



NFC Handset APIs Requirement Specification

Version 4.1

November 2013

This is a Non-binding Permanent Reference Document of the GSMA

Security Classification: Non-confidential

Access to and distribution of this document is restricted to the persons permitted by the security classification. This document is confidential to the Association and is subject to copyright protection. This document is to be used only for the purposes for which it has been supplied and information contained in it must not be disclosed or in any other way made available, in whole or in part, to persons other than those permitted under the security classification without the prior written approval of the Association.

Copyright Notice

Copyright © 2013 GSM Association

Disclaimer

The GSM Association ("Association") makes no representation, warranty or undertaking (express or implied) with respect to and does not accept any responsibility for, and hereby disclaims liability for the accuracy or completeness or timeliness of the information contained in this document. The information contained in this document may be subject to change without prior notice.

Antitrust Notice

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

Table of Contents

1	Introduction	3
1.1	Overview	3
1.2	Scope	4
1.3	Use Cases/Services	4
1.4	Definition of Terms	4
2	References	5
3	Terminology	6
4	Generic Device Architecture	6
4.1	Dual Application architecture	6
4.2	Security	7
4.3	Generic Device APIs	8
4.4	Mobile Wallet	8
4.5	Security	8
4.6	Introduction	9
4.7	NFC Architecture	9
4.8	Access to Secure Elements	10
4.9	Handling NFC Controller, Card emulation mode and Secure Elements	10
4.10	Handling “EVT Transaction” events	12
4.10.1	Securing transaction events	13
4.10.2	Receiving transaction events	14
4.11	Protecting APIs with Operator Certificate	15
5	JavaME	16
5.1	Introduction	16
5.2	Key APIs	17
5.3	Security	17
5.4	API requirements	18
5.5	Delta to Generic Architecture	18
6	Key NFC requirements	18
6.1	NFC controller Required Features	18
6.1.1	NFC controller support for ETSI interfaces with the UICC	19
6.2	Mobile Device Modem Support	19
6.3	Mobile Device APIs	19
6.4	Mobile Device APN management	20
6.5	UI application triggering	20
6.6	Remote Management of NFC services (Access to UICC in connected mode)	20
6.7	NFC Controller TAG Support	20
6.8	Multiple SE support (UICC SE and others)	21
6.9	SCWS support	21
6.10	Access Control	21
Annex A	CAT Letter Classes Support	23
Annex B	UICC CLF Interface, Supported Options as per ETSI TS 102 613	24
Annex C	Other Elective Requirements	25
C.1	Requirments	25
C.1.1	Battery Modes	25
C.1.2	Multiple Secure Elements (UICC SE and others)	25
Document Management		26
	Document History	26
	Other Information	26

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 with '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 Annex D, including the operators who support this 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 & Original Device Manufacturers (OEMs and ODMs) and defines at a high level the application architecture and lower layer enablers, required to fulfil the *secured* NFC use cases. It further expands upon this, by detailing the particular mobile device Application Programming Interfaces (APIs) per OS to enable a *secured* service use case and the atomic requirements necessary to fulfil the NFC enabler 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 primarily targets the *secured* service use cases, and can provide for the following propositions, 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', (Mobile Network Operators' (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 (V9.5.0) Universal Subscriber Identity Module (USIM) Application Toolkit (USAT) 3GPP TS 31.116 (V9.3.0) Remote APDU Structure for (Universal) Subscriber Identity Module (U)SIM Toolkit applications
ETSI SCP Specifications	ETSI TS 102 613 (V9.2.0): UICC – Contactless Front End (Physical and data link layer characteristic) ETSI TS 102 622 (V9.4.0): UICC – Contactless Front End, (HCI) ETSI TS 102 223 (V9.3.0): Card Application Toolkit ETSI TS 102 226 (V.9.3.1): 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 v2.02
GSMA Requirements	GSMA Requirements for SWP NFC handsets V 4.0

GSMA Requirements	NFC UICC Requirement Specification, Version 4.0
AFSCM Specification	090929 - AFSCM TECH - PROC - NFC Mobile Handset High Level Requirements - V3.2
Global Platform	Secure Element Access Control

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.

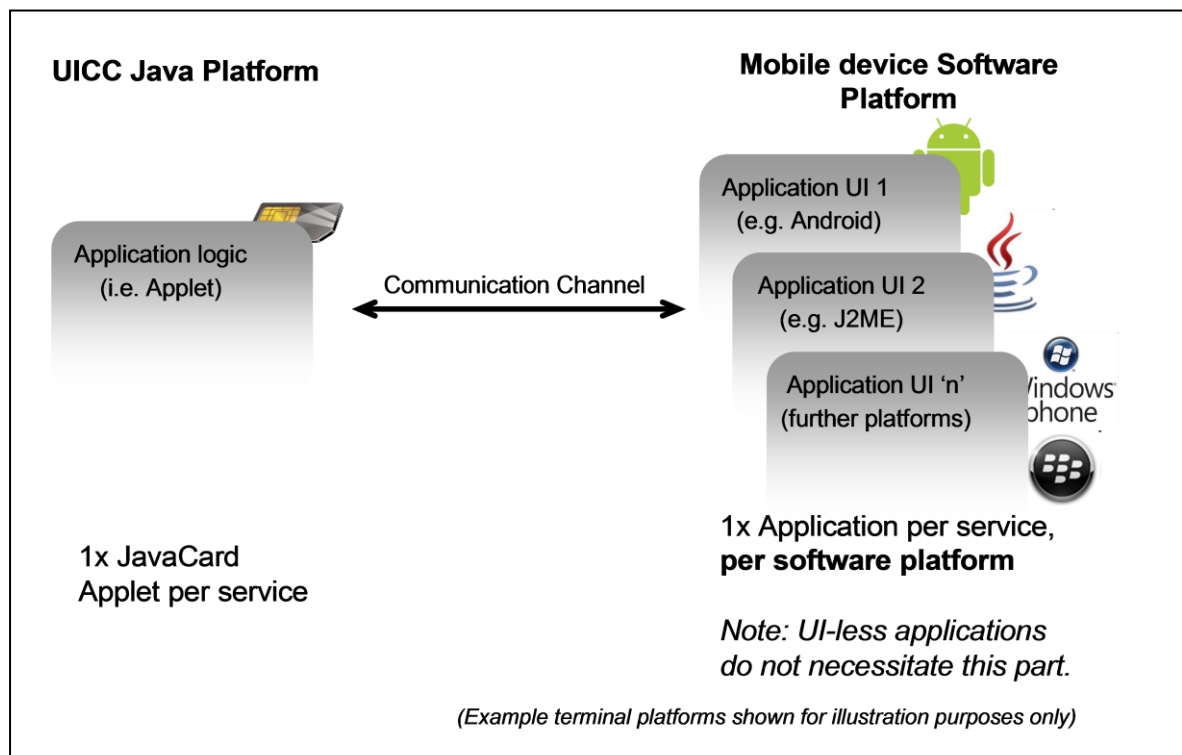


Figure 1: Dual application architecture

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).

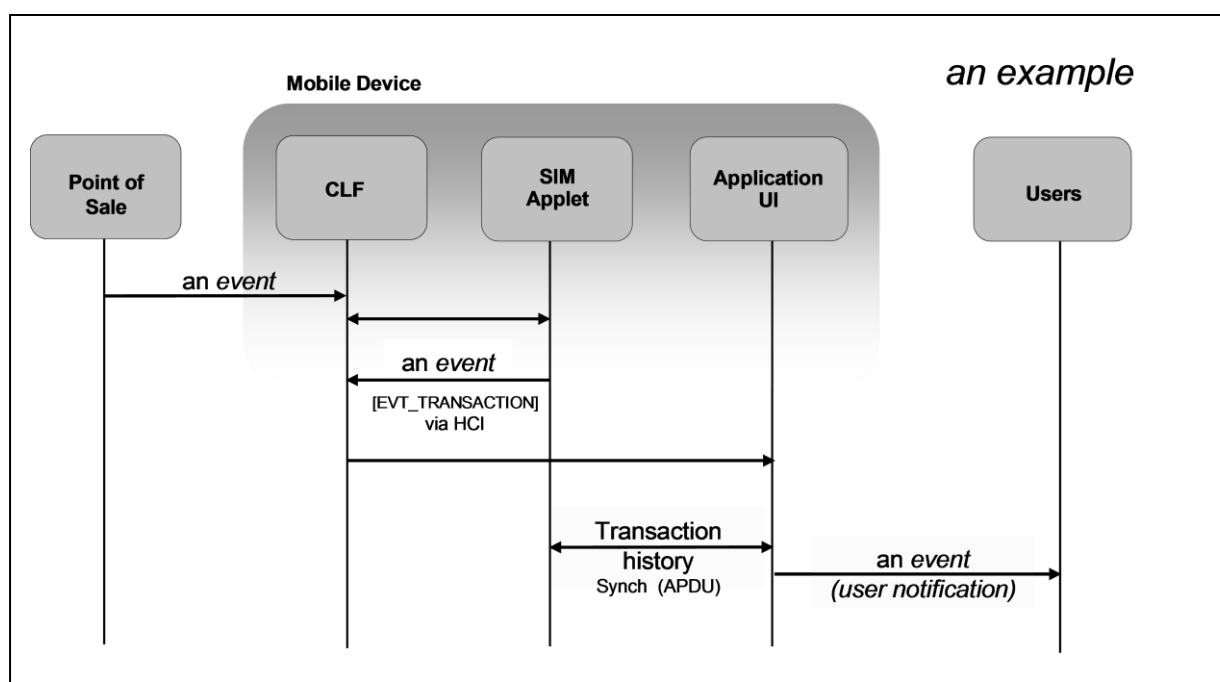


Figure 2: NFC Event routing, an example, for card-emulation mode

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 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 Element), and provide a controlled access path to relevant parts of its internal memory.

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, which delivers key use cases for NFC.

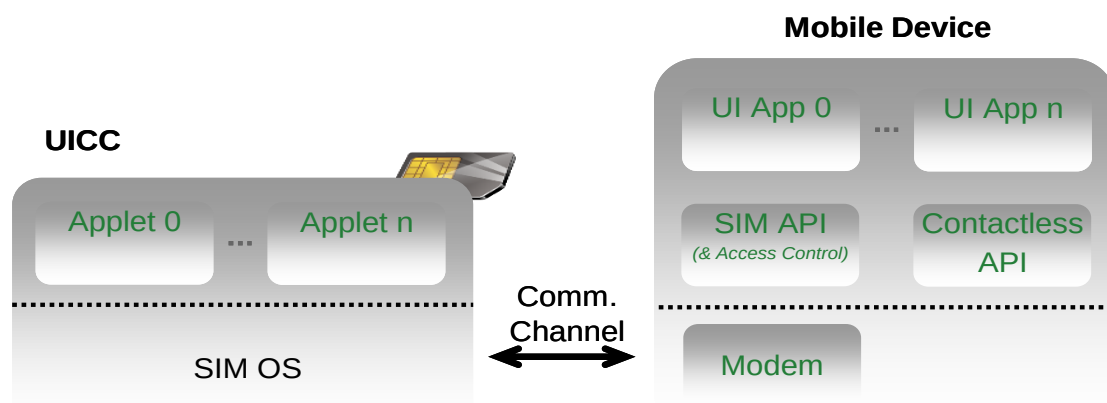


Figure 3: Mobile Device API generic software stack

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 and interoperability of MNO and SP services.

- Key APIs necessary to facilitate NFC use cases include, handling of multiple Secure Elements, the state of the NFC Controller,
- and state of Card Emulation mode.

Note: Further information can be obtained from the *NFC UICC requirement specification* With an appropriate security mechanism to protect and enforce against un-authorised usage.

4.4 Mobile Wallet

The *Mobile Wallet* is intended to facilitate the user experience, and allow the MNO or SP to optionally 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

4.5 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 device communicates 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.

4.6 Introduction

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

4.7 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.

The SIM Alliance group has published the “Open Mobile API” specification. And from this document, any mobile device manufacturer will be able to provide standardised APIs for access to the different Secure Elements such as the UICC SE.

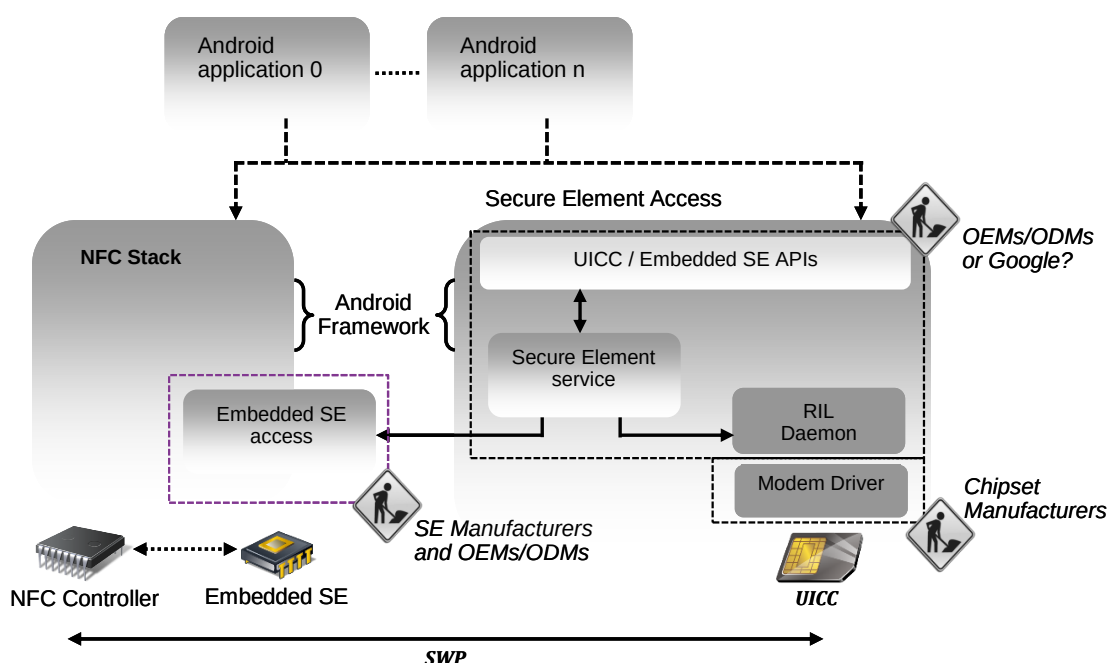


Figure 4: Android NFC software stack

Figure 4 gives an overview of an 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.

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

4.8 Access to Secure Elements

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 a SEService is connected to the device.

A complete description of the different functions is available in the document 'SIMalliance Open Mobile API Specification'.

4.9 Handling NFC Controller, Card emulation mode and Secure Elements

This section details the additional classes to be implemented within the Android Framework, in order to provide new features linked to the handling of the NFC Controller, Card Emulation mode and multiple Secure Elements.

- Framework shall create only one instance of the following classes (singleton) and handles must be only returned through dedicated methods.
- "com.gsma.services.nfc.jar" shall contain the different APIs.

public class

NfcController

com.gsma.services.nfc.NfcController

Nested Classes

Interface [NfcController.Callbacks](#)

Constants

public static final int *CARD_EMULATION_DISABLED*

Card Emulation Mode has been disabled.

Constant Value: 0 (0x00000000)

public static final int *CARD_EMULATION_ENABLED*

Card Emulation mode has been enabled.

Constant Value: 1 (0x00000001)

public static final int *CARD_EMULATION_ERROR*

An error occurred when handset tried to enable/disable "Card Emulation" mode.

Constant Value: 100 (0x00000100)

Public Methods

public static NfcController **getDefaultController** (Context context)

Helper to return the instance of the NFC Controller.

Parameters

context The calling application's context

Returns

The instance of the NFC Controller or `null` if NFC is not available.

public boolean **isEnabled** ()

Returns NFC Controller's state.

Returns

`true` if the NFC adapter is enabled; `false` otherwise.

public void **enableNfcController** (*NfcController.Callbacks* cb)

Asks the system to enable the NFC Controller and registers a callback function to run when the activation is finished. If the device does not support NFC, the callback will be called immediately.

Throws

SecurityException If the application is not allowed to use this API.

Application must be signed with system certificates or certificates from UICC to call this method.

public boolean **isCardEmulationEnabled** ()

Returns Card Emulation Mode's state

Returns

`true` if the Card Emulation Mode is enabled; `false` otherwise

public void **enableCardEmulationMode** (*NfcController.Callbacks* cb)

Asks the system to enable the "Card Emulation" mode and registers a callback function to run when the process is finished. If the device does not support NFC, the callback will be called immediately.

Throws

IllegalStateException If NFC adapter is not enabled

SecurityException If the application is not allowed to use this API.

When UICC is the active SE, only applications signed with certificates from UICC are granted to call this method. When eSE is the active SE, only applications signed with system certificates are granted to call this method.

public void **disableCardEmulationMode** (*NfcController.Callbacks* cb)

Asks the system to disable the "Card Emulation" mode and registers a callback function to run when the process is finished. If the device does not support NFC, the callback will be called immediately.

Throws

SecurityException If the application is not allowed to use this API.

When UICC is the active SE, only applications signed with certificates from UICC are granted to call this method. When eSE is the active SE, only applications signed with system certificates are granted to call this method.

public static interface

NfcController.Callbacks

`com.gsma.services.nfc.NfcController.Callbacks`

Public Methods

public abstract void **onNfcController** (boolean success)

Called when process for enabling the NFC Controller is finished

Parameters

success `true` if the NFC adapter is enabled; `false` otherwise

public abstract void **onCardEmulationMode** (int status)

Called when process for enabling/disabling “Card Emulation Mode” is finished

Parameters

status - `CARD_EMULATION_ENABLED`

when “Card Emulation” mode has been enabled.

- `CARD_EMULATION_DISABLED`

when “Card Emulation” mode has been disabled.

- `CARD_EMULATION_ERROR`

if a problem occurred during the enabling/disabling process

public class

SEController

com.gsma.services.nfc.SEController

Public Methods

public static SEController **getDefaultController** (Context context)

Helper to return the instance of the SE Controller

Parameters

context The calling application’s context

Returns

The instance of the SE Controller

public String **getActiveSecureElement** ()

Returns the name of the active Secure Element

Returns

Name of the active Secure Element

public void **setActiveSecureElement** (String SEName)

Sets a specified Secure Element as the active one

Parameters

SEName User-friendly name of the Secure Element

Throws

IllegalStateException If the NFC adapter is not enabled.

SecurityException If application is not allowed to use this API.

Application must be signed with system certificates or certificates from UICC to call this method.

4.10 Handling “EVT Transaction” events

When a transaction has been executed by an applet, it may need to inform application layer. To do this, applet may trigger an event known as “EVT_TRANSACTION”. This section intends to standardise the construct of this event message.

Action	<i>com.gsma.services.nfc.action.TRANSACTION_EVENT</i>
Mime type	-
URI	<i>nfc://secure:0/<SEName>/<AID></i> - <i>SEName</i> reflects the originating SE It must be compliant with SIMAlliance Open Mobile APIs - <i>AID</i> reflects the originating UICC applet identifier

Table 1: EVT Transaction

- When *AID* is omitted from the URI, application component are registered to any “EVT_TRANSACTION” events sent from the specified Secure Element.
- For UICC, *SEName* must follow the format hereafter: “*SIM*” [*smartcard slot*] (e.g. SIM1, SIM2, ... SIMn)

Transaction event data must be set in the following extended field.

com.gsma.services.nfc.extra.AID <i>ByteArray</i>	Contains the card “Application Identifier” <i>[optional]</i>
com.gsma.services.nfc.extra.DATA <i>ByteArray</i>	Payload conveyed by the HCI event

Table 2: Transaction event data

4.10.1 Securing transaction events

“EVT_TRANSACTION” messages are sensitive data. Intercepting these events might help a malicious application to lure a user into entering sensitive information into a fake UI.

Usage of “EVT_TRANSACTION” events must request the following permissions. All of them must be set to “dangerous” protection level.

NFC Controller	android.permission.NFC
Transaction Event	com.gsma.services.nfc.permission.TRANSACTION_EVENT

Table 3: Transaction event

Transaction intents link an Android application and an applet installed on a Secure Element. For this reason, securing them shall be done with the same rules that restrict applet access by the Android application through the SIMAlliance Open Mobile APIs.

If an application is registered to any “EVT_TRANSACTION”, by omitting the AID in the Intent, it will receive the events of any applets to those accessible using the “Access Control”.

The NFC stack shall therefore use internal “*Access Control*” APIs to check that the recipient activity has been signed with an authorised certificate. This check is performed at the time the event is being forwarded from the lower layers to the target application. If no application is authorised as per “*Access Control*” check, then the event is discarded.

4.10.2 Receiving transaction events

- Unicast - Reception by only one activity

By default, each time an “EVT_TRANSACTION” event is sent, it must be received by only one activity. Then, when several activities have registered their interest to receive a same event (based on an identical URI), implementation has to select one following the priority scheme described below.

1. android:priority” level defined in the applications Intent Filters shall be compared to establish the highest priority.
2. If several activities are subscribed to the same event with the same priority, the first installed has a higher priority.

In addition to the priority scheme, mobile device shall also implement the following methods allowing foreground applications to intercept the intent and claim priority over other activities that handle the same intent.

public class

SEController

com.gsma.services.nfc.SEController**Public Methods**

public void **enableEvt_TransactionFgDispatch** (*Activity* activity, *IntentFilter[]* filters)

Enables “EVT_TRANSACTION” foreground dispatch to the given Activity.
This will give the priority to the foreground activity when dispatching transaction events.

If any IntentFilters are provided to this method, they are used to match dispatch Intents for “com.gsma.services.nfc.action.TRANSACTION_EVENT”. If *null* is passed as a “filters” parameter, foreground activity will receive all the transaction events granted by the “*Access Control*”.

This method must be called from the main thread, and only when the activity is in the foreground (resumed). Before the completion of their “onPause()”, activities must call “disableEvt_TransactionFgDispatch(…)” callback to disable foreground dispatch after it has been enabled.

Parameters

<i>Activity</i>	The Activity to dispatch to.
<i>Filters</i>	The IntentFilters to override dispatching for, or null to always dispatch.

Throws

<i>IllegalStateException</i>	If the Activity is not currently in the foreground.
<i>SecurityException</i>	If application is not allowed to use this API.

When UICC is the current SE, only applications signed with certificates from UICC are granted to call this method. When eSE is the current SE, only applications signed with system certificates are granted to call this method.

public void **disableEvt_TransactionFgDispatch** ([Activity](#) activity)

Disables “EVT_TRANSACTION” foreground dispatch to the given Activity.

After calling “enableEvt_TransactionFgDispatch (...)”, an activity must call this method before its “onPause ()” callback completes.

This method must be called from the main thread.

Parameters

[Activity](#) The Activity to disable dispatch to.

Throws

[IllegalStateException](#) If the Activity has already been paused.
[SecurityException](#) If application is not allowed to use this API.

When UICC is the current SE, only applications signed with certificates from UICC are granted to call this method. When eSE is the current SE, only applications signed with system certificates are granted to call this method.

For any “unicast” reception, framework shall use “startActivity (...)” methods to inform the selected activity of the event.

Multicast - Reception by several application components (activity, services...)

When specific use cases request several registered/granted application components have to be notified (multicast), default behaviour can be changed by invoking the following API.

public class

SEController

com.gsma.services.nfc.SEController

Public Methods

public void **enableMultiEvt_transactionReception** (String SENAME)

After a call to this method, any authorized/registered components can receive a transaction event. This call shall not imply a power cycle of the mobile device, the modem or the UICC.

Throws

[SecurityException](#) If application is not allowed to use this API

When UICC is specified as SE, only applications signed with certificates from UICC are granted to call this method. When eSE is specified as SE, only applications signed with system certificates are granted to call this method.

After enabling the “multicast” reception, framework shall use “Broadcast Receivers” for alerting about the “EVT_TRANSACTION” events.

4.11 Protecting APIs with Operator Certificate

In order to protect the usage of these *sensitive* APIs, the mechanism for granting access to them, shall be based on certificates stored within the UICC as illustrated below in Fig 5.

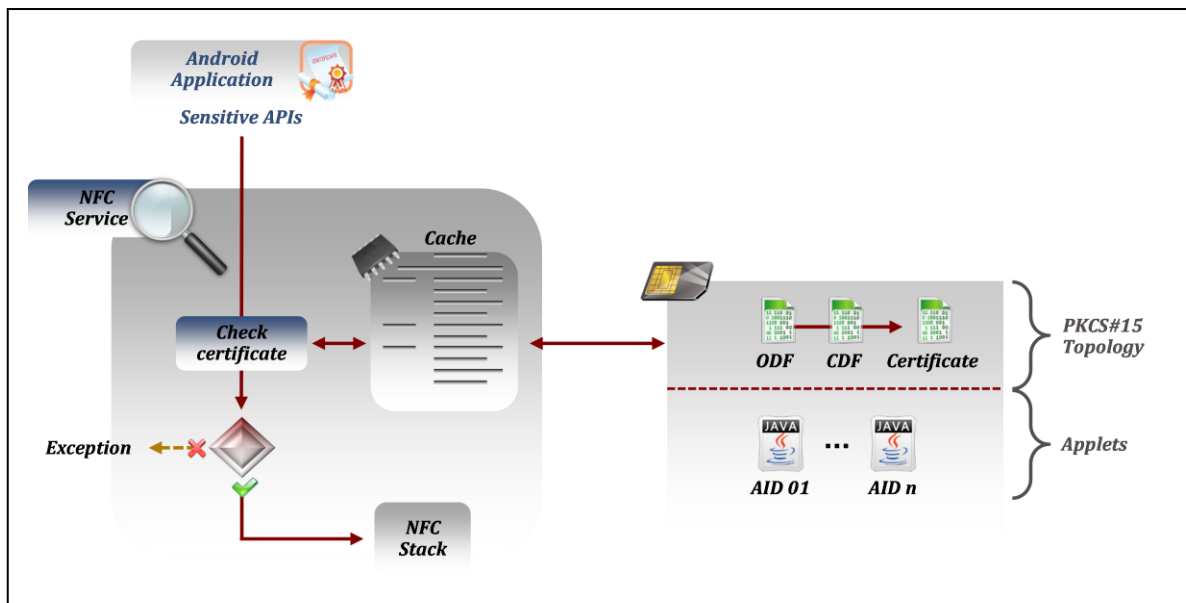


Figure 5: Protecting sensitive APIs with an Operator Certificate

As defined in the “PKCS #15 v 1.1: Cryptographic Token Information Syntax” specification from RSA, the certificates can be retrieved from the PKCS#15 structure where the EF_ODF points to elementary files that can be regarded as directories of certificates: “Certificate Directory Files” (CDFs).

Any sensitive API shall check if the application has been signed with one of the certificates stored in the UICC before executing the operation, and:

- Only authorised applications shall be granted access to this function.
- For any other non-authorised applications an exception needs to be raised.

5 JavaME

5.1 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.

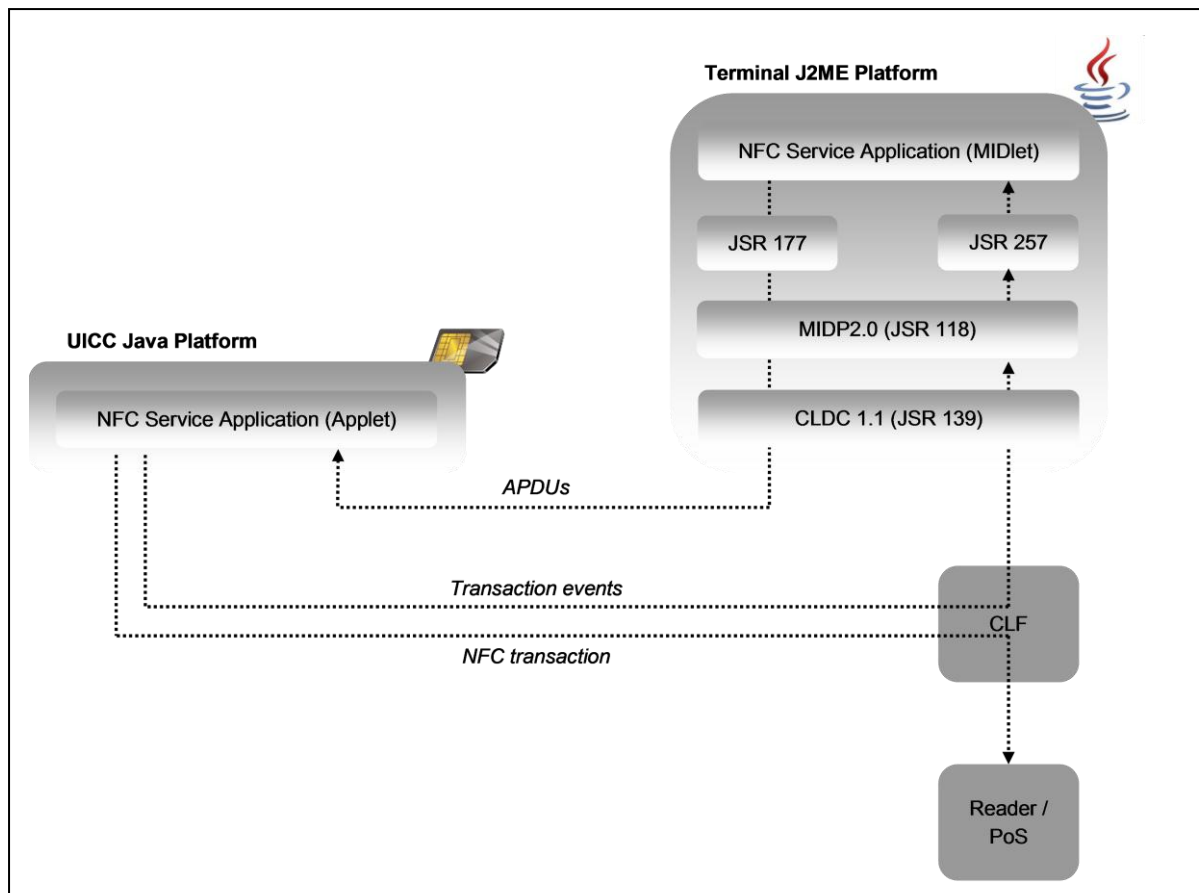


Figure 6: JavaME NFC software stack

5.2 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.

5.3 Security

Presently, the mobile device 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.

5.4 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.

5.5 Delta to Generic Architecture

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

6 Key NFC requirements

6.1 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)
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_13	The mobile device SHALL support NFC Forum Tag Type 3
Note: Annex D provides a MNO elective priority for this requirement	
NFC_REQ_14	The mobile device SHALL support NFC Forum Tag Type 4
NFC_REQ_15	The reader mode events SHALL be routed exclusively to the UICC or the

	Application processor at any one time.
NFC_REQ_16	Where the default routing for the reader mode events SHALL be via the Application processor.
NFC_REQ_17	The CLF SHALL route the reader mode events to the UICC when the UICC registers itself to receive the reader mode events.
NFC_REQ_18	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_19	Contactless tunnelling (CLT) mode SHALL be supported for SWP (per ETSI TS 102.613).

6.1.1 NFC controller support for ETSI interfaces with the UICC

NFC_REQ_20	The NFC controller SHALL support SWP interface with the UICC as per ETSI TS 102.613.
NFC_REQ_21	The NFC controller SHALL support HCI interface with the UICC as per ETSI TS 102.622.
NFC_REQ_22	Card Emulation mode SHALL be enabled as soon as the NFC hardware is turned on.
NFC_REQ_23	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.
NFC_REQ_24	The CLF configuration within the device SHALL enable card emulation mode with the UICC SE by default.

6.2 Mobile Device Modem Support

NFC_MOD_25	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.	
NFC_MOD_26	Access to the UICC (logical channel) SHALL be allowed even when the mobile device is in a Radio OFF state, i.e. flight mode, airplane mode etc

6.3 Mobile Device APIs

API_REQ_27	APDU APIs SHALL prevent access to basic channel (channel 0).
Note: For implementation purposes this can be achieved by raising an exception.	
API_REQ_28	APDU APIs SHALL prevent access to SELECT CHANNEL command.
API_REQ_29	APDU APIs SHALL prevent access to MANAGE CHANNEL command.
API_REQ_30	Devices SHALL provide Open Mobile SIM APIs (per SIM Alliance spec) for developers to use, specifically only the <i>transport layer</i> SHALL be

	implemented.
Note: Reference implementations for Open Mobile APIs and Access Control exist for Android.	
API_REQ_31	Devices SHALL provide an API which allows the handling of the NFC controller state.
API_REQ_32	Devices SHALL provide an API which allows for handling of the Card Emulation mode state. Mobile Device UI requirements
UI_REQ_33	The API SHALL be made available which allows to enable or disable the NFC hardware. (Note: as with Wi-Fi, GPS etc.).

6.4 Mobile Device APN management

APN_REQ_34	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.
APN_REQ_35	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 and the device SHALL use the handset default APN.
APN_REQ_36	If the APN information provided by the network in the OPEN CHANNEL command is empty the device SHALL always use the handset default APN

6.5 UI application triggering

The NFC controller must be able to trigger the appropriate UI application as per section 4.5.

UIApp_REQ_37	The mobile NFC Handset SHALL support HCI event EVT_TRANSACTION as per ETSI TS 102.622
UIApp_REQ_38	The AID parameter SHALL be used during the process of triggering the UI application.
API_REQ_39	Devices SHALL implement the card transaction event, intent format, as detailed in the document, Section 5.5.

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

RemMan_REQ_40	The mobile device SHALL support BIP in UICC client mode for UDP.
RemMan_REQ_41	The mobile device SHALL support BIP in UICC client mode for TCP.
RemMan_REQ_42	The mobile device SHALL support two concurrent channels, BIP in UICC client mode.
RemMan_REQ_43	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

6.7 NFC Controller TAG Support

TAG_REQ_44	The mobile device SHALL support Writer Mode.
TAG_REQ_45	The transaction SHALL take 500ms or less. (e.g. reading TAG to display, paying for transportation, etc.)
TAG_REQ_46	The mobile device SHALL be able to read/write the NFC Forum Smart Poster

	RTD.
TAG_REQ_47	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_48	The TAG SHALL be read from a distance of 0cm – 2 cms and SHOULD be read within the 2cms – 4cms range.
TAG_REQ_49	Following a TAG read, the user SHALL be prompted before any action is performed, and SHALL be able to dismiss this prompt.
TAG_REQ_50	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.

6.8 Multiple SE support (UICC SE and others)

MultiSE_REQ_51	Only one SE SHALL be active at any one time.
MultiSE_REQ_52	The mobile device software platform SHALL expose an API to select the active SE. And a change in the default SE selection SHALL not imply a power cycle of the handset, modem or the UICC.
MultiSE_REQ_53	If there is more than one SE, the default active SE SHALL be the UICC SE.
MultiSE_REQ_54	An API SHALL exist which allows for toggle switching between different SEs.
Note: Section 5.4 provides the guidelines for this implementation.	

6.9 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_55	The mobile device SHALL support, BIP in UICC server mode as per ETSI TS 102 223 R7 letter class “e”.
-------------	--

6.10 Access Control

The main objective of the Access Control mechanism is to protect access to the two main functions defined by the SIM Alliance Open Mobile API: “*Open Basic Channel*” and “*Open Logical Channel*”. Further, 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 UI application being alerted.

Similar to JSR 177, the rules used to grant or refuse 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, where 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 UI application and the SE applet (application signature/AID) is valid, the progress to authorise a communication or send an exception.

API_REQ_56	Open OS devices SHALL provide SIM API access control as per Global Platform, Secure Element Access Control specification.
API_REQ_57	When no access control data (files or applets) is found on the UICC the APDU API SHALL deny access to the UICC.
UIApp_REQ_58	The handset SHALL 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.
Note: Access Control reference code for device manufacturers will continue to be evolved to reflect the current Global Platform Access Control specification. System integrators should get in touch with the GSMA to obtain the source code	

Annex A 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 (Requirement is Market dependent)
	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	

Annex B 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 C Other Elective Requirements

C.1 Requirments

C.1.1 Battery Modes

NFC_REQ_C.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_C.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.	

C.1.2 Multiple Secure Elements (UICC SE and others)

MultiSE_REQ_C.3	The mobile device SHOULD support UICC SE only.
-----------------	--

Notes

Felica support

TAG Type 3 pertains to Felica which is not required.

Document Management

Document History

Version	Date	Brief Description of Change	Approval Authority	Editor / Company
1.0	July 2011	First draft submitted to DAG and PSMC for approval	PSMC and NFC (Rejected at PSMC level)	Sameer Tiku, Vodafone
2.0	November 2011	Second draft incorporating PSMC feedback submitted to DAG and PSMC for approval	PSMC and NFC (V2.0 Approved and published)	Sameer Tiku, Vodafone
3.0	03 October 2012	Submitted to DAG and PSMC for 7 day Committee Email approval	PSMC and NFC	Sameer Tiku, Vodafone
4.0	17 September 2013	Re-Submitted to DAG and PSMC for approval following comments received from PSMC	PSMC and NFC	Sameer Tiku, Vodafone
4.1	6 November 2013	Corrected and updated namespace to <com.gsma.services.nfc> . Impacted sections: 4.9, 4.10	Terminal Steering Group	Katrin Jordan, DT / TSG

Other Information

Type	Description
Document Owner	TSG
Editor / Company	Katrin Jordan

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@gsma.com

Your comments or suggestions & questions are always welcome.