android v2.3 release)*; however one of the major issues remaining, is the lack of access to the UICC SE.

5.2 NFC Architecture

As previously stated, and unlike JavaME with the help of the JSR177, Android does not currently provide an API for accessing the different Secure Elements. However, this situation is likely to change.

Recently, the SIM Alliance released the "Open Mobile API specification". From this document, any mobile device manufacturer will be able to provide standardised APIs to developers to have access to different Secure Elements such as the UICC SE.

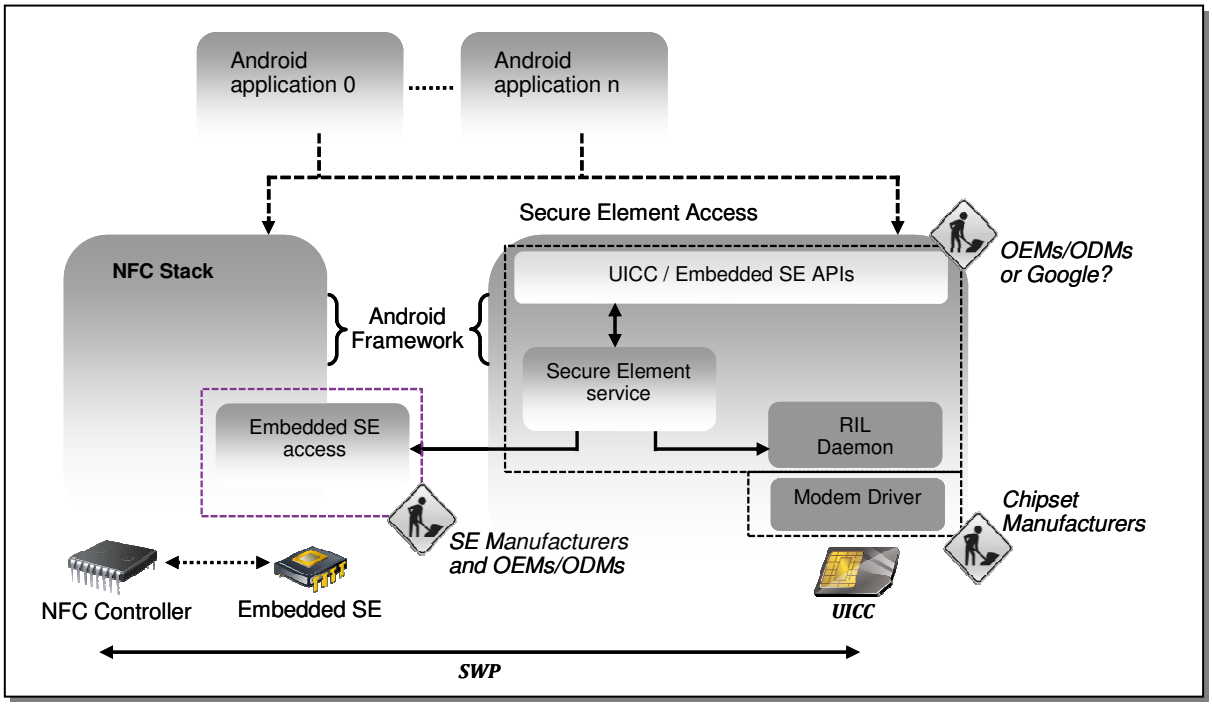

Figure 4: Android NFC software stack

Figure 4 gives an overview of a potential Android implementation for this proposal. The core of this architecture is encapsulated in an Android service. Having a single service ensures that security checks (who is accessing the service) and resource management (freeing up a logical channel) can be guaranteed.

his background component relies on a RIL daemon extension for accessing the UICC and on some specific libraries, for communicating with any embedded secure elements.

5.3 Key APIs

All of the APIs necessary for accessing the SE are detailed by SIM platform and are built around four main classes:

1. ***SEService***: As the entry point for establishing the APDU communication, it is used to connect to the infrastructure and to retrieve the list of available Secure Elements.

2. ***Reader***: This class represents an instance of one Secure Element connected to the device. They can be removable (or not) physical or virtual devices..
3. ***Session***: A session characterises a set of event connections with the Secure Element.
4. ***Channel***: An instance of this class symbolizes an ISO/IEC 7816-4 channel opened to a Secure Element. It can be either a logical channel or the default channel.

In addition to these classes, an interface is also available. Its objective is to receive call-backs when an SEService is connected to the device.

A complete description of the different functions is available in the document 'SIM Alliance Open Mobile API Specification v1.01'.

5.4 Security

Access to services inside a Secure Element is not without complications as a high level of security is required by some Service Providers. It is necessary to manage which applications on the devices communicate with applets in the UICC. In addition to existing protection mechanisms provided by Android, the main purpose of this *Access Control* is typically to prevent service attacks from malware applications. Further details of a proposed Access Control solution are provided in Annex 5, Elective Requirements: Access Control.

6 JavaME

6.1 Introduction

The following J2ME NFC architecture is used within mobile devices currently commercialised for NFC services. They host and run services based on the UICC card with a companion application to perform user-friendly display and interactions.

6.2 NFC Architecture

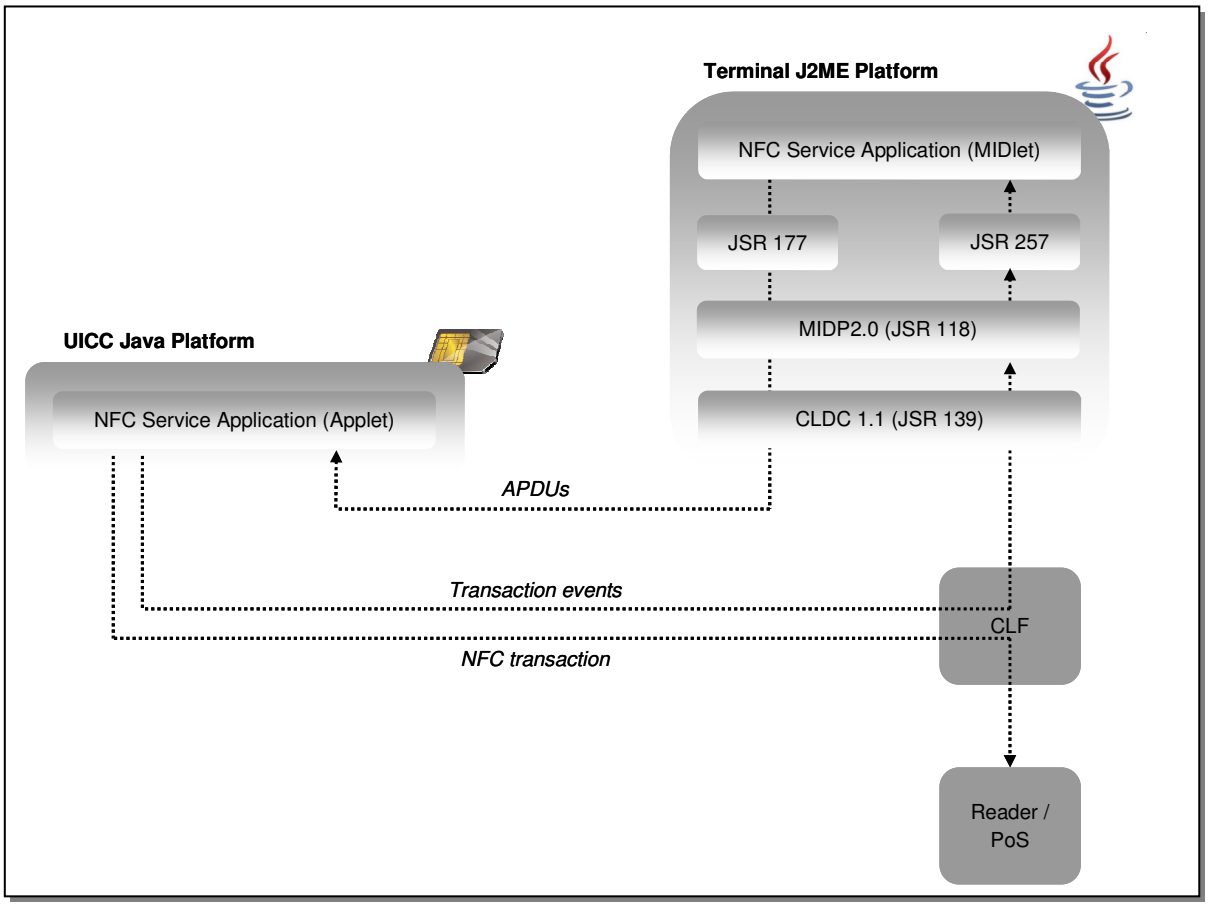

Figure 5: JavaME NFC software stack

6.3 Key APIs

Service applications (MIDlets) communicate with their partner UICC application through JSR177.

Contactless transactions are registered for using JSR257 (push MIDlet mechanism). These events trigger the start of the MIDlet, which then exchanges data with the associated UICC application.

6.4 Security

Presently, the handset is not considered a safe environment when compared to the UICC. Therefore service developers should not make assumptions about the security that the mobile phone OS provides to their data.

However, to protect against unauthorised applications, access to UICC is limited as follows: An application's signature which does not link the application to a trusted domain has no access to the UICC;

- UICC defines the chain of certificates that the MIDlets need to be signed up with, in order to be attached to a trusted domain;
- Further restrictions can be specified using Access Control Files and Applets to limit some trusted applications to only access certain UICC applications and to only send certain APDUs (as per JSR177 Annex A).

Protection of card transaction events is not granted until the associated service is installed. Once installed, the MIDlet registered on the transaction events will block further registrations on these notifications.

6.5 API requirements

API_J2ME_REQ_01	The J2ME implementation SHALL support JSR 177: UICC secure application management (APDU Exchange) *This capability enables to communicate with applications loaded into the UICC by exchanging ISO 7816-4 APDU.*
API_J2ME_REQ_02	The mobile device SHALL support SATSA-APDU package as per JSR 177 specifications.
API_J2ME_REQ_03	The mobile device SHALL support Annex A of JSR 177 specifications.
API_J2ME_REQ_04	The mobile device SHALL support Annex B of JSR 177 specifications.
API_J2ME_REQ_05	The mobile device SHALL support JSR 257: Contactless Communication API specifications.
API_J2ME_REQ_06	For JSR 257 implementation, the mobile device SHALL NOT allow access to the UICC.

Note: Where JSR 177 should be used for communication with the UICC.

API_J2ME_REQ_07	The mobile device SHALL check that the relationship between the UI application and its corresponding UICC applet exists before allowing the UI application registration to an event within the Push registry.

6.6 Delta to Generic Architecture

The current implementation has been accepted by major players within the ecosystem. Services such as *banking* and *transport* are already deployed with this architecture. Therefore, currently, no future work is planned regarding this platform. Recent efforts in this area aim to develop various handset models with a wider range of suppliers, in particular targeting the low-tier device market space.

7 Key NFC requirements

7.1 NFC controller Required Features

NFC_REQ_08	The mobile device including NFC chipset and antenna SHALL be compliant with contactless reader infrastructure (ISO/IEC 14443 A & B, protocol implemented in the UICC)
NFC_REQ_09	The mobile device SHALL support Card-emulation as per ISO/IEC 14443 Type A and Type B PICC.
NFC_REQ_10	The mobile device SHALL support Reader/Writer Mode as per (ISO/IEC 14443 Type A and Type B PCD.

NFC_REQ_11 The mobile device SHALL support NFC Forum Tag Type 1

NFC_REQ_12 The mobile device SHALL support NFC Forum Tag Type 2

NFC_REQ_12.1 The mobile device SHALL support NFC Forum Tag Type 3

NFC_REQ_13 The mobile device SHALL support NFC Forum Tag Type 4

NFC_REQ_14 The reader mode events SHALL be routed exclusively to the UICC or the Application processor at any one time.

NFC_REQ_15 Where the default routing for the reader mode events SHALL be via the Application processor.

NFC_REQ_16 The CLF SHALL route the reader mode events to the UICC when the UICC registers itself to receive the reader mode events.

NFC_REQ_17 Automatic and continuous switch between card emulation and reader mode. If this switch is permanent, the increase of the power consumption of the phone in standby mode shall be less than 10% compared to the standby time when the NFC is totally off. If this 10% threshold cannot be reached, the automatic switch shall be applied only when the keyboard is activated or the screen backlight activated.

Note: Default mode Card emulation mode, with a poll for Reader mode, the frequency for the Reader mode poll SHALL be such that the battery power consumption is kept to a minimum. This implementation will require on-going optimisation; however, the aim is to provide good responsiveness to the consumer.

NFC_REQ_18 Contactless tunnelling (CLT) mode SHALL be supported for SWP (per ETSI TS 102.613)

The NFC controller shall support the following ETSI interfaces with the UICC.

NFC_REQ_19 The NFC controller SHALL support SWP interface with the UICC as per ETSI TS 102.613 Rel 7. (latest)

NFC_REQ_20 The NFC controller SHALL support HCI interface with the UICC as per ETSI TS 102.622 Rel 7. (latest)

NFC_REQ_21 Card Emulation mode SHALL be enabled as soon as the NFC hardware is turned on.

NFC_REQ_22 For Card emulation mode, during battery full mode the read distance SHALL be in the 0cm – 4cms range, and for battery off mode the read distance SHALL be in the 0cm – 2cms range.

7.2 Mobile Device Modem Support

NFC_MOD_23 The modem/baseband SHALL enable access to logical channels from the application layer.

Note: This is intended to be an “internal” API, and is not for the purposes of application development.

7.3 Mobile Device APIs

API_REQ_24 APDU APIs SHALL prevent access to basic channel (channel 0).

Note: For implementation purposes this can be achieved by raising an exception.

API_REQ_25 APDU APIs SHALL prevent access to SELECT CHANNEL command.

API_REQ_26 APDU APIs SHALL prevent access to MANAGE CHANNEL command.

API_REQ_27 Android devices SHALL provide Open Mobile SIM APIs (per SIM Alliance spec) for developers to use.

Note: Reference implementations for Open Mobile APIs and access control exist for Android.

7.4 Mobile Device UI requirements

UI_REQ_28 The API SHALL be made available which allows to enable or disable the NFC hardware. (Note: as with Wi-Fi, GPS etc).

7.5 Mobile Device APN management

APN_REQ_29 For mobile devices supporting multiple APNs, the device SHALL be able to set-up a SIM OTA channel using the APN information that is provided in the OPEN CHANNEL command. For devices which are configured as "Always-ON" and only support a single APN, the APN information provided in the OPEN CHANNEL command SHALL be ignored by the device and the default APN SHALL be used.

7.6 UI application triggering

The NFC controller must be able to trigger the appropriate UI application.

UIApp_REQ_30 The mobile NFC Handset SHALL support HCI event EVT_TRANSACTION as per ETSI TS 102.622

UIApp_REQ_31 The AID parameter SHALL be used during the process of triggering the UI application.

7.7 Remote Management of NFC services (Access to UICC in connected mode)

RemMan_REQ_32 The mobile device SHALL support BIP in UICC client mode for UDP.

RemMan_REQ_33 The mobile device SHALL support two concurrent channels, BIP in UICC client mode.

RemMan_REQ_34 The mobile device SHALL support the SMS push (per ETSI TS 102.226) to establish an open BIP channel as per ETSI TS 102.223 Open Channel Command

7.8 NFC Controller TAG Support

TAG_REQ_35 The mobile device SHALL support Writer Mode.

TAG_REQ_36 The transaction SHALL take 500ms or less.

(e.g. reading TAG to display, paying for transportation, etc.)

TAG_REQ_37 The mobile device SHALL be able to read/write the NFC Forum Smart Poster RTD.

TAG_REQ_38 The mobile device SHALL be able to verify the TAG signature when associated certificates are installed. The TAG signature follows the NFC Forum RTD signature.

TAG_REQ_39 The TAG SHALL be read from a distance of 0cm – 2 cms and SHOULD be read within the 2cms – 4cms range.

TAG_REQ_40 Following a TAG read, the user SHALL be prompted before any action is performed, and SHALL be able to dismiss this prompt.

TAG_REQ_41 When reading a *TAG* that cannot be authenticated by the *TAG* reading application, the user SHALL be explicitly prompted that the *TAG* is un-trusted. However, reading the *TAG* SHALL remain possible. It SHALL be possible to enable/disable this prompt at factory settings.

7.9 Multiple SE support (UICC SE and others)

MultiSE_REQ_42 Only one SE SHALL be active at any one time.

MultiSE_REQ_43 The mobile device software platform SHALL expose an API to select the active SE.

MultiSE_REQ_44 If there is more than one SE, the default active SE SHALL be the UICC SE.

7.10 SCWS support

Due to the mobile device distribution model in certain markets, a UICC driven approach is required. This is where the Smart Card Web Server (SCWS) feature is required to expose the service UI to the device, through an appropriate rendering application, e.g. the on board device browser.

SCWS_REQ_45 The mobile device SHALL support, BIP in UICC server mode as per ETSI TS 102 223 R7 letter class “e”.

Annex 1: Authors

This document defines the Handset APIs and requirements necessary to deliver NFC Secured Services, and has been jointly developed by France Télécom, Telefónica, Telecom Italia, Deutsche Telekom and Vodafone. This specification will be shared with other operators, device manufacturers, and with Service Providers and 3rd Party developers.

Contributors and reviewers:

Fabio Di Benedetto (Telecom Italia)

Mike Grover (Vodafone)

Bas Hoeksel (Vodafone)

Natalia Jimeno (Telefónica)

Erwan Louet (Orange)

Simonetta Mangiabene (Telecom Italia)

Thierry Morel (Orange)

Mehdi Ouled-Ali (Deutsche Telekom)

Davide Pratone (Telecom Italia)

Gilles Printemps (Deutsche Telekom)

Fabio Ricciato (Telecom Italia)

Ahmad Saif (Orange)

Sameer Tiku (Vodafone) + Document Editor

Annex 2: CAT Letter Classes Support

The following table lists the minimum letter classes support for NFC handsets.

Letter classes	Command/function description	Support
c	Proactive command: LAUNCH BROWSER	SHALL
	Event download: Browser termination event	
	Event download: Browsing status event	
e	Proactive command: OPEN CHANNEL	SHALL[2]
	Proactive command: CLOSE CHANNEL	
	Proactive command: RECEIVE DATA	
	Proactive command: SEND DATA	
	Proactive command: GET CHANNEL STATUS	
	Event download: Data available	
	Event download: Channel status	
l	Proactive command: ACTIVATE	SHALL
m	Event download: HCI connectivity event	SHALL
r	Proactive command: CONTACTLESS STATE CHANGED	SHOULD
	Event download: Contactless state request	

RemMan_REQ_46 The mobile device SHOULD support BIP in UICC client mode for TCP.

[2] Requirement is Market dependent

Annex 3: UICC CLF Interface, Supported Options as per ETSI TS 102 613

Minimum set required SHALL be:

Item	Option	Support	Mnemonic
1	Class B	M	O_CLASS_B
2	Class C full power mode	M	O_CLASS_C_FULL
3	Class C low power mode	M	O_CLASS_C_LOW
6	Terminal supports DEACTIVATED followed by subsequent SWP interface activation in full power mode	M	O_DEAC_SUBACT_FULL
7	Window size of 3	M	O_WS_3
8	Window size of 4 (see note)	M	O_WS_4
9	HCI as per TS 102 622 [4]	M	O_102_622
10	CLT, ISO/IEC 14443 [5] Type A	M	O_CLT_A
12	RF technology ISO/IEC 14443-4 [6] Type A	M	O_RFTYPE_A
13	RF technology ISO/IEC 14443-4 [6] Type B	M	O_RFTYPE_B
NOTE: If the terminal supports O_WS_4, then it also shall support O_WS_3.			

Annex 4: Common requirements between 'NFC Handset API and Requirements' and 'GSMA Requirements for SWP NFC handsets' document

The below table highlights a subset of the requirements within this document which can be directly mapped to, i.e. are equivalent to, requirements already introduced within 'GSMA Requirements for SWP NFC handsets v4.0'.

NFC Handset APIs Requirements		GSMA Requirements for SWP NFC Handsets
NFC controller Required Features		
NFC_REQ_08: The mobile device, NFC chipset and antenna SHALL be compliant with contactless reader infrastructure (ISO/IEC 14443 A & B, protocol implemented in the UICC)		3.2.1-2 The handset is required to be fully interoperable with the existing contactless infrastructures being deployed for applications such as proximity payment, transport ticketing, etc. As a minimum therefore, the ISO 14443 Type A, Type B and ISO 18092 passive mode communication protocols must be supported.
NFC_REQ_09: The mobile device SHALL support Card-emulation as per ISO/IEC 14443 A & B PICC.		3.1-2 Support of Card emulation mode
NFC_REQ_10: The mobile device SHALL support Reader/Writer Mode as per (ISO/IEC 14443 Type A and Type B PCD.		3.2.1-1 Reader/Writer mode – supports NFC Forum tags (Types 1 to 4)
NFC_REQ_11: The mobile device SHALL support NFC Forum Tag Type 1		3.2.1-1 Reader/Writer mode – supports NFC Forum tags (Types 1 to 4)
NFC_REQ_12: The mobile device SHALL support NFC Forum Tag Type 2		3.2.1-1 Reader/Writer mode – supports NFC Forum tags (Types 1 to 4)
NFC_REQ_13: The mobile device SHALL support NFC Forum Tag Type 4		3.2.1-1 Reader/Writer mode – supports NFC Forum tags (Types 1 to 4)
NFC_REQ_5.4: At the end of the standby time, when the mobile device is automatically switched off, the mobile device SHALL have the capability to supply a source of power to		3.5-2 Support of Power Low and Power Off transactions should be at the discretion of the service provider and MNO. To facilitate the above requirement, it shall be possible for

perform a few transactions (15 transactions) in card emulation mode for at least 24 hours.		the applications on the secure element to detect the current power state of the device.
NFC_REQ_18 Contactless tunnelling (CLT) mode SHALL be supported for SWP (per ETSI TS 102.613)		4.10 Mifare Support
NFC_RFQ_19: The NFC controller SHALL support SWP interface with the UICC as per ETSI TS 102.613 Rel 7. (latest)		3.2.-1 A NFC controller/chipset connected to the UICC and also to the RF antenna and baseband controller. 3.2.3-4 The physical interface and the data link layer between the UICC and the NFC chip shall be based on the Single Wire Protocol specification as defined in ETSI TS 102 613. CLT LLC defined in ETSI TS 102 613 must be supported in order to ensure ISO/IEC 18092 212/424 kbps passive mode based card emulation and ISO/IEC 1444-3 Type A based card emulation.
NFC_RFQ_20: The NFC controller SHALL support HCI interface with the UICC as per ETSI TS 102.622 Rel 7. (latest)		3.2-1 A standardised connection to the UICC and also to the RF antenna and baseband controller. 3.2.3-5 The logical interface between the UICC and the NFC chip shall be based on the Host Controller Interface (HCI) specification as defined in ETSI TS 102 622.
NFC Controller TAG Support		
TAG_REQ_36: The transaction SHALL take 500ms or less. (*e.g. reading TAG to display, paying for transportation, etc)		3.8-1 While in card emulation mode with battery on, the performance of the phone shall be such that the overall transaction duration time shall not be significantly different from the transaction duration time when contactless cards are used.
NFC_REQ_22: For Card emulation mode, during battery full mode the read distance SHALL be in the 0cm – 4cms range, and for battery off mode the read distance SHALL be in the 0cm – 2cms range.		3.5-3 The RF power provided in the handset in the card emulation mode shall allow a transaction to take place when the handset is at *any* point between zero and 4 cm from the terminal. This requirement is relaxed for transactions in the Power Off mode to *some* point between 0 and 4 cm from the terminal.
Multiple SE support		
MultiSE_REQ_5.6: The mobile device SHOULD support UICC SE only.		3.2.3-2 Mobile Network Operators (MNOs) have recommended the UICC as the most appropriate NFC Secure Element (SE) in mobile phones for card emulation and

		reader/writer modes, offering many unique advantages for the customer, including: universal deployment, portability, remote management, standards based solution and a long operational lifecycle (Ref. 3). The interface between the UICC and the handset must be based on the approved standards as identified in this document.
User Control/Interface		
UI_REQ_28: An API SHALL be made available which allows to enable or disable the NFC hardware. (Note: as it is possible with Wi-Fi,GPS etc).		3.6.1-1 The handset shall provide a simple mechanism for the user to ensure that NFC radio functions cannot be utilised without the user's awareness and explicit consent. This mechanism should be controllable from the user interface application. 4.5-3 The user should have the ability to enable or disable all NFC services with a minimum set of key strokes or by using a physical switch (see Security section). 4.9-2 It shall be possible for the NFC application to select the NFC mode automatically. For example, this could be used to improve the performance of the NFC service.

Annex 5: Elective Requirements: Access Control

The requirements within this Annex are mandated by those MNOs listed below in Annex 5.2.

5.1 Requirements

Access Control

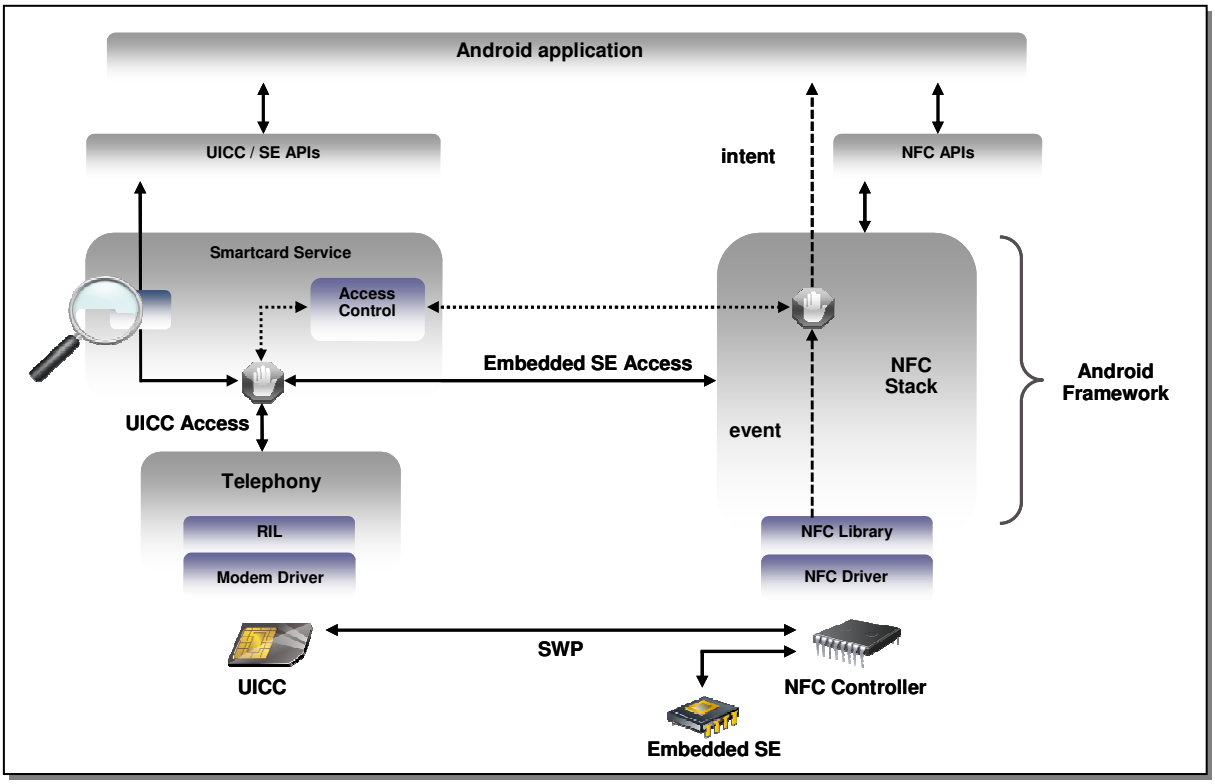

Figure 6: Android Security (Access Control Integration)

The main objective of the Access Control mechanism is to protect access to the two main functions defined by the SIM Alliance API: *"Open Basic Channel"* and *"Open Logical Channel"*. It will grant or refuse the communication to applets stored in the UICC SE. These decisions will be based on rules defined by the MNO.

Similarly, it can also be used to verify events received from the CLF, providing a method of filtering events being sent to the relevant application being alerted. The diagram above (Figure 6) describes in detail how this component will function.

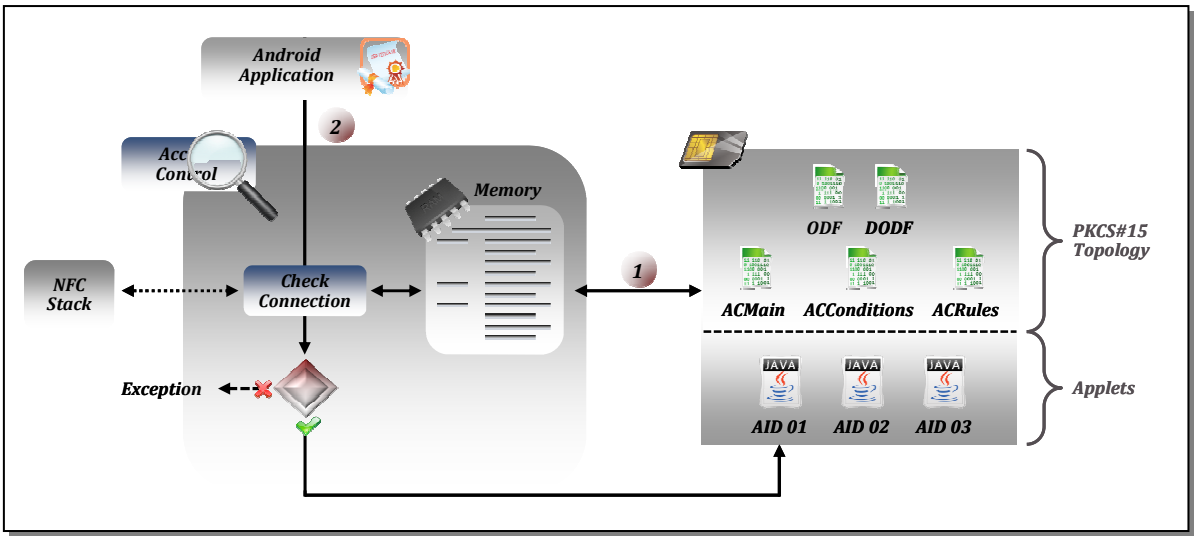

Figure 7: Access Control Mechanism

Similarly to JSR177, the rules used to grant or reject access are based on a standardised PKCS#15 topology. To increase performance and to avoid physical access to the UICC each time a control check is done, most of the data which is read and put may be stored in a cached memory during boot time.

From this cache, the Access Control can determine if the relationship between the Android application and the SE applet (application signature/AID) is valid, then progress to authorise a communication or send an exception.

A specification for an implementation of this mechanism is presented in Annex 6 of this document.

API_REQ_5.1	Open OS devices SHALL provide SIM API access control as per Section 5.1 Access control, above.
API_REQ_5.2	When no access condition files are found on the UICC the APDU API SHALL deny access to the UICC.

Note: This behaviour is different from the 'default grant access' policy that is specified in the referenced v1.3.1.

UIApp_REQ_5.3	The handset SHOULD prevent the case that an application UI is triggered from an applet when the access conditions would not allow the application UI to exchange APDUs with this applet.

5.2 MNOs electing to support this Annex

This Annex applies to the following National MNO Companies.

Note: this list is under construction.

Telecom Italia

France Telecom

Deutsche Telekom AG and subsidiaries

Telefónica Ireland
Telefónica UK
Telefónica Germany
Telefónica Czech Republic
Telefónica Slovakia
Telefónica Spain
Telefónica Mexico
Telefónica Nicaragua
Telefónica Guatemala
Telefónica Panama
Telefónica Costa Rica
Telefónica El Salvador
Telefónica Colombia
Telefónica Venezuela
Telefónica Brasil
Telefónica Ecuador
Telefónica Peru
Telefónica Chile
Telefónica Uruguay
Telefónica Argentina
Vodafone Spain
Vodafone Netherlands
Vodafone Germany
Vodafone Australia
Vodafone Czech Republic
Vodafone Greece
Vodafone India
Vodafone Ireland
Vodafone Italy
Vodafone Hungary
Vodafone Malta
Vodafone Portugal
Vodafone Qatar
Vodafone Romania
Vodafone Albania
Vodafone Turkey
Vodafone UK

Vodafone New Zealand

Vodafone Ghana

Vodafone Egypt

Vodafone South Africa

KPN

Optus

SMART

China Unicom

Turkcell

Telenor

Annex 6: Secure Element access control technical solution v1.3.1

Introduction

This is an elective annex as defined in section 1.1. It should be read in conjunction with any Elective Annex that refers to it – currently Annex 5 only.

Purpose

The amount of new applications and new security requiring use cases deployed on mobile devices is increasing every day. In several areas (like for example Mobile NFC payment), these new applications will rely for one part on the device itself and for the other part on secure elements (UICC, embedded Secure Elements, etc…), characterized by a tamper proof resistance. In order to address such distributed configuration new APIs (called Secure Element Access API) are needed on the device OS to enable the communication between the device applications and the applications running in the Secure Elements. In order to enforce the security model of the Access APIs, there is a need to provide an additional security mechanism that will restrict the access to secure elements applications only to authorized terminal applications.

This security mechanism, called the Access Control to the Secure Element, defined in this specification, comes in addition to existing protection mechanisms (like permissions or security OS policy limiting the access to sensitive APIs). It is designed to provide a way for the secure element issuer or secure element application issuers to prevent their secure element applications to be used by unauthorized terminal applications.

The purpose of this access control is to prevent unauthorized access to resources in the secure elements and typically to prevent denial of services attacks (PIN blocking, selection of non multi-selectable applets).

This document provides a specification for an Access Control mechanism that can be applied to any API providing an access to a Secure Element. This specification is device platform agnostic and could be used with any device platform compliant with a set of requirements described later in this document (see chapter 1.3).

Putting in place a security model requires significant effort and time and it is therefore very important to stay compliant with security models that were already deployed in the field and validated by Mobile Network Operators, Banking Organisations and Service Providers. As a consequence, the objective of this specification is to reference as much as possible existing standards (like PKCS#15) and so avoiding proprietary implementations.

Scope

This specification defines the access control mechanism, what it controls, and how the relevant data are stored in the Secure Element using a PKCS#15 structure.

Terminal platform prerequisites

The prerequisites on the terminal side are:

- There is a terminal API providing access to the Secure Elements physically to this terminal.
- This API has a connection-based behaviour: a terminal application opens a channel to a secure element application and then uses this channel to exchange APDUs with the secure element application.
- This API can support an access control mechanism that will be applied when opening a connection with a secure element application.
- This API shall not allow the use of MANAGE_CHANNEL and SELECT_BY_NAME commands by terminal applications. These commands are used internally by the

implementation but forbidden at the application level, which must use a higher abstraction level.

- Terminal application provisioning is based on a signature scheme: the terminal operating system verifies the signature of the applications before their installation. Note that this model is available in most of the terminal platforms.
- The terminal operating system provides a way to retrieve the certificate chain (or at least the SHA-1 of each certificate in the chain) used to sign the terminal application.

References

The following table gives the references of the specifications used in this document:

Document Title	Description	Organization	Version and date	Ref
ISO7916-4	Identification cards -- Integrated circuit cards -- Part 4: Organization, security and commands for interchange	ISO	ISO/IEC 7816-4:2005 January 5, 2005	1
PKCS#15	Cryptographic Token Information Syntax Standard	RSA	V 1.1 June 6, 2000	2

Description of the Access Control mechanism

A signature-based mechanism

The specified access control mechanism relies on the ability to retrieve, from the terminal, the chain of certificates used to sign the terminal application.

This certificate chain is used to verify that the terminal application was signed by an authority recognized by the secure element issuer and thus trusted to be granted with the access to one or several applications in the Secure Element.

By using the certificate chain and not only the last certificate in a chain, a delegation of trust can be ensured. A secure element issuer can grant access to the secure element applications to any terminal application coming from a recognised organisation (if it provisions a certificate for the organisation that will be used as part of the certification chain); this principle also allows to grant access to only one terminal application by generating a specific certificate for a dedicated application.

A 'per Secure Element' mechanism

It is the responsibility of each secure element issuer or secure element application issuer to define which terminal application will be granted with the right to use the secure element applications. Therefore the data used to perform this recognition are stored in the secure element.

Some rules are defined to allow a harmonized management of access rights:

If a terminal application requires access to an application in a secure element, then the access control mechanism shall only use the data stored in this secure element to check whether or not the access can be granted. In case two secure elements are connected to a device, it is strictly forbidden to use the data stored in a secure element (like SE#2) to give access to an application running in another secure element (like SE#1).

If access control data are present in the secure element, then a policy based on these data is applied.

If no access control data are present in a secure element or if access control data are empty, then access to all application residing in this secure element is forbidden

A connection-based access control

The access control is involved when the terminal application opens a connection with a secure element application on any channel. This can be done in two ways:

- Opening a channel by specifying the AID of the secure element application
- Opening a channel without specifying the AID of a secure element application (default application)

In the first case, the AID is used by the access control mechanism to check whether or not the terminal application has the right to access the targeted application in the secure element.

In the second case, the access control mechanism must check if there is a rule involving the default selected secure element application and if this rule grants access to the terminal application. The “default selected application” must be understood in the Global Platform context, i.e. the application selected on the basic channel after a reset of the secure element. On the UICC, the default selected application should be the default NAA (SIM, USIM, CSIM, etc.).

The current specification does not propose to manage differently access control on basic channel and on other logical channels. This is the responsibility of the API and/or secure element to allocate a channel to the terminal application. It can be the basic channel if available or any other logical channel.

Note; that according to the SIM alliance Secure Element Access API specification, using the basic channel on the UICC might be forbidden.

Once the connection has been established (i.e. if access is granted), the Secure Element Access API process the APDU commands according to its own behaviour. As specified in the terminal prerequisites, this API shall not allow sending MANAGE_CHANNEL and SELECT_BY_NAME commands. Thanks to this filtering, no other Secure Element application can be selected by the terminal application and the access control mechanism cannot be bypassed.

Additional APDU filtering (based on pattern matching) might be added in a future release of this specification.

For more information on the management of the rules, please refer to the “Combination of the access control” rules chapter.

Architecture of the Access Control mechanism

The architecture involves three entities:

- The Terminal Application requiring to access to a Secure Element Application
- The Secure Element Application
- The Secure Element Access API that provides this access, which includes the Access Control mechanism

The Access Control mechanism itself is divided in three sub-entities:

- The Access Control Rules Database & Engine located in the terminal
- The Access Control Filtering located in the terminal
- The Access Control Data stored in each secure element

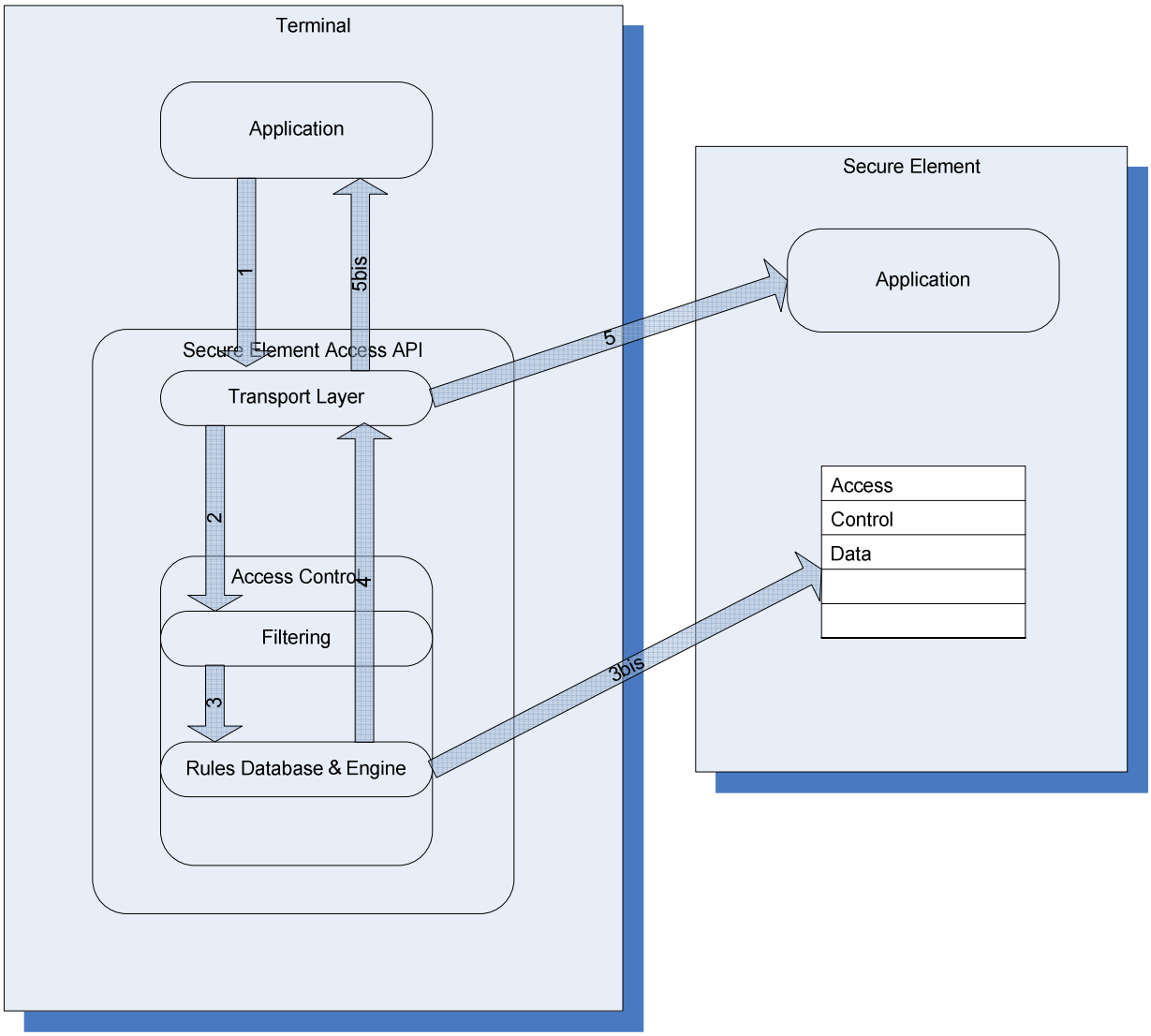

Figure 8: Architecture Overview

The Access Control Rules Database & Engine

This module is in charge of building and maintaining a representation of the access control rules in the terminal. This information is used when appropriate by the filtering engine.

These rules are built from the access control data stored in the secure element.

A Secure Element has no access data (thus always forbidding an access to all its applications) if one of the following conditions is valid:

- There is no PKCS#15 file structure in the Secure Element (i.e.; there is no PKCS#15 application available on the secure element and in case of the UICC, there is also no application/ PKCS#15 DF referenced in the EFdir file).
- There is a PKCS#15 file structure but the specific OID of the access control mechanism is no present in the DODF file.

By default, and in any conditions, the device shall apply a conservative security approach meaning that all accesses are denied if there are no explicit rules specified.

The rules must be built on a per-secure element basis, and must apply in the same way, separately, for each secure element. It shall be forbidden for a secure element to hold access control data regarding applications running in another secure element.

In addition of the rules derived from the secure element access control data, this module is also responsible for retrieving the signatures of the terminal applications (at least the ones needing to use the Secure Element Access API). More specifically, what must be retrieved and kept for future use by this module is the list of SHA-1 of every certificate in the chain used to sign the terminal applications, on a per-application basis.

This module is in charge of answering to the following request: does the specified terminal application have access to an application identified by its AID on the specified secure element?

Here is a list of constraints that must be expressed by the rules:

- What is the list of authorities that are granted to access to all applications in the secure element?
- What is the list of authorities that are granted to access to a specific application identified by its AID in the secure element?
- What is the list of authorities that are granted to access to the default selected application of the secure element?

The Access Control Filtering

This module is activated when a terminal application requests access to an application in a secure element. This module must collect the information needed to identify the requesting terminal application, the targeted secure element and the AID of the targeted application within this secure element.

This information, once retrieved, is sent to the access control rules engine, that will determine if the access shall be granted or not.

If the access is granted, operations continue normally and the channel is opened (regular filtering of the Secure Element Access API still applies).

If the access is not granted, then an error condition is notified to the calling terminal application, indicating that it does not have access to the specified secure element application. It is also recommended that the terminal system is notified (by logging or any other means of this security-related event).

The Access Control Data

This access control data are stored in each Secure Element needing to protect the access to its applications.

If a Secure Element does not contain access control data, then no access control mechanism applies, and all the terminal applications can access all the secure element applications.

If a Secure Element contains an access control data structure, but this structure is empty, then access control mechanism applies and no terminal applications can access applications on this Secure Element.

The access control rules are structured in the following way:

A rule applies either to:

- The default selected application on the secure element
- An application on the Secure Element, identified by its AID (as per ISO7816-5)
- All the secure element applications: it then defines the behaviour for all the applications not mentioned in any other rule

A rule indicates the conditions to satisfy to be granted to access the targeted secure element application.

These conditions can be:

- Any terminal application can be granted access, whatever its signature
- No terminal application can be granted access
- A list of signing authorities are explicitly granted access

The list of signing authorities is made of a list of SHA-1 hashes of their respective certificates.

A terminal application is granted access if it is signed using a certificate chain with at least one of the certificate having its hash present in the list.

More conditions may be added in a future release (like a pattern matching to apply to APDU command headers).

The Access Control Main File

The access control main file is part of the access control data and must be used to store configuration parameters, as well as the indirection to the file containing the rules. The only parameter defined in the current version of this specification is the refresh tag.

The refresh tag is an 8-byte random octet string identifying the access control data set. This refresh tag must be randomly updated by OTA servers after any modification of the data structure (adding or removing of rules or conditions). Each time a new request to open a connection to a Secure Element application is done, the refresh tag shall be used by the device to verify if the data structure has been updated. If the value in the secure element is different than the value stored in the device cache, the device shall update the access control rules.

The access control, step by step

Step 1: the terminal application uses the secure element access API to open a communication channel with an application in a secure element.

Step 2: before any other action, this request is forwarded to the access control filtering mechanism.

Step 3: the access control filtering mechanism retrieves the certificate chain used to sign the calling application, and checks with the access control rules engine if one of the authorities in the chain has the rights to access to the secure element application.

Step 3bis: If there are no rules defined yet, or if the refresh tag has changed, the rules engine tries to fetch a new version of the rules data from the secure element hosting the secure element application being accessed. This new version, if available, becomes the new reference set of rules for this secure element. The purpose of the refresh tag in the secure element is to support the use case of an application being loaded, and the access control data being updated over the air, without the terminal being aware of it.

Step 4: the result of the rules combination is transmitted back to the filtering mechanism and then to the transport layer.

Step 5: if the access is granted, the connection with the secure element application is authorized and communication can happen between the terminal application and the secure element application.

Step 5bis: if the access is not granted, the connection is not established and the terminal application is informed that it cannot access the secure element application for security reasons.

Combination of the access control rules & rights

If there is no access control data in a secure element, access to this (and only this) secure element applications (including the default selected application) is always denied. If there are access control data in the secure element, but there are no rules in it, access is never granted. This means that the only way to give access to a Secure Element application is to define an explicit rule, such rule can be a specific rule (only applied to one application) or a generic rule.

The general rules (i.e. rules that are not explicitly referencing a secure element application) apply to all the secure element applications that are not explicitly referenced by another rule.

The specific rules (i.e. rules that are explicitly referencing a secure element application by its AID or the default application) are only applied for this application. As a general matter when considering the rules, the “default” application is seen as “a specific application with an unknown AID”.

If several rules exist that identify explicitly a secure element application (by its AID) or that are explicitly referencing the default selected application, then these rules are aggregated.

If several general rules exist then these rules are aggregated

The priority of a rule is NOT linked at all to the reading order of the rules when parsing the PKCS#15 file structure. The order of priority and precedence is the following one. This is a strict order and it shall be enforced by the device:

- Rules specifying an explicit AID or related to the default application (higher priority)
- General rules involving “other” applications (lowest priority)

This means that the device SHALL first look for rules that are applied to a specific AID and only look for generic rules if the no specific rule was found. If one or more specific rules are specified for a given AID, generic rules SHALL never be used for this AID.

As for rules, rights granted are also aggregated together with the exception of the value “NEVER”. Both for specific rules and generic rules, during the aggregation process, if the value “NEVER” is associated to a rule than the device SHALL forbid the access whatever the other values can be. Rules based on certificates and on the “ALWAYS” right are aggregated together.

The following table provides some combinations of the rules & rights and the expected result (“Rxx” means a rule defined in an ACCF file, “CER#y” means a certificate.):

“NOT USED” means that the AID or general rule is not used (i.e. is not referenced in the PKCS#15 provisioning)

SE App.#1	SE App.#2	Other AIDs (general rule)	Access result for the secure element applications
R01: CER#1	NOT USED	NO RULE	- Access to SE App. #1. is granted to device applications signed with CER#1 (and only CER#1) - Access to any other SE application is denied.
R01: CER#1 R02: CER#2	NOT USED	NOT USED	- Access to SE App. #1. is granted to device applications signed with CER#1 or with CER#2 - Access to any other SE application is denied.

R01: CER#1 R02: CER#2 R03: ALWAYS	NOT USED	NOT USED	- Access to SE App. #1. is granted to all device applications - Access to any other SE application is denied.
R01: CER#1 R02: CER#2 R03: NEVER	NOT USED	NOT USED	- Access to SE App. #1. is denied for all device applications - Access to any other SE application is denied.
R01: CER#1 R02: CER#2 R03: NEVER R04: ALWAYS	NOT USED	NOT USED	- Access to SE App. #1. is denied for all device applications - Access to any other SE application is denied.
R01: CER#1	R02: CER#2	NO RULE	- Access to SE App. #1. is granted to device applications signed with CER#1 - Access to SE App. #2. is granted to device applications signed with CER#2 - Access to any other SE application is denied.
R01: CER#1	NO RULE	R02: CER#2	- Access to SE App. #1. is granted to device applications signed with CER#1 - Access to SE App. #2 is denied for all device applications - Access to all other SE applications (except SE App. #1 and SE App. #2) is granted to all device applications signed with CER#2. Note: access to SE App. #2 is denied as the AID is specified but with no associated rule.
R01: NEVER	R02: ALWAYS	R03: CER#2	- Access to SE App. #1 is denied for all device applications. - Access to SE App. #2 is granted for all device applications - Access to all other SE applications (except SE App. #1) is granted to all device applications signed with CER#2.
R01: ALWAYS	NO RULE	R02: NEVER	- Access to SE App. #1 is granted for all device applications - Access to SE App. #2 is denied for all device applications - Access to any other SE application is denied.

R01: CER#1	NO RULE	R02: ALWAYS	- Access to SE App. #1 is granted for all device applications signed with CER#1 - Access to SE App. #2 is denied for all device applications - Access to any other SE application is granted for all device applications.
R01: ALWAYS	R02: ALWAYS	R03: ALWAYS	- Access to all SE application is granted for all device applications.
R01: CER#1	NOT USED	R02: NEVER	- Access to SE App. #1. is granted to device applications signed with CER#1 - Access to any other SE application is denied.
R01: CER#1	NO RULE	R02: CER#2 R03: ALWAYS	- Access to SE App. #1 is granted for all device applications signed with CER#1 - Access to SE App. #2 is denied for all device applications - Access to any other SE application is granted for all device applications.
R01: CER#1	NO RULE	R02: CER#2 R03: NEVER	- Access to SE App. #1 is granted for all device applications signed with CER#1 - Access to SE App. #2 is denied for all device applications - Access to any other SE application is denied for all device applications.

Format of the Secure Element Access Control data

The access control data are stored in PKCS#15 file structure in the secure element. The following sections define this structure.

File paths

All the paths used for the access control mechanism described in this specification (DODF, ACMF, and ACRF files) SHALL be relative paths from PKCS#15 directory. The full path starting from the UICC MF shall not be used.

File padding

If needed, the end of the files can be padded using 0xFF, and only 0xFF, bytes. The parsing engine used to manage the files involved in the access control mechanism shall support such padding.

PKCS#15 selection

Selection of the PKCS#15 file structure or application is not within the scope of this specification and can be managed in different ways by the devices.

However a recommended way to perform the selection of the PKCS#15 application is to use the following sequence:

Step 1: the device sends a SELECT_BY_NAME command with PKCS#15 AID (A0 00 00 00 63 50 4B 43 53 2D 31 35). If the select is successful, the device can start reading PKCS#15 files (ODF, DODF…)

Step 2: if the previous select fails, the device sends SELECT commands to select the MF and the EF DIR, and then reads the EF DIR in order to locate an entry with the PKCS#15 AID. If a matching entry is found, the device must select the PKCS#15 DF path, and then it can start reading PKCS#15 files (ODF, DODF…)

The PKCS#15 DODF

The PKCS#15 DODF used as the entry point to the access control data has an OidDO entry with an OID that must be in the Global Platform scope.

For information, GlobalPlatform OIDs are constructed as this: {iso(1) member-body(2) country-USA(840) Global-Platform(114283)} More info can be found at: http://www.oid-info.com/get/1.2.840.114283

The registered OID for this specification is: {iso(1) member-body(2) country-USA(840) Global-Platform(114283) device(200) seAccessControl(1) accessControlMainFile(1)}

The DODF OidDO contains a path to the Access Control Main File and the path structure is defined with the following ASN.1 syntax:

```
Path ::= SEQUENCE
  path OCTET STRING,
  index INTEGER (0..65535) OPTIONAL,
  length [0] INTEGER (0..65535) OPTIONAL
}( WITH COMPONENTS {..., index PRESENT, length PRESENT}|
WITH COMPONENTS {..., index ABSENT, length ABSENT})
-- the path of a file (as per PKCS#15)
```

The Access Control Main File (ACMF)

The Access Control Main File or ACMF (referenced from the DODF) has the following structure:

Identifier: 'xxxx'	Structure: transparent file		Mandatory
File length: n bytes	Update activity: low		
Access Conditions: READ ALW UPDATE ADM DEACTIVATE ADM ACTIVATE ADM			
Bytes	Description	M/O	Length
1 to n	AccessControlMainFile	M	n bytes

There shall be only one ACMF file per Secure Element. If a Secure Element is holding several ACMF files then the terminal shall consider that the security is compromised and shall forbid the access to all the secure element applications for, and only for, this specific Secure Element,

The AccessControlMainFile object is related to the secure element which contains it.

The AccessControlMainFile object contains the refresh tag and the path to the rules, but additional fields can be added in future versions of this specification.

The AccessControlMainFile object is defined using the following ASN.1 syntax:

```
-- The access control main file object
AccessControlMainFile ::= SEQUENCE {
  -- the refresh tag
  refreshTag OCTET STRING (SIZE(8)),
  -- the path to the access control rules file
  rulesFile Path,
  -- RFU
  ...
}
```

The Access Control Rules File (ACRF)

The rules are stored in the Access Control Rules File or ACRF (referenced from the ACMF), which has the following structure:

Identifier: 'xxxx'	Structure: transparent file		Mandatory
File length: n bytes	Update activity: low		
Access Conditions: READ ALW UPDATE ADM DEACTIVATE ADM ACTIVATE ADM			
Bytes	Description	M/O	Length
1 to n	List of Rules	M	n bytes

There shall be only one ACRF file per Secure Element. If a Secure Element is holding several ACRF files then the terminal shall consider that the security is compromised and shall forbid the access to all the secure element applications for this specific Secure Element,

The access to other applications running in another Secure Element shall be managed according to the security rules defined for this other Secure Element.

The Rules objects define an access rule for one or more secure element applications.

They identify explicitly or implicitly a set of applications, and refer to the Access Control Conditions Files that describe how these applications can be accessed.

The Rules objects are defined using the following ASN.1 syntax:

```
-- An access control rule entry
Rule ::= SEQUENCE {
  -- the target of this policy entry,
  target Target,
    -- the path to the access control conditions file applicable
  -- for this target
  conditionsFile  Path,
  -- RFU
  ...
}
```

The Target object indicates whether the rule applies to one secure element application (identified by its AID) or the default selected application, or all the applications (i.e.; all the applications that are not explicitly protected by a specific rule).

It is defined using the following ASN.1 syntax:

```
-- An access control target: either a named application, the default selected application or all other applications,
Target ::= CHOICE {
  -- the AID of the targeted secure element application
  aid [0]AID,
  -- the (unnamed) default selected secure element application
  default [1]NULL,
    -- indentifies all other the applications,
  -- that are not referenced in another rule
  others [2]NULL,
  -- RFU
  ...
}
```

The AID object uses the following ASN.1 syntax:

```
-- as per ISO7816-5
AID ::= OCTET STRING
```

The Access Control Conditions File (ACCF)

The conditions are stored in the Access Control Conditions File or ACCF (referenced by each Rule).

The conditions are expressed as a list of entries, each entry containing a SHA-1 of a certificate identifying an authority that is granted access.

If this file is empty, it means that rules pointing to this file are denying access to any terminal application.

If this file contains a condition without a certificate hash, then rules pointing to this file are granting access to any terminal application.

Identifier: 'xxxx'	Structure: transparent file	Mandatory
File length: n bytes	Update activity: low	
Access Conditions: READ ALW UPDATE ADM DEACTIVATE ADM ACTIVATE ADM		

Bytes	Description	M/O	Length
1 to n	List of Conditions	M	n bytes

The Conditions have the following ASN.1 syntax:

```
-- A Condition entry
Condition ::= SEQUENCE {
  -- the hash of the certificate of the authorized entity;
  -- if not indicated, then the Rule pointing to this Condition
  -- applies to all the terminal applications
  cert  CertHash OPTIONAL,
  -- RFU
  ...
}
–- SHA1 of the certificate of the authority being granted access
CertHash   ::= OCTET STRING (SIZE(20))
```

PKCS#15 Parsing

When the device is checking if the access control mechanism shall be applied to a given secure element, different cases may occur:

1) The access control mechanism is explicitly provisioned in the secure element

2) There is enough information available in the secure element to be sure that the secure element issuer intention was to ship a secure element without this mechanism.

3) Some information is available but not enough to know if it is an invalid provisioning of the secure element or if the intention was to ship the secure element without the security mechanism. In this last case, the device SHALL consider that the mechanism is activated and restrict the access.

Please remind that in both case 2) and case 3), the result will be the same: access to all applications of the secure element will be denied for all device applications.

The secure element has an explicit access control mechanism (as described in this specification) when both following conditions are satisfied:

5. There is a PKCS#15 application (selectable using the standard AID) in the secure element **OR**

There is a PKCS#15 application/file structure referenced in the secure element EFdir (in case of UICC)

6. The OID of the access control mechanism is specified in the DODF provisioning file

In the same way, the secure element has no access control mechanism (as described in this specification) when one of the following conditions is satisfied:

1. There is no PKCS#15 application (selectable using the standard AID) in the secure element And

There is no EFdir file in the UICC

2. There is no PKCS#15 application (selectable using the standard AID) in the secure element **AND**

There is no PKCS#15 application/file structure referenced in the UICC EFdir

3. There is a PKCS#15 provisioning in the secure element (either an application or a file structure) **AND**

The OID of the access control mechanism is not referenced in a valid DODF file.

All other cases shall be considered as an implicit use of the access control mechanism and if there is an error in the data stored then the device SHALL denied any access to this secure element applications. Such cases include for example:

There is a PKCS#15 provisioning but the ODF file is not correctly formatted (invalid file content) or not present, or any other error that may occur when selecting/reading the content of this file.

There is a PKCS#15 provisioning but the DODF file is not correctly formatted (invalid file content) or not present, or any other error that may occur when selecting/reading the content of this file.

There is an EFdir file (for UICC case) but the file cannot be read or the provisioning of the file is not valid (i.e. the content is not formatted according to the ETSI TS 102 221 specification).

Other errors may occur when parsing the access control mechanism structure (ACMF, ACRF, ACCF files) after the reading of the OID in the DODF:

If the ACMF file cannot be found or cannot be read or its content is invalid then the device SHALL denied any access to this secure element applications.

If the ACRF file cannot be found or cannot be read or its content is invalid then the device SHALL denied any access to this secure element applications.

If one of ACCF file referenced in the ACRF file cannot be found or cannot be read or its content is invalid:

If the rule is targeting all secure element applications then the device SHALL denied any access to this secure element applications whatever the other rules are.

If the rule is targeting the default application of the secure element then the device SHALL denied any access to this default applications whatever the other rules are.

If the rule is targeting a specific application (AID) of the secure element then the device SHALL denied any access to this specific application (AID) whatever the other rules are.

Example of access control data

For the sake of simplicity, in the following examples, AIDs are all in the form of A0 00 00 01 51 xx, and the certificate hash are fake values like 111…, 222…, 333…, etc.

First example

In this first example, we have the following setup:

AID1 = A0 00 00 01 51 01 → access denied for all apps → conditions 1

AID2 = A0 00 00 01 51 02 → access allowed for 1 app (hash1) → conditions 2

AID3 = A0 00 00 01 51 03 → access allowed for 1 app (hash1) → conditions 2

Any other AIDs → access allowed for all apps → conditions 3

Here's a summary of the PKCS#15 file system personalization:

Hierarchical view (file system):

MF (3F00)

|-EF DIR (2F00) --> reference DF PKCS-15

|

|-DF PKCS-15 (7F50)

|-ODF (5031) --> reference DODF

|-DODF (5207) --> reference EF ACMain

|-EF ACMain (4200) --> reference EF ACRules

|-EF ACRules (4300) --> reference EF ACConditions...

|-EF ACConditions1 (4310)

|-EF ACConditions2 (4311)

|-EF ACConditions3 (4312)

Hierarchical view (applet):

Applet (AID: A0 00 00 00 63 50 4B 43 53 2D 31 35)

|-ODF (5031) --> reference DODF

|-DODF (5207) --> reference EF ACMain

|-EF ACMain (4200) --> reference EF ACRules

|-EF ACRules (4300) --> reference EF ACConditions...

|-EF ACConditions1 (4310)

|-EF ACConditions2 (4311)

|-EF ACConditions3 (4312)

EF DIR: 3F00/2F00

Based on this ASN.1 syntax:

```
DIRRecord ::= [APPLICATION 1] SEQUENCE {
        aid [APPLICATION 15] OCTET STRING,
        label [APPLICATION 16] UTF8String OPTIONAL,
        path [APPLICATION 17] OCTET STRING,
        ddo [APPLICATION 19] DDO OPTIONAL
}
```

aid PKCS-15 = A0 00 00 00 63 50 4B 43 53 2D 31 35

label = "PROVISIONING" = 50 52 4F 56 49 53 49 4F 4E 49 4E 47

path = 3F00/7F50

binary coding:

61 22 4F 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 50 0C 50 52 4F 56 49 53 49 4F 4E 49 4E 47 51 04 3F 00 7F 50

ODF: 3F00/7F50/5031

References file 5207.

Binary coding

A7 06 30 04 04 02 52 07

DODF: 3F00/7F50/5207

OID GlobalPlatform ::= {iso(1) member-body(2) country-USA(840) Global-Platform(114283) device(200) seAccessControl(1) accessControlMainFile(1)}

http://www.oid-info.com/get/1.2.840.114283

==> HEX encoding = 2A 86 48 86 FC 6B 81 48 01 01

application name = "GP SE Acc Ctl" (example: value to be confirmed)

path to EF ACMain = 4200

binary coding:

A1 29 30 00 30 0F 0C 0D 47 50 20 53 45 20 41 63 63 20 43 74 6C A1 14 30 12 06 0A 2A 86 48 86 FC 6B 81 48 01 01 30 04 04 02 42 00

EF ACMain: 3F00/7F50/4200

Refresh tag value is 01 02 03 04 05 06 07 08

path to EF ACRules = 4300

binary coding:

30 10 04 08 01 02 03 04 05 06 07 08 30 04 04 02 43 00

EF ACRules: 3F00/7F50/4300

AID1 --> EFConditions 4310 --> access denied for all apps

AID2 --> EFConditions 4311 --> access allowed for 1 app (hash1)

AID3 --> EFConditions 4311 --> access allowed for 1 app (hash1)

* --> EFConditions 4312 --> access allowed for all apps

binary coding:

30 10 A0 08 04 06 A0 00 00 01 51 01 30 04 04 02 43 10

30 10 A0 08 04 06 A0 00 00 01 51 02 30 04 04 02 43 11

30 10 A0 08 04 06 A0 00 00 01 51 03 30 04 04 02 43 11

30 08 82 00 30 04 04 02 43 12

EF ACConditions1: 3F00/7F50/4310 (access denied for all apps)

binary coding:

(empty file)

EF ACConditions2: 3F00/7F50/4311 (access allowed for 1 app)

Hash1 has the value 111…

binary coding:

30 16 04 14 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11

EF ACConditions3: 3F00/7F50/4312 (access allowed for all apps)

binary coding:

30 00

Second example

In this second example, the setup is:

AID1 = A0 00 00 01 51 01	→ access allowed for all apps	→ conditions 1
AID2 = A0 00 00 01 51 02	→ access allowed for 1 app (hash1)	→ conditions 2
AID3 = A0 00 00 01 51 03	→ access allowed for 3 apps (h1, h2, h3))	→ conditions 3
AID4 = A0 00 00 01 51 04	→ access denied for all apps	→ conditions 4

AID5 = A0 00 00 01 51 05 → access denied for all apps → conditions 4

Any other AIDs → access denied for all apps → conditions 4

Here's a summary of the PKCS#15 file system personalization:

Hierarchical view (file system):

MF (3F00)
|-EF DIR (2F00) --> reference DF PKCS-15
|
|-DF PKCS-15 (7F50)
  |-ODF (5031) --> reference DODF
  |-DODF (5207) --> reference EF ACMain
  |-EF ACMain (4200) --> reference EF ACRules
  |-EF ACRules (4300) --> reference EF ACConditions...
  |-EF ACConditions1 (4310)
  |-EF ACConditions2 (4311)
  |-EF ACConditions3 (4312)
  |-EF ACConditions4 (4313)

Hierarchical view (applet):

Applet (AID: A0 00 00 00 63 50 4B 43 53 2D 31 35)
  |-ODF (5031) --> reference DODF
  |-DODF (5207) --> reference EF ACMain
  |-EF ACMain (4200) --> reference EF ACRules
  |-EF ACRules (4300) --> reference EF ACConditions...
  |-EF ACConditions1 (4310)
  |-EF ACConditions2 (4311)
  |-EF ACConditions3 (4312)
  |-EF ACConditions4 (4313)

EF DIR: 3F00/2F00

Based on this ASN.1 syntax:

```
DIRRecord ::= [APPLICATION 1] SEQUENCE {
        aid [APPLICATION 15] OCTET STRING,
        label [APPLICATION 16] UTF8String OPTIONAL,
        path [APPLICATION 17] OCTET STRING,
        ddo [APPLICATION 19] DDO OPTIONAL
}
```

aid PKCS-15 = A0 00 00 00 63 50 4B 43 53 2D 31 35

label = "PROVISIONING" = 50 52 4F 56 49 53 49 4F 4E 49 4E 47

path = 3F00/7F50

binary coding:

61 22 4F 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 50 0C 50 52 4F 56 49 53 49 4F 4E 49 4E 47 51 04 3F 00 7F 50

ODF: 3F00/7F50/5031

References file 5207.

binary coding:

A7 06 30 04 04 02 52 07

DODF: 3F00/7F50/5207

OID GlobalPlatform ::= {iso(1) member-body(2) country-USA(840) Global-Platform(114283) device(200) seAccessControl(1) accessControlMainFile(1)}

http://www.oid-info.com/get/1.2.840.114283

==> HEX encoding = 2A 86 48 86 FC 6B 81 48 01 01

application name = "GP SE Acc Ctl" (example: value to be confirmed)

path to EF ACMain = 4200

binary coding:

A1 29 30 00 30 0F 0C 0D 47 50 20 53 45 20 41 63 63 20 43 74 6C A1 14 30 12 06 0A 2A 86 48 86 FC 6B 81 48 01 01 30 04 04 02 42 00

EF ACMain: 3F00/7F50/4200

Refresh tag value is 01 02 03 04 05 06 07 08

path to EF ACRules = 4300

binary coding:

30 10 04 08 01 02 03 04 05 06 07 08 30 04 04 02 43 00

EF ACRules: 3F00/7F50/4300

AID1 --> EFConditions 4310 --> access allowed for all apps

AID2 --> EFConditions 4311 --> access allowed for 1 app (h1)

AID3 --> EFConditions 4312 --> access allowed for 3 apps (h1, h2, h3)

AID4 --> EFConditions 4313 --> access denied for all apps

AID5 --> EFConditions 4313 --> access denied for all apps

* --> EFConditions 4313 --> access denied for all apps

binary coding:

30 10 A0 08 04 06 A0 00 00 01 51 01 30 04 04 02 43 10

30 10 A0 08 04 06 A0 00 00 01 51 02 30 04 04 02 43 11

30 10 A0 08 04 06 A0 00 00 01 51 03 30 04 04 02 43 12

30 10 A0 08 04 06 A0 00 00 01 51 04 30 04 04 02 43 13

30 10 A0 08 04 06 A0 00 00 01 51 05 30 04 04 02 43 13

30 08 82 00 30 04 04 02 43 13

EF ACConditions: 3F00/7F50/4310 (access allowed for all apps)

binary coding:

30 00

EF ACConditions: 3F00/7F50/4311 (access allowed for 1 app)

binary coding:

30 16 04 14 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11

EF ACConditions: 3F00/7F50/4312 (access allowed for 3 apps)

binary coding:

30 16 04 14 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11

30 16 04 14 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22

30 16 04 14 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33

EF ACConditions: 3F00/7F50/4313 (access denied for all apps)

binary coding:

(empty file)

Annex 7: Other Elective Requirements:

The requirements within this Annex are mandated by those MNOs listed below in Annex 7.2.

7.1 Requirements

Battery Modes

NFC_REQ_7.1	At the end of the standby time, when the mobile device is automatically switched off, the mobile device SHALL have the capability to supply a source of power to perform a few transactions (15 transactions) in card emulation mode for at least 24 hours
NFC_REQ_7.2	NFC transactions SHALL be possible either in powered by the field mode (battery off) or battery low mode.

Note: This is important for public transport services.

Multiple Secure Elements (UICC SE and others)

MultiSE_REQ_7.3	The mobile device SHOULD support UICC SE only.

Notes

Felica support

TAG Type 3 pertains to Felica which is not required.

7.2 MNOs electing to support this Annex

This Annex applies to the following National MNO Companies.

Note: this list is under construction.

Telecom Italia

France Telecom

Deutsche Telekom AG and subsidiaries

Telefónica Ireland

Telefónica UK

Telefónica Germany

Telefónica Czech Republic

Telefónica Slovakia

Telefónica Spain

Telefónica Mexico

Telefónica Nicaragua

Telefónica Guatemala
Telefónica Panama
Telefónica Costa Rica
Telefónica El Salvador
Telefónica Colombia
Telefónica Venezuela
Telefónica Brasil
Telefónica Ecuador
Telefónica Peru
Telefónica Chile
Telefónica Uruguay
Telefónica Argentina
Vodafone Spain
Vodafone Netherlands
Vodafone Germany
Vodafone Australia
Vodafone Czech Republic
Vodafone Greece
Vodafone India
Vodafone Ireland
Vodafone Italy
Vodafone Hungary
Vodafone Malta
Vodafone Portugal
Vodafone Qatar
Vodafone Romania
Vodafone Albania
Vodafone Turkey
Vodafone UK
Vodafone New Zealand
Vodafone Ghana
Vodafone Egypt
Vodafone South Africa
KPN
Optus
Turkcell
Telenor

Document Management

Document History

Ver	Date	Brief Description of Change
1.0	July 2011	First draft for review
2.0	November 2011	Second draft incorporating feedback received to date

It is our intention to provide a quality product for your use. If you find any errors or omissions, please contact us with your comments. You may notify us at prd@gsm.org
Your comments or suggestions & questions are always welcome.