

NFC Handset Requirements

Version 13.0

04 June 2018

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 © 2018 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 4
1.1 Purpose 4
1.2 Scope and Objective 4
1.3 Use Cases/Services 5
1.4 Abbreviations 5
1.5 Definition of Terms 6
2 References 9
3 Terminology 10
4 Guidance for GSMA API Implementation 10
5 Generic Device Architecture 12
5.1 Dual Application architecture 12
5.2 Security 14
5.3 Mobile Wallet 14
6 Generic Device Requirements 15
6.1 NFC Device Architecture 15
6.2 Core Required NFC Features 16
6.2.1 NFC Controller Management 18
6.2.2 Card Emulation Mode Requirements 18
6.2.3 Reader/writer mode & TAG management requirements 19
6.3 Secure Element Access & Multiple Secure Elements Management 20
6.3.1 Mobile Device Modem Requirements 20
6.3.2 Secure Element Access API requirements 21
6.3.3 Multiple CEE support 21
6.4 UI Application triggering requirements 25
6.5 Remote Management of NFC services 25
6.5.1 Mobile Device APN Management Requirements 25
6.5.2 UICC Remote Management (Access to UICC in connected mode) requirements 26
6.6 Security 26
6.6.1 NFC Event & Access Control requirements 27
6.6.2 Protecting APIs with Operator Certificate 27
6.7 SCWS support 28
6.8 Card Application Toolkit Support 28
6.9 Platform Dependent Properties 29
6.10 Personalization of the eSE 30
7 Android Operating System 31
7.1 NFC Device Architecture 31
7.2 Core Required NFC Features 31
7.2.1 NFC Controller management API 32
7.2.2 Card Emulation mode requirements 32
7.2.3 Reader/writer & TAG management requirements 32
7.3 Secure Element Access & Multiple Secure Elements Management 32

7.3.1 Mobile Device Modem requirements 32
7.3.2 Secure Element Access API requirements 32
7.3.3 Multiple CEE support 33
7.4 UI Application triggering requirements 38
7.4.1 Unicast – Reception by only one activity 39
7.4.2 Multicast – Reception by several application components (activity, services…) 40
7.5 Remote Management of NFC Services 40
7.6 Security 40
7.6.1 Access API & Secure Element Access Control requirements 41
7.6.2 NFC Event & Access Control requirements 41
7.6.3 Protecting sensitive APIs 42
7.7 SCWS support 42
7.8 Card Application Toolkit Support 42
7.9 Platform Dependent Properties 42
8 VOID 42
9 VOID 43
10 VOID 43
11 Android Wear Operating System 43
Annex A Implementation help of REQ 94.1 for multi eSE on Android 44
Annex B Document Management 45
B.1 Document History 45
Other Information 47

1 Introduction

1.1 Purpose

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

This document lists requirements for devices to support NFC services primarily focused on NFC services based on the UICC and eSE. It sets out a common framework of requirements, identifying and referencing relevant standards (or elements thereof), selecting options from among those allowed by existing standards to ensure interoperability. A list of relevant standards is captured in section 2 and further detailed by explicit requirements.

This document is delivered by the GSMA Terminal Steering Group (TSG), taking forward work driven by the GSMA TSG NFC Handset Requirements group. It is an update to and replaces all previous versions of TS.26, “NFC Handset Requirements” Specification.

Given the complexity of some of the underlying technology components and the variances across OS implementations, not all requirements could be finalised at this time. Where requirements are still work in progress, these are marked *yellow. Work is ongoing to finalise these as soon as possible as well as to further enhance requirements and details/applicability for the various OS and to publish updates with the next document versions.

This document applies to devices supporting a UICC, an eUICC and an eSE. As indicated in the definition of eUICC in section 1.5, an eUICC is a particular type of UICC. Therefore, when this document uses the term “UICC”, this incorporates both the standard UICC and the eUICC.

Basic Devices are also in the scope of this document and some requirements are specific to this category of devices.

The eUICC related specifications are being developed by the GSMA and ETSI. The latest version of the GSMA RSP specifications includes NFC support.

In case of any feedback or questions, you may notify us at prd@gsma.com.

1.2 Scope and Objective

The body of this document sets out requirements to be supported by mobile devices needed to support NFC services that are agreed globally, according to the GSMA’ s processes for consulting its members.

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

- Embracing new standards as and when they are published by the relevant industry organisations;
- Adding further requirements or further evolving current requirements as needed

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

- Align members' terminal requirements for SE based NFC services
- 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 device requirements.

This document defines at a high level the application architecture and lower layer enablers, required to fulfil NFC use cases. It further expands upon this, by detailing the particular mobile device Application Programming Interfaces (APIs) per OS (as applicable/ available) to enable a secured service use case and the requirements necessary to fulfil the NFC enabler software architecture.

Other specific OS requirements will be considered when contributions are received.

Note: this Permanent Reference Document (PRD) does not exclude the possibility for support of additional NFC capabilities not mentioned in this document.

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 SE based NFC *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

It is required that the device and the SE provide a *secured* environment, i.e. an environment which satisfies the security needs of Service Providers' (Mobile Network Operators' (MNOs)) and consumers.

1.4 Abbreviations

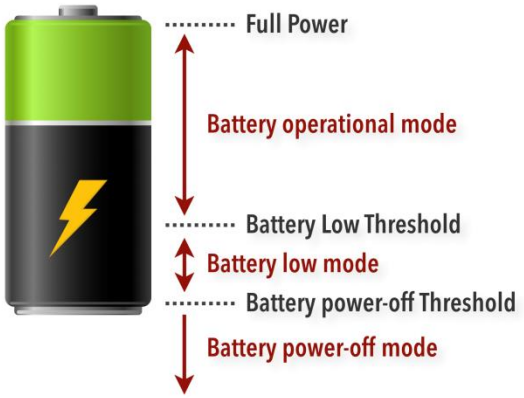
Term	Description
AC	Access Control
AID	Application ID
API	Application Programming Interface
APDU	Application Protocol Data Unit
APN	Access Point Name

Term	Description
BIP	Bearer Independent Protocol
CE	Card Emulation
CEE	Card Emulation Environment
CLF	Contactless Frontend
DUT	Device Under Test
eSE	Embedded SE
eUICC	A removable or non-removable UICC which enables the remote and/or local management of Profiles in a secure way. NOTE: The term originates from "embedded UICC".
JVM	Java Virtual Machine
HCE	Host Card Emulation
HCI	Host Controller Interface
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
PRD	Permanent Reference Document
RIL	Radio Interface Layer
SCWS	Smart Card Web Server
SDO	Standards Organisations
SE	Secure Element
SIM	Subscriber Identity Module
SP	Service Provider
SWP	Single Wire Protocol
TSG	Terminal Steering Group
UI	User Interface
UICC	Universal Integrated Circuit Card
UID	User Identification

1.5 Definition of Terms

Term	Description
Active CEE	An Active CEE is a CE environment that can receive data based on routing mechanisms (e.g. AID, Protocol and Technology).
Active UICC Profile	When the physical UICC is a standard UICC: the UICC itself. When the physical UICC is an eUICC: the combination of the Enabled Profile and the eUICC onto which the Profile has been provisioned.
AID Conflict	When two or more applications register with the same Application Identifier

Term	Description
AID Conflict Detection	Conflict Detection is the procedure to check for AID Conflict. Conflict Detection can be done 1. During registration of Application Identifiers 2. When an Application selection is being received from a contactless reader.
APDU	An Application Protocol Data Unit (APDU) is the communication unit between a smart card reader and a smart card.
Application Identifier	AID, as defined in ISO/IEC 7816-4, used to address a card emulation service or application
Basic Device	A Device with no screen or with a small basic display (i.e. not able to display menus)
Battery Operational Mode	The battery of the DUT has sufficient power to support all functions in the mobile devices.
Battery Low Mode	The battery of DUT has reached “Battery Low Threshold” at which the display and most functionalities of the DUT are automatically switched off, except the clock and a few remaining functions. The battery of the DUT only has sufficient power to support NFC controller to function in card emulation mode.
Battery Power-off Mode	The battery of the DUT has reached “Battery Power-off threshold” at which there is no residual power to support NFC controller to function. No functions are available in the DUT. The NFC controller can function if power is provided via the contactless interface (i.e. power by the field).



Battery power levels within the NFC mobile devices

Card Emulation Environment	A Card Emulation Environment is an execution environment used together with a NFC controller to manage a Card Emulation transaction. It can be a Secure Element (e.g. UICC, embedded Secure Element or micro-SD) or an application running in a device host.
Default AID Route	The Default AID route is the route used by the NFC Controller when a NFC reader explicitly selects a NFC Service by its AID but the AID is not defined in the NFC Controller’s routing table. Note: this definition is only relevant for devices which support the Multiple Active CEEs model.

Device	In the context of this specification, the term Devices is used to represent any electronic equipment supporting NFC functionality into which a NFC Secure Element can be inserted, and that provides a capability for a server to reach the UICC through an Over The Air (OTA) channel. E.g. smartphones, wearables.
Embedded UICC	A removable or non-removable UICC which enables the remote and/or local management of Profiles in a secure way. NOTE: The term originates from "embedded UICC".
Embedded SE	Secure Element which is a separated chipset and integrated into the devices, owned by the device manufacturer and cannot be removed.
Multiple Active CEEs model	A model where the device can activate several CEE at the same time. RF traffic can be provided to a CEE based on routing mechanisms. Note: an implementation may support Multiple Active CEEs model in Battery Operational Mode and Single Active CEE model in Battery Low or Power-Off Mode.
Operator	Refers to a Mobile Network Operator who provides the technical capability to access the mobile environment using an Over The Air (OTA) communication channel. The OPERATOR is also the UICC Issuer. An OPERATOR provides a UICC OTA Management System, which is also called the OTA Platform.
Prefix of AIDs	An AID prefix will match all AIDs that are starting with the same N bytes of the mentioned AID prefix.
Screen Lock	The device functionality can only be accessed via a user intervention.
Screen ON	The battery of the device is in Battery Operational Mode and the screen of the device was turned on by the end-user (i.e. the screen is active).
Screen OFF	The battery of the device is in Battery Operational Mode and the screen of the device was turned off either by the end-user or automatically by the device after a timeout.
Switched OFF	The device was turned OFF by the end-user or the device is in battery low mode or the device is in battery power-off mode.
Secure Element	A SE is a tamper-resistant hardware component which is used to provide security, confidentiality, and multiple application environments required to support various business models. In TS.26, the term SE includes UICC, eUICC and eSE.
Sensitive API	An API which shall be protected from malicious use.
Single Active CEE model	A model where the device only activates one CEE at a time. Other CEEs, if available, are not active.

2 References

Note: Testing shall be based on the exact versions as indicated below. However if the manufacturers use a later release and/or version this should be indicated. TSG will take efforts to continually align with other SDOs for timely information about release plans.

3GPP Specifications	3GPP TS 31.111 (Release V15.0.0 or later) Universal Subscriber Identity Module (USIM) Application Toolkit (USAT) 3GPP TS 31.116 (Release V14.0.0 or later) Remote APDU (Application Protocol Data Unit) Structure for (Universal) Subscriber Identity Module (U)SIM Toolkit applications Later releases of 3GPP specifications shall be backward compatible The manufacturer can use Release 9 or a later release of the specifications
EMVCo Specifications	EMV Contactless Communication Protocol Specification, Book D, Version 2.6 (or later)
ETSI SCP Specifications	ETSI TS 102 221 (Release V13.1.0 or later): Smart Cards; UICC- Terminal interface; Physical and logical characteristics ETSI TS 102 613 (Release V11.0.0 or later): UICC – Contactless Front End (Physical and data link layer characteristic) ETSI TS 102 622 (Release V13.0.0 or later): UICC – Contactless Front End, HCI (Host Controller Interface) ETSI TS 102 223 (Release V12.0.0 or later): Card Application Toolkit ETSI TS 102 226 (Release V13.0.0 or later): Remote APDU structure for UICC based applications Later releases of ETSI-SCP specifications shall be backward compatible The manufacturer can use Release 9 or a later release of the specifications
Javadoc	Javadoc containing API's requested within this PRD: http://gsmaterminals.github.io/NFC-Handset-Requirements/Android/doc/index.html Note: From version 11 of TS.26 GSMA APIs are deprecated.
NFC Forum Specifications	NFCForum-TS-T1T-1.0 (or later) NFCForum-TS-T2T-1.0 (or later) NFCForum-TS-T3T-1.0 (or later) NFCForum-TS-T4T-1.0 (or later) NFCForum-TS-RTD_SP-1.0 (or later) NFCForum-TS-RTD-1.0 (or later) NFCForum-TS-RTD_TEXT-1.0 (or later) NFCForum DeviceRequirements_2.0 (or later) NFCForum-TS-Analog-2.0 (or later) NFCForum-TS-Digital-1.1 (or later) NFCForum-TS-Activity-1.1 (or later) NFCForum-TS-NDEF-1.0 (or later) NFCForum-TS-NCI 1.1 (or later)
SIMalliance	SIMalliance Open Mobile API Specification v3.2 and Second Errata
GlobalPlatform	Open Mobile API Specification v3.3 or later

GSMA Requirements	SGP.03 NFC UICC Requirement Specification, Version 7.0 SGP.12 NFC Multi Protocol for Interoperability, Version 2.0
GlobalPlatform	Secure Element Access Control Version 1.0
GlobalPlatform	GlobalPlatform Secure Element Configuration Version 1.0
GlobalPlatform	GlobalPlatform Card Specification Version 2.3
RSA Laboratory	PKCS #15 v 1.1: Cryptographic Token Information Syntax

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
MAY	Denotes Optional

4 Guidance for GSMA API Implementation[1]

TS26_NFC_REQ_001	Where an OS API call does not return immediately, the OS implementation SHOULD be non-blocking.
TS26_NFC_REQ_002	Where an OS platform supports an exception mechanism, API design SHOULD use exceptions to indicate error conditions.
TS26_NFC_REQ_003	There MAY be a single canonical API supporting the GSMA NFC Handset requirements for each OS.
TS26_NFC_REQ_004	Where an API is marked Deprecated, it SHALL be implemented by OS device platforms conforming to the API revision in which it was deprecated and SHOULD be implemented by device platforms conforming to the next major revision. It MAY be implemented by device platforms conforming to later revisions. Note: As an example, if an API is deprecated from v11, it shall be implemented after the v11 publication. After the release of v12, it should be implemented. After the release of v13, it may be implemented.
TS26_NFC_REQ_005	Where support of the requirements in this document requires additions to the APIs provided by the vendor of an OS implementation, these additions SHALL be defined using a naming convention which reflects GSMA authorship of the API.

[1] Note: This section is considered relevant for any OS. It may however not equally and fully apply for every OS. Work will continue to make this section even more OS-agnostic, complemented by OS-specific guidance in the later chapters.

Note: From version 11 of TS.26 GSMA APIs are deprecated. EVT_TRANSACTION (TS26_NFC_REQ_071) is not affected by the deprecation.

TS26_NFC_REQ_003 requires that there shall be a single canonical API supporting NFC Handset Requirements on each OS platform. This is intended to ensure that application developers for a given OS platform have a single, stable API to which they can develop on a given OS.

It is preferred that OS vendors provide support for all of the GSMA requirements in their NFC and Secure Element API definitions, since this provides the best assurance of a stable and well maintained developer API.

The GSMA may choose to define API extensions which provide the necessary features (for example, with the Android OS). Implementers of such extensions should take care to re-use existing OS mechanisms and practices where possible, to ensure the most consistent behaviour between different implementations of GSMA API extensions.

Where OS extensions are defined by the GSMA, the API for these extensions should be written clearly to identify GSMA authorship. The recommended way to implement this, for OS platforms supporting namespaces, is to use the namespace ‘com.gsma.services.nfc’ throughout the API definitions. For platforms not supporting namespaces – an example would be a C language API – a similar convention should be adopted, consistent with language limitations (e.g. com_gsma_services_nfc_ApiFunction() may be appropriate for a C language API). This is codified in TS26_NFC_REQ_005.

To assist platform vendors in implementing any GSMA platform specific requirements/APIs, the API definitions should include the following information:

Pre-conditions: These are assertions about platform state when an API is called. For example:

- NFC is enabled;
- The supplied instance of NfcAdapter is not null;

Processing: This describes the function performed by the API. It should include details of behaviour such as: blocking/non-blocking behaviour; threading behaviour (e.g. ‘should only be called from the main UI thread’); timing constraints for operation, as appropriate.

Post-conditions: These are assertions about the platform state when the API returns. This section will typically indicate how the systems state should change after execution of the API.

Exception Handling: API definitions should include a clear description of expected API behaviour when an error occurs in processing, and/or when a precondition is not fulfilled. Experience shows that inadequately specified exception handling is among the most common causes of differing behaviour between implementations.

5 Generic Device Architecture

5.1 Dual Application architecture

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

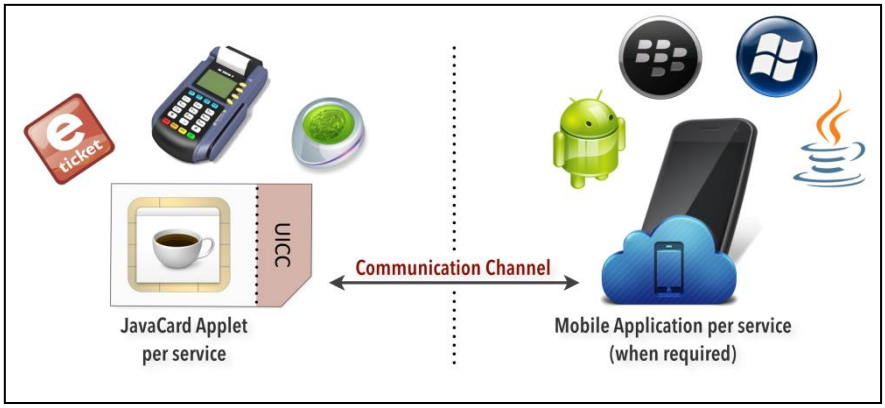


Figure 1**: Dual application architecture**

The mobile device User Interface (UI) application executing on the device OS is the consumer facing component. In this example, the UI application interacting with the application on an SE, communicating with the NFC reader, allows the customer to interact with the service functionalities, e.g. with a PoS (point of sale) for a financial service use case or a physical ticketing barrier in the case of an e-ticketing application. However the UI Application component is not seen as mandatory for all use cases, where the Service Provider (SP) could decide to have a UI-less service, including when the service is intended to be deployed on Basic Devices. It could be also the case that device applications without UI are deployed and finally a User Interface does not necessarily require the presence of a display, but it could be achieved by sounds, LEDs or vibrates. In the rest of the document the term “UI” designates all kind of interfaces allowing an interaction with or a simple notification to the user.

The *applet* component resides within the SE, and works in tandem with the UI application when applicable. It holds the logic of the application and performs actions such as holding *secure authentication keys* or *time-stamped transaction data* 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. This communication channel could be used to transmit status information passed from the application in the SE to the UI for notifying the user on NFC events. It could also be used for more information exchanges between the SE and the device UI like user authentication toward a SE applet (e.g. PIN code verification).

As the communication channel accesses a *secured* storage space on the SE, the communication channel itself must have attributes which allow it to be accessed only by authorised UI applications.

The following illustration gives an overview of the device software components required to satisfy the dual application architecture, which delivers key use cases for NFC, in case of a NFC handset with a UICC.

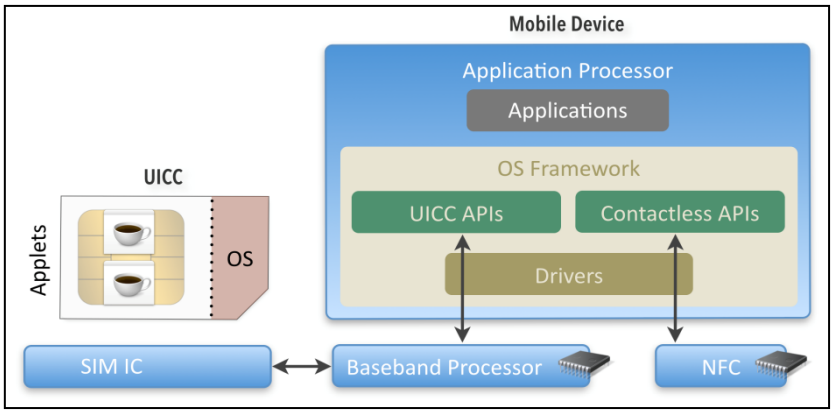


Figure 2**: Mobile Device API generic software stack**

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

The following figure depicts the typical data flow for a NFC transaction, between a PoS and a UICC, including 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 activity the nature of the *event* between the various components can be determined, for example where the *event* needs to be protected and has attributes which will allow for, or not allow for, any modification. The same flow will take place between a PoS and an eSE

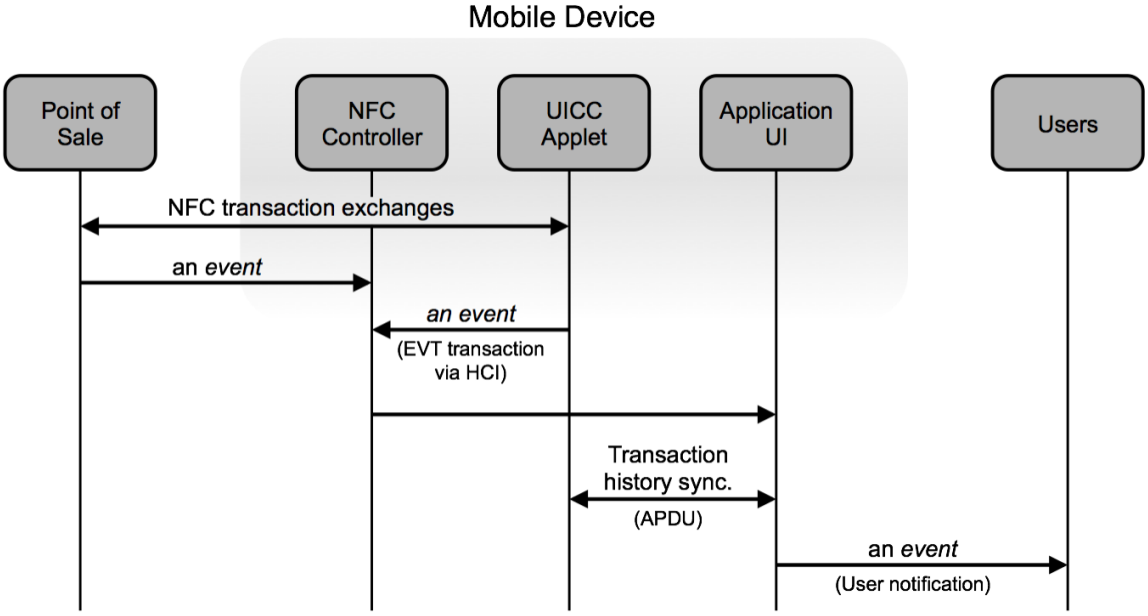


Figure 3**: Typical data flow for card-emulation mode**

5.2 Security

For the *secured* services use case it is imperative for MNOs and SPs to continuously strive to provide best possible secured and trusted communication along the end-to-end chain of the various components necessary.

- Two key areas where security is important are the Secure Element and the privileges available to communicate with the NFC service applet in the SE. The SE will securely hold protected information, and provide a controlled access path to relevant parts of its internal memory.
- Access to services inside a SE is requesting a specific care as a high level of security is required by some Service Providers. It is necessary to manage which device applications communicate with applets in the SE. In addition to existing protection mechanisms provided by the mobile OS, a dedicated Access Control mechanism based on rules/rights provided by the SE is needed. The main purpose of this *Access Control* is typically to prevent service attacks from malware applications.

5.3 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 SEs. The wallet application, for example, can typically list all SP services loaded into the mobile device or SE and display their current status. Additionally, this application may also allow the users to manage the NFC settings of their mobile device. This type of application is not relevant for Basic Devices. Users could manage their NFC services installed on such Basic Devices from a *Mobile Wallet* installed on a paired smartphone. The way this is achieved is out of scope of this document.

6 Generic Device Requirements

6.1 NFC Device Architecture

The following figure provides an overview of a typical Mobile NFC architecture:

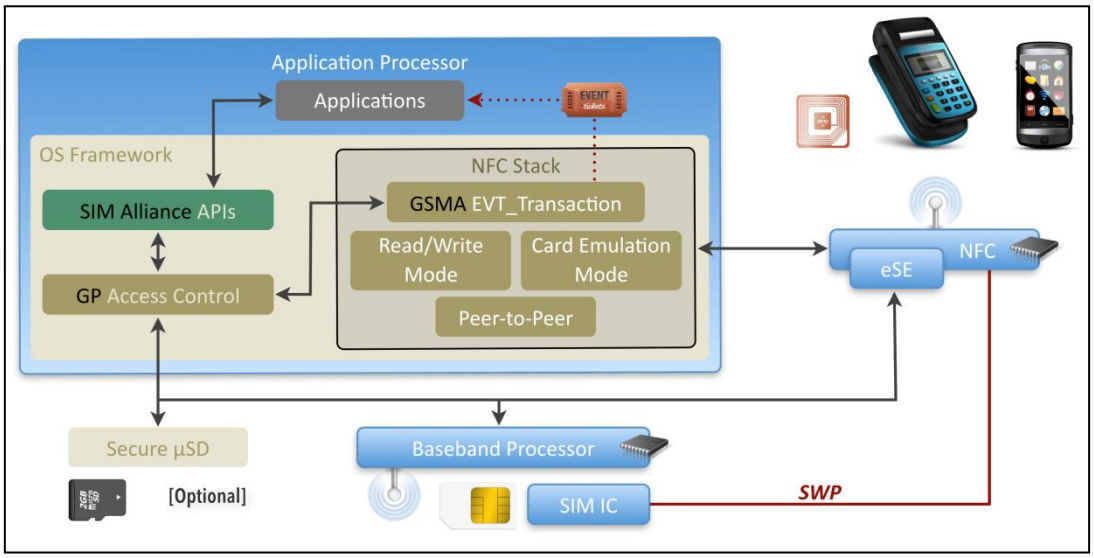


Figure 4**: Mobile NFC Architecture**

The device provides, as standard component, a NFC controller and one or more SEs.

The NFC Stack is driving the NFC Controller and is typically providing software APIs enabling:

- Management of Multiple Secure Element (activation, deactivation, routing, etc.)
- Management of the NFC events
- An external API available for 3rd party applications to manage reader/writer mode, Peer to Peer mode and Card Emulation mode from Device
- An internal API to provide a communication channel with an embedded Secure Element for APDU exchanges

The Secure Element Access API provides a communication channel (using APDU commands) allowing 3rd party applications running on the Mobile OS to exchange data with Secure Element Applets. This API provides an abstraction level common for all Secure Elements and could rely on different low level APIs for the physical access:

- RIL extension for accessing the UICC
- Specific libraries for communicating with other embedded secure elements

In order to implement security mechanisms (e.g. Secure Element Access Control), the Secure Element Access API shall use Mobile OS mechanisms such as UIDs or application certificates to identify the calling application.

6.2 Core Required NFC Features

TS26_NFC_REQ_006	The NFC controller SHALL support SWP (Single Wire Protocol) interface with the UICC as per ETSI TS 102 613.
TS26_NFC_REQ_173	A device MAY be shipped with an eSE for NFC services.
TS26_NFC_REQ_173.1	The NFC controller SHALL support an interface with the eSE. Note: The interface can be SWP or any other interface.
TS26_NFC_REQ_007	The NFC controller SHALL support HCI with the UICC as per ETSI TS 102 622.
TS26_NFC_REQ_164	For devices with more than one UICC slot the NFC controller SHALL support SWP and HCI interface with at least one UICC slot of the device as per ETSI TS 102 613 and ETSI TS 102 622.
TS26_NFC_REQ_164.1	For devices with more than one NFC capable UICC slot, there SHALL only be one NFC capable UICC slot active at any one time.
TS26_NFC_REQ_164.2	Each NFC capable UICC slot SHALL be indicated to the user.
TS26_NFC_REQ_164.3	For devices with more than one NFC capable UICC slot, a menu SHALL give the option to the user to select on which NFC capable UICC slot, NFC is active.
*TS26_NFC_REQ_165	If the device has both an eUICC and a UICC slot the device SHALL implement the NFC card emulation on both.
TS26_NFC_REQ_166	The device SHALL send the Terminal Capability command to the UICC indicating that the UICC-CLF interface (SWP) is supported as per ETSI TS 102 221.
TS26_NFC_REQ_008	Contactless tunnelling (CLT=A) mode SHALL be supported for SWP (per ETSI TS 102 613).
TS26_NFC_REQ_009	VOID
TS26_NFC_REQ_009.1	Contactless tunnelling (CLT=F) mode SHALL be supported for SWP (per ETSI TS 102 613).
TS26_NFC_REQ_010	The device interface with UICC SHOULD support Class B.
TS26_NFC_REQ_011	The device interface with UICC SHALL support Class C.
TS26_NFC_REQ_012	VOID
TS26_NFC_REQ_137	The NFC controller interface with UICC SHALL use ETSI TS 102 613 full power mode when the device is in battery operational mode.
TS26_NFC_REQ_013	VOID
TS26_NFC_REQ_154	The NFC controller interface with UICC MAY use ETSI TS 102 613 full power mode when the device is in battery low mode.
TS26_NFC_REQ_138	The NFC controller interface with UICC MAY use ETSI TS 102 613 full power mode in battery power-off mode.
TS26_NFC_REQ_139	If the NFC controller interface with UICC is not supporting ETSI TS 102 613 full power mode in battery low mode, then the NFC Controller SHALL support ETSI TS 102 613 low power mode when the device is in battery power-low mode.

TS26_NFC_REQ_140	If the NFC controller interface with UICC is not supporting ETSI TS 102 613 full power mode in battery power-off mode, then the NFC Controller SHALL support ETSI TS 102 613 low power mode when the device is in battery power-off mode.
TS26_NFC_REQ_107	The device manufacturer SHALL provide information to the user about the position of the NFC antenna reference point. Examples: a marker on the device, a removable sticker, a device user manual or a tutorial that is started when activating NFC for the first time.
TS26_NFC_REQ_014	The device interface with UICC SHALL support DEACTIVATED followed by subsequent SWP interface activation in full power mode.
TS26_NFC_REQ_015	The NFC controller SHOULD support both windows size set to 3 and set to 4.
TS26_NFC_REQ_016	VOID
TS26_NFC_REQ_017	The NFC controller SHALL ensure that the UICC SWP/HCI initialization is finished before deactivating the SWP without full power down UICC. Note: The SWP/HCI specification is not integrating a recovery mechanism so in case of SWP line deactivation in the middle of the activation, it may lead to blocking situation, with SWP-UICC interface not usable until the next device boot.
TS26_NFC_REQ_018	VOID
TS26_NFC_REQ_019	The NFC Controller SHALL support configuration of the listen mode routing for Card Emulation, by the device manufacturer or operator, for at least ISO DEP, NFCA, NFCB & NFCF.
TS26_NFC_REQ_020	If NFC was enabled, when the mobile device is automatically switched off, and enters battery low mode, the mobile device SHALL be able to perform 15 transactions in card emulation within the following 24 hours.
TS26_NFC_REQ_174	If NFC was enabled when the device is switched off by the user, the device SHALL be able to perform card emulation transactions.
TS26_NFC_REQ_021	If NFC is enabled, NFC transactions SHALL be possible either in battery power off or battery low mode. Note: This is important for public transport services.
TS26_NFC_REQ_167	The NFC controller SHALL support at least 16 AIDs of 16 bytes in the routing table. Note: 16 AIDs is the minimum and in the future the requirement might be updated.
TS26_NFC_REQ_175	In case the NFC Controller receives a RF parameters configuration request from a CEE enabling a Mifare Classic service with either UID 4 or 7 bytes, the corresponding RF parameters profile “Profile 2 ” as defined in chapter 3.1.2.2 of GSMA SGP.12 SHALL apply
TS26_NFC_REQ_176	In case the NFC Controller receives a RF parameters configuration request from a CEE enabling a Mifare DESFire service, the RF parameters profile “Profile 3” as defined in chapter 3.1.2.2 of GSMA SGP.12 SHALL apply
TS26_NFC_REQ_177	In other cases, the RF parameters profile “Profile 1” as defined in chapter 3.1.2.2 of GSMA SGP.12 SHALL apply

6.2.1 NFC Controller Management

The following features are needed:
Management of the NFC Controller state

- Check if the NFC Controller is enabled
- Activate the NFC Controller
- Check if Card Emulation mode is activated

TS26_NFC_REQ_022	There SHOULD be an API to ask the system to enable the NFC functionality. User input SHALL be required to enable NFC. This UI dialogue SHALL be generated by the OS and not by the calling application.
TS26_NFC_REQ_023	Devices SHALL provide an API which allows the query of the NFC controller's state.
TS26_NFC_REQ_024	The OS implementation MAY provide an API that allows an application to determine the state of the Card Emulation.
TS26_NFC_REQ_108	VOID
TS26_NFC_REQ_109	In non Basic Devices the OS SHALL allow the user to turn NFC on and off when the device is in Radio Off Mode.
TS26_NFC_REQ_178	In Basic Devices the OS SHOULD allow the user to turn NFC on and off when the device is in Radio Off Mode. If not the NFC function SHALL always be enabled.

6.2.2 Card Emulation Mode Requirements

TS26_NFC_REQ_025	The mobile device SHALL support Card Emulation as per NFC Forum Technical Specifications: TS-Analog, TS-Digital and TS-Activity.
TS26_NFC_REQ_025.1	VOID
TS26_NFC_REQ_026	Card Emulation mode SHALL be enabled when the NFC is turned on.
TS26_NFC_REQ_027	For Card emulation mode the read distance SHALL be in the 0cm – 2cms range for battery operational mode, battery low or power off mode.
TS26_NFC_REQ_028	In Single Active CEE model the Active UICC Profile SHALL be the default CEE, that is the active CEE at first start up or after a factory reset.
TS26_NFC_REQ_029	Manufacturers SHALL provide to operators the capability to customise settings for defining if the UICC card emulation is enabled / disabled when the device is powered off, screen is off or locked. Note: this will not be via a UI.
TS26_NFC_REQ_030	In the case of a factory reset, the operator customised settings (as per TSG26_NFC_REQ_029) SHALL remain.
TS26_NFC_REQ_031	Operator settings as stated in TS26_NFC_REQ_029 above SHALL only be valid if NFC is enabled.
TS26_NFC_REQ_157	The device SHALL implement the requirements of the EMV Contactless Communication Protocol Specification, Book D.
TS26_NFC_REQ_158	Card emulation mode SHALL support APDU transmission case 1, 2, 3 & 4 as defined in ISO/IEC 7816-4 including Extended Length Field support. Command and response data field size minimum of 2048 bytes SHALL be supported. Note 1: Currently, the support for extended length APDU is not a common feature of NFC-UICC. At this point in time, NFC-UICC in the field typically don't support extended length APDU. Both handset architecture and NFC-

	UICC have to be compliant in order for the device to support the extended length APDU feature. Note 2: The implementation of the protocol and the mechanisms leading to the use of the extended length APDU option according to ISO/IEC 7816-4 have to be ensured by a negotiation between the contactless reader and the selected application in the NFC-UICC.

6.2.3 Reader/writer mode & TAG management requirements

All requirements in this chapter are optional for Basic Devices.

TS26_NFC_REQ_032	VOID
TS26_NFC_REQ_033	The mobile device SHALL support Reader/Writer Mode as per NFC Forum Technical Specifications: TS-Analog, TS-Digital and TS-Activity.
TS26_NFC_REQ_034	The mobile device SHALL support NFC Forum Type 1 Tag, as specified in NFC Forum Type 1 Tag Operation Specification. Requirement applies to both protocol and application level.
TS26_NFC_REQ_035	The mobile device SHALL support NFC Forum Type 2 Tag, as specified in NFC Forum Type 2 Tag Operation Specification. Requirement applies to both protocol and application level.
TS26_NFC_REQ_036	The mobile device SHALL support NFC Forum Type 3 Tag, as specified in NFC Forum Type 3 Tag Operation Specification. Requirement applies to both protocol and application level.
TS26_NFC_REQ_037	The mobile device SHALL support NFC Forum Type 4 Tag, as specified in NFC Forum Type 4 Tag Operation Specification. Requirement applies to both protocol and application level.
TS26_NFC_REQ_038	Reader mode events SHALL be routed exclusively to the UICC or the Application processor.
TS26_NFC_REQ_039	The default routing for the reader mode events SHALL be via the Application processor.
TS26_NFC_REQ_040	The NFC Controller SHOULD support Reader Mode as per ETSI TS 102 622.
TS26_NFC_REQ_041	The device SHALL support automatic and continuous switching between card emulation and reader mode.
TS26_NFC_REQ_159	VOID
TS26_NFC_REQ_160	The mobile device SHALL support APDU transmission case 1, 2, 3 & 4 including Extended Length Field support as defined in ISO/IEC 7816-4 with 32767 bytes command and response data field size for the Reader/Writer mode.

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.

TS26_NFC_REQ_042	A transaction time SHALL take 500ms or less for TAG message length not exceeding 100 bytes. The transaction time is defined from the start of the

	frame of the first RF command receiving an answer, to the end of the frame of the response to the last received RF command by a device, where the RF command is used to read the content in a tag.
TS26_NFC_REQ_043	The mobile device SHALL be able to read/write the NFC Forum Smart Poster RTD.
TS26_NFC_REQ_044	The TAG SHALL be read at a distance of 1 cm and at distances between 0 to 1 cm. Note: This requirement will be tested with a TAG Test Reference system agreed in the Test Book group.
TS26_NFC_REQ_110	The TAG SHOULD be read at a distance from 1 cm to 4 cm. Note: This requirement will be tested with a TAG Test Reference system agreed in the Test Book group.

6.3 Secure Element Access & Multiple Secure Elements Management

This section details functionality which the GSMA requires to be implemented within the NFC Framework, in order to support requirements in this document related to handling of the NFC Controller, Card Emulation mode and multiple Secure Elements.

6.3.1 Mobile Device Modem Requirements

TS26_NFC_REQ_045	For handling logical channel Baseband SHALL provide interfaces based on either the following AT commands or equivalent functionality: • AT+CCHO • AT+CCHC • AT+CGLA
TS26_NFC_REQ_045.1	Modem SHALL support all lengths of AID from 5 bytes to 16 bytes as defined in ISO/IEC 7816-4.
TS26_NFC_REQ_141	The modem SHALL provide a way for the application processor to retrieve the answer from the UICC after the selection of an AID.
TS26_NFC_REQ_111	Modem SHALL provide an interface based on AT+CRSM (Restricted SIM Access) or equivalent functionality for internal communication on basic channel.
TS26_NFC_REQ_112	The Modem SHALL prevent the usage of the AT+CSIM or equivalent functionality.
TS26_NFC_REQ_113	Modem SHALL support APDU transmission case 1, 2, 3 & 4 as defined in ISO/IEC 7816-4.
TS26_NFC_REQ_161	Modem SHALL support Extended Length APDU as defined in ISO/IEC 7816-4 with at least 2048 bytes command and response data field size. Note: This requirement will become effective from the 1st January 2018.
TS26_NFC_REQ_114	For all APDU exchanges originating from the Secure Element Access API the Modem driver SHALL forward warning status codes (SW=62XX or 63XX) directly to the application level without any change.
TS26_NFC_REQ_155	For all APDU exchanges originating from the Secure Element Access API the Modem driver SHALL allow the mobile application to perform a GET RESPONSE after any warning status code (SW=62XX or 63XX) is sent

	back by the UICC.
TS26_NFC_REQ_115	VOID
TS26_NFC_REQ_046	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.
TS26_NFC_REQ_142	The modem SHALL support 19 logical channels in addition to the basic channel.

6.3.2 Secure Element Access API requirements

The SIMalliance group has published the “Open Mobile API” specification until version 3.2. The specification has thereafter moved to GlobalPlatform Device committee. From this document, any mobile device manufacturer will be able to provide a standardised API for access to the different Secure Elements such as the UICC SE.

TS26_NFC_REQ_047	OS implementations SHALL provide an API for communicating with all NFC SE inside the device (UICC, eUICC, eSE).
TS26_NFC_REQ_047.1	Communication with SEs SHALL be done through the logical channels.
TS26_NFC_REQ_047.2	Communication with the Active UICC Profile SHALL prevent access to basic channel (channel 0).
TS26_NFC_REQ_047.3	The API SHALL implement the SIMalliance Open Mobile API transport layer or provide an equivalent set of features. SIMalliance OMAPI specification does not define values for EventTypes. This is a known issue and is expected to be addressed in a future version of the SIMalliance OMAPI specification. For implementations based on the current version of TS.26, the following values SHALL be used for EventTypes: Reader:IOErrorEventType = (int) 0x1001 Reader:SEInsertedEventType = (int) 0x2001 Reader:SERemovalEventType = (int) 0x2002 It is anticipated that a future version of the SIMalliance OMAPI specification will use the same values as specified above.
TS26_NFC_REQ_048	VOID
TS26_NFC_REQ_049	VOID
TS26_NFC_REQ_050	VOID
TS26_NFC_REQ_183	The API SHALL be able to send the Select by AID command with zero length AID (as defined in GlobalPlatform card specification) to the eSE. Note: In order to select the Issuer Security Domain without knowing the AID.

6.3.3 Multiple CEE support

6.3.3.1 VOID

6.3.3.1.1 VOID

TS26_NFC_REQ_051	VOID
TS26_NFC_REQ_051.1	VOID

TS26_NFC_REQ_051.2	VOID
TS26_NFC_REQ_052	VOID

6.3.3.1.2 VOID

TS26_NFC_REQ_053	VOID
TS26_NFC_REQ_054	VOID
TS26_NFC_REQ_116	VOID

6.3.3.2 Multiple Active CEE model

The following requirements only apply where a device supports the Multiple Active CEEs model.

TS26_NFC_REQ_117	In Multiple Active CEE model, the UICC SHALL be one of the active CEEs if the inserted UICC supports SWP.
TS26_NFC_REQ_179	In Multiple Active CEE model, the eSE MAY be one of the active CEEs if an eSE is part of the device.

6.3.3.2.1 NFC Controller Management API

TS26_NFC_REQ_055	The device SHALL support an API that allows applications to dynamically register and un-register NFC application by list of AIDs.
TS26_NFC_REQ_055.1	All the AIDs registered dynamically SHALL stay persistent (still available after a power off/on of the device).
TS26_NFC_REQ_056	VOID
TS26_NFC_REQ_057	The device SHOULD support an API that allows applications to dynamically register and un-register NFC application by pattern (e.g. DESFIRE).
TS26_NFC_REQ_058	An API SHALL be offered by the OS to provide a way for the applications to specify which card emulation environment their AIDs are to be routed to. Note: To allow applications to register AIDs belonging to UICC CEE and/or eSE CEE
TS26_NFC_REQ_059	API MAY provide a way for a mobile application to list all available secure elements in battery operational mode for NFC transactions.
TS26_NFC_REQ_060	API MAY provide a way for a mobile application to list all configured secure elements in NFC controller for battery low and battery power off mode.
TS26_NFC_REQ_061	The device MAY support an OS mechanism that allows applications to statically register NFC application by list of AIDs.
TS26_NFC_REQ_062	The device SHOULD support an OS mechanism that allows applications to statically register NFC application by pattern (e.g. DESFIRE).
TS26_NFC_REQ_168	The device SHALL support an OS mechanism for the registration of prefix AIDs and offer a way for applications to use it.
TS26_NFC_REQ_168.1	The device SHALL only allow prefixes longer than or equal to 5 bytes and less than 16 bytes. In the case of an AID registered matching a prefix also registered, the rule is to route according to the longest match.

6.3.3.2.2 Card Emulation Mode Requirements

TS26_NFC_REQ_095	The device SHALL support routing to active CEEs.
TS26_NFC_REQ_118.1	At first power up or factory reset the device SHALL set the route for NFCA and NFCB technologies (as defined in NFC Forum specification) to the UICC.
TS26_NFC_REQ_118.2	At first power up or factory reset the device SHALL set the route for ISO_DEP protocol (as defined in NFC Forum specification) to the UICC.
TS26_NFC_REQ_118.3	At first power up or factory reset the device SHOULD set the Default AID route to the UICC.
TS26_NFC_REQ_162	When a NFC Reader explicitly selects a NFC Service by its AID but the AID is not defined in the NFC Controller's routing table, the NFC Controller SHALL route the transaction to the Card Emulation Environment identified as Default AID route.
TS26_NFC_REQ_162.1	The default AID route SHALL be independent of any other routes configured in the NFC controller (such as those for RF Protocol (for example: ISO_DEP, T1t, T2t, T3t) or RF Technology (for example: NFC_A, NFC_B, NFC_F)).
TS26_NFC_REQ_119	VOID
TS26_NFC_REQ_063	When a NFC application is uninstalled, the device SHALL remove all information related to this application from the routing table.
TS26_NFC_REQ_063.1	When a NFC application is disabled, the device SHALL remove all information related to this application from the NFC routing table. Note: this also applies for preinstalled applications that cannot be uninstalled but that can only be disabled.
TS26_NFC_REQ_064	When a NFC application is updated or re-enabled, the device SHALL update the routing table according to the new registration information (removing/adding elements). Note: Static elements from the previous version will be removed and static elements from the new version will be added.
TS26_NFC_REQ_143	When the device needs to update the routing table because of new AID registration; **AND** • there is not enough space in the routing table for all required AIDs while maintaining the current default route; **AND** • there would be enough space in the routing table for all required AIDs if the default AID route was changed to one of the other card emulation environments, **THEN** the device SHALL change the default AID route automatically to one of those other card emulation environments and SHALL update the routing table accordingly. In such situation, there SHALL be no user interaction at all. Note: This mechanism needs to be totally transparent for end users as this is a solution resolving the chipset routing table size shortage.

<table>
<tr><td>TS26_NFC_REQ_065[2]</td><td>The device SHALL provide a routing mechanism using the following priority: AID, then default AID route, then RF protocol and then RF technology[3].</td></tr>
<tr><td>TS26_NFC_REQ_065.1</td><td>When the device is powered off and a NFC reader is trying to select by AID a NFC service relying on the HCE technology, the NFC Controller SHALL return an ISO error code (‘6A82’) indicating this service is not available.
<table>
<tr><th>Routing Table</th><th>NFC Controller action when device is powered off</th></tr>
<tr><td>• Contains AID based on HCE
• Default AID route set to “Off-Host”</td><td>• Return an ISO error code (‘6A82’) for AID stored in the routing table and related to HCE.
• Forward others request to the “Off-Host”</td></tr>
<tr><td>• Contains “Off-Host” AIDs
• Default AID route set to HCE</td><td>• Forward the request for all the AID stored in the routing table to OFF-Host
• - Return an ISO error code (‘6A82’) for all the other AID select requests</td></tr>
</table></td></tr>
<tr><td>TS26_NFC_REQ_066</td><td>VOID</td></tr>
<tr><td>TS26_NFC_REQ_067</td><td>When manual mechanism is used to register the new NFC service, the device SHOULD provide high level information to the user (i.e. NFC service name and not AID, etc.).</td></tr>
</table>

6.3.3.2.3 AID Conflict Resolution

TS26_NFC_REQ_068	The device SHALL provide a mechanism to handle AID Conflict.
TS26_NFC_REQ_068.01	AID Conflict resolution SHOULD follow the same mechanisms whether NFC services registration is dynamic or static.
TS26_NFC_REQ_068.02	When managing AID conflict resolution, the device SHALL follow the end-user preferences. Note: In case of a Basic Device, end-user preference may have been set via a paired device or from a connected PC. The way this is achieved is out of scope of this document.
TS26_NFC_REQ_068.03	For non Basic Devices, when AID conflict resolution implies user choice at transaction time the user SHOULD be able to make their decision persistent, until the terms of the AID conflict change (another application is installed or removed, which claims the same AID), or the user goes into the menus to revisit their decision. Note: For Basic Devices the end-user is not able to make such choice at transaction time. It is the responsibility of the device manufacturer to implement mechanisms that will detect such conflict at installation time and ask for end-user preferences.

[2] Note: This requirement will be updated to also reference pattern, as soon as relevant specifications from NFC Forum or GlobalPlatform are available.

[3] Refers to NCI specification [NFC Forum specifications] for details of routing based on AID, RF protocol and RF technology.

6.4 UI Application triggering requirements

When a transaction has been executed by an applet on a Secure Element, it may need to inform the application layer. To do this, an applet may trigger an event known as "EVT_TRANSACTION". This HCI event will be sent to the NFC Controller over SWP line. The NFC Controller will then forward this event to the device application processor where the event may trigger an authorized registered mobile application.

How to register a mobile application including the exact mechanism depends on the mobile OS used. This section intends to define the content of this event message and the main principles for its management.

The event message holds the following information:

- *SEName* (mandatory) reflecting the originating SE. It must be compliant with SIMalliance Open Mobile API naming convention and below complementary requirement in case of UICC, using types which are appropriate to the OS programming environment.
- *AID* (mandatory) reflecting the originating SE (UICC) applet identifier if available
- *Parameters (mandatory)* holding the payload conveyed by the HCI event EVT_TRANSACTION if available
- When *AID* is omitted from the URI, application component are registered to any "EVT_TRANSACTION" events sent from the specified Secure Element.

TS26_NFC_REQ_069	For UICC, *Secure Element Name* SHALL be *SIM[smartcard slot]* (e.g. SIM/SIM1, SIM2… SIMn).
TS26_NFC_REQ_070	For embedded SE, *Secure Element Name* SHALL be *eSE[number]* (e.g. eSE/eSE1, eSE2, etc.).
TS26_NFC_REQ_071	The device SHALL support HCI event EVT_TRANSACTION as per ETSI TS 102 622.
TS26_NFC_REQ_072	The OS implementation SHALL provide a mechanism to inform authorised OS applications of Transaction Events and this SHALL include the Secure Element name and the AID of the applet which triggered the transaction and PARAMETERS holding the payload conveyed by the HCI EVT_TRANSACTION event.
TS26_NFC_REQ_073	VOID
TS26_NFC_REQ_074	VOID
TS26_NFC_REQ_144	Across all the different system components and the APIs exposed to the developers, OS/Framework SHALL make available only existing Secure Elements and SHALL name them in a coherent way (i.e. using the same Secure Element names for OMAPI, GSMA APIs and Transaction events).

6.5 Remote Management of NFC services

6.5.1 Mobile Device APN Management Requirements

TS26_NFC_REQ_075	For mobile devices supporting multiple APNs, the device SHALL be able to set-up an OTA channel with the Active UICC Profile using the APN information that is provided in the OPEN CHANNEL command.
TS26_NFC_REQ_076	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 device default APN.
TS26_NFC_REQ_077	If the APN information provided by the network in the OPEN CHANNEL command is empty the device SHALL always use the device default APN.

6.5.2 UICC Remote Management (Access to UICC in connected mode) requirements

TS26_NFC_REQ_078	The mobile device SHALL support BIP in UICC client mode for UDP.
TS26_NFC_REQ_079	The mobile device SHALL support BIP in UICC client mode for TCP.
TS26_NFC_REQ_080	The mobile device SHALL support two concurrent channels, BIP in UICC client mode.
TS26_NFC_REQ_081	The mobile device SHALL support the SMS push (per ETSI TS 102 226 and 3GPP TS 31.116) to establish an open BIP channel as per ETSI TS 102 223 Open Channel Command.
TS26_NFC_REQ_120	The device SHALL support the BIP session regardless of incoming or outgoing calls, incoming or outgoing MMS, SMS. Note: This is not applicable if the device is on a 2G network.

6.6 Security

Access API & Secure Element Access Control Requirements

The main objective of the Access Control mechanism is to protect communication with the Secure elements.

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

TS26_NFC_REQ_082	Open OS devices SHALL provide access control as per GlobalPlatform, Secure Element Access Control specification for each available SE.
TS26_NFC_REQ_082.1	The access rules & associate caching mechanism, a.k.a Access Control Enforcer, SHALL be specific to each SE.
TS26_NFC_REQ_083	When no access control data (files or applets) is found on a SE the OS SHALL deny access to this SE.
TS26_NFC_REQ_121	Access Control Enforcer SHALL check if Access Rules has been updated only when a new logical channel is open. All rules SHALL stay valid until the channel is closed.
TS26_NFC_REQ_122	Access Control Enforcer SHALL cache the rules from the Secure Element (ARF mechanism or ARA with GET DATA[ALL]).
TS26_NFC_REQ_122.1	Access Control Enforcer cache SHALL be rebuilt when the device is switched on or the Secure Element is powered on.
TS26_NFC_REQ_122.2	When the Access Control Enforcer checks if the Access Rules have been updated, the cache SHALL only be refreshed if the “Refresh Tag” is updated.
TS26_NFC_REQ_163	The device SHALL not log any APDU or AID exchanged in a communication with an applet located in an SE (UICC, eSE, …).

TS26_NFC_REQ_169	The Access Control Enforcer SHALL be able to parse rules with unknown or missing tags.
TS26_NFC_REQ_169.1	The Access Control Enforcer SHALL ignore rules with unknown or missing tags and SHALL process the rest of the ruleset.

6.6.1 NFC Event & Access Control requirements

"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.

The NFC stack shall therefore implement GlobalPlatform Secure Element Access Control specification 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.

TS26_NFC_REQ_084	The OS implementation SHALL support the use of the GlobalPlatform Secure Element Access Control enforcer to manage Transaction Events originating from a Secure Element and SHALL ensure that this event is made available only to authorised OS applications.
TS26_NFC_REQ_085	The device 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 and there is no rule explicitly granting the NFC event permission.

6.6.2 Protecting APIs with Operator Certificate

TS26_NFC_REQ_145	When OS uses certificates for signing applications, any device implementing GSMA API SHALL implement a mechanism based on certificates stored in the Active UICC Profile for protecting sensitive APIs as illustrated in the figure below.

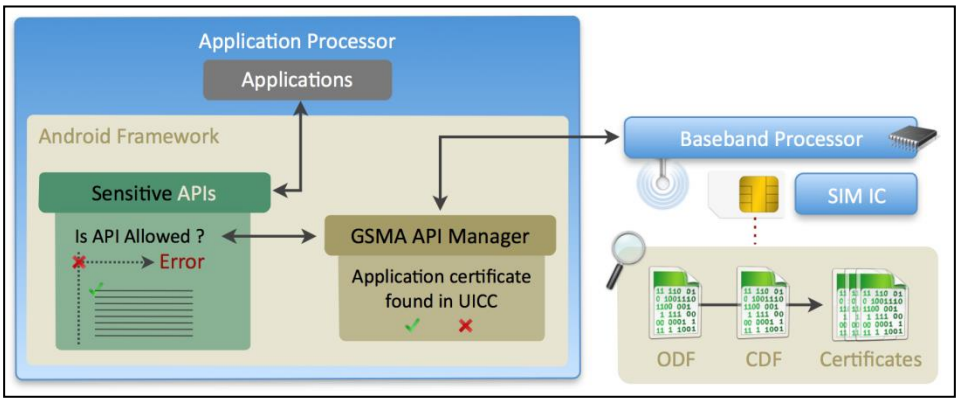


Figure 5**: Protecting sensitive APIs with an Operator Certificate**

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).

TS26_NFC_REQ_145.1	Each time the device is booting or a Secure Element containing PKCS#15 certificates is powered on, the API Protection mechanism SHALL cache the PKCS#15 certificates from all the Secure Elements.
TS26_NFC_REQ_145.2	VOID
TS26_NFC_REQ_145.3	Any sensitive API SHALL check if the application has been signed with one of the certificates stored in the Protection mechanism cache 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.

6.7 SCWS support

TS26_NFC_REQ_086	VOID

6.8 Card Application Toolkit Support

The following requirements list the minimum letter classes’ support for NFC device.

TS26_NFC_REQ_087	A device which implements Rel-11 or earlier of ETSI TS 102 223 SHOULD support the letter class “c” with the following command and events: • Proactive command: LAUNCH BROWSER • Event download: Browser termination event • Event download: Browsing status event
TS26_NFC_REQ_087.1	A device which implements Rel-12 or later of ETSI TS 102 223 SHOULD support the letter class “ab” with the following command: • Proactive command: LAUNCH BROWSER
TS26_NFC_REQ_088	The device SHALL support the letter class “e” with the following commands and events[4]: • Proactive command: OPEN CHANNEL (UICC in client mode and with the support of UDP/TCP bearer) • 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
TS26_NFC_REQ_088.1	For OPEN CHANNEL related to Default (network) Bearer, the device SHALL also support an optional Network access name (APN) occurring

[4] Note: Letter class **e** (BIP commands and events set) is related to the requirement of section 6.5.2. (UICC Remote Management (Access to UICC in connected mode) requirements).

	after the Buffer size. If supplied, the Network Access Name provides information to the device necessary to identify the Gateway entity which provides interworking with an external packet data network.
TS26_NFC_REQ_089	The device SHALL support the letter class “l” with the following command: • Proactive command: ACTIVATE
TS26_NFC_REQ_090	The device SHOULD support the letter class “m” with the following command and event: • Event download: HCI connectivity event
TS26_NFC_REQ_091	The device SHOULD support the letter class “r” with the following commands and events: • Proactive command: CONTACTLESS STATE CHANGED • Event download: Contactless state request

6.9 Platform Dependent Properties

This is intended to provide a mechanism to query for necessary platform dependent characteristics such as:

- number of secure elements supported
- use of battery low mode vs. battery off mode
- support for NFC polling with backlight off
- support for NFC listening with backlight off

TS26_NFC_REQ_092	An API MAY be provided to enable a calling application to query implementation specific choices and information. The information item made available provided by this API is referred to in subsequent clauses as a ‘property’[5]
TS26_NFC_REQ_092.01	The value returned by a property query is read-only. It SHALL not be possible for a caller to change the value returned in a property query. The value returned SHALL not change while the OS is running, but MAY change after a reboot or system update.
TS26_NFC_REQ_092.02	A property SHALL be provided that returns the version of device requirements supported. For implementations conforming to the requirements in this version of the document, the value 13000 SHALL be returned as a platform-appropriate integer value of not less than 32 bits precision.
TS26_NFC_REQ_092.03	VOID
TS26_NFC_REQ_092.04	A property SHALL be provided that returns TRUE on devices supporting operation in Battery Low mode, otherwise it SHALL return FALSE.
TS26_NFC_REQ_092.05	A property SHALL be provided that returns TRUE on devices supporting operation in Battery Power Off mode, otherwise it SHALL return FALSE.
TS26_NFC_REQ_092.06	A property SHALL be provided that returns TRUE on devices supporting NFC Forum Type 3 Tag card emulation on the device host, otherwise it SHALL return FALSE.

[5] Note: This requirement does not take precedence above related mandatory requirements. Rather it facilitates to provide confirmation to calling applications that the requested functions are indeed supported.

TS26_NFC_REQ_092.07	A property SHALL be provided that returns TRUE on devices supporting Card Emulation for FeliCa on UICC, otherwise it SHALL return FALSE.
TS26_NFC_REQ_092.08	A property SHALL be provided that returns TRUE on devices supporting MIFARE Classic Reader/Writer, otherwise it SHALL return FALSE.
TS26_NFC_REQ_092.09	A property SHALL be provided that returns TRUE on devices supporting MIFARE DESFire Reader/Writer, otherwise it SHALL return FALSE.
TS26_NFC_REQ_092.10	A property SHALL be provided that returns TRUE on devices supporting OMAPI features, otherwise it SHALL return FALSE.
TS26_NFC_REQ_092.11	A property SHALL be provided that returns TRUE on devices supporting HCI/SWP functionality between the CLF & the UICC, otherwise it SHALL return FALSE.

Note: Version Numbering

Each revision of this document defines a version number for the Handset Requirements contained herein. Implementations are required, by TS26_NFC_REQ_092.02, to return the version number of the requirements supported.

The convention chosen is to return the version number as an integer value, with the precise type used being one which is natural to the OS platform (with the proviso of a minimum of 32 bits precision).

The overall version number is calculated based on a major version number and minor version number. The major version number is a value between 5 and 2147483 and the minor number is a value between 0 and 999. The calculation is:

- version_number = (major_version_number * 1000) + minor_version_number

For this document, the major version number is ‘13’ and the minor version number is ‘0’.

Document versioning should be managed in such a way that an application can use the major version number alone to determine the supported requirements and API. Minor version number changes should be used for backward compatible changes and clarifications only.

6.10 Personalization of the eSE

TS26_NFC_REQ_184	The eSE SHOULD comply with GlobalPlatform Secure Element Configuration.
TS26_NFC_REQ_185	The Issuer Security Domain on the eSE SHALL be personalized with CIN (Card Image Number) and IIN (Issuer Identification Number) as defined in GlobalPlatform Secure Element Configuration.

7 Android Operating System

7.1 NFC Device Architecture

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 Active UICC Profile.

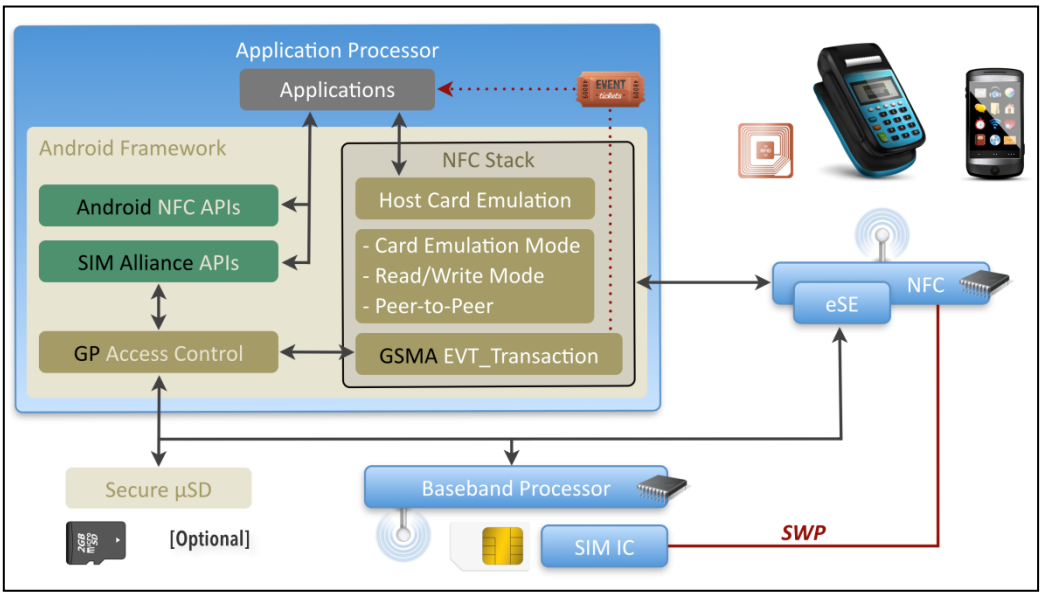


Figure 6**: Android NFC software stack**

Previous figure gives an overview of a possible Android implementation as an example showing how this requirement can be mapped to an OS.

On Android the architecture could be 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.

On Android, such a background component might rely on a RIL extension for accessing the UICC and on some specific libraries, for communicating with any embedded secure elements.

7.2 Core Required NFC Features

TS26_NFC_REQ_181	If the GSMA API is implemented, then all related GSMA API requirements SHALL be implemented.
TS26_NFC_REQ_123	If GSMA API is implemented then the API SHALL be included in the following jar file: “com.gsma.services.nfc.jar” Note: Library SHALL be built as a shared library in order to be used with the following tag: <uses-library android:name="com.gsma.services.nfc” android:required="false"/>

7.2.1 NFC Controller management API

<table>
<tr><td>TS26_NFC_REQ_093</td><td>There MAY be an API (as described below) to check if the NFC Controller is enabled and to ask the system to enable the NFC functionality. User input SHALL be required to enable NFC. This UI dialogue SHALL be generated by the OS and not by the calling application.
Note: Refer to the Javadoc linked to this document for more details.
<table>
<tr><th colspan="2">“com.gsma.services.nfc.NfcController”</th></tr>
<tr><th>Classes</th><th>Methods</th></tr>
<tr><td>nested classes (NfcController.Callbacks)</td><td>All methods except “onCardEmulationMode”</td></tr>
<tr><td>NfcController</td><td>isEnabled, enableNfcController</td></tr>
</table>
Note: this requirement fulfils the generic requirements TS26_NFC_REQ_022.</td></tr>
<tr><td>TS26_NFC_REQ_146</td><td>The device MAY implement the method as described below:
Note: Refer to the Javadoc linked to this document for more details.
<table>
<tr><th colspan="2">“com.gsma.services.utils.Handset”</th></tr>
<tr><th>Classes</th><th>Methods</th></tr>
<tr><td>Handset</td><td>getAvailableSecureElements</td></tr>
</table>
Note: this requirement fulfils the generic requirements TS26_NFC_REQ_059 and TS26_NFC_REQ_060.</td></tr>
</table>

7.2.2 Card Emulation mode requirements

No specific requirement, see Generic Device Requirements.

7.2.3 Reader/writer & TAG management requirements

No specific requirement, see Generic Device Requirements.

7.3 Secure Element Access & Multiple Secure Elements Management

7.3.1 Mobile Device Modem requirements

No specific requirement, see Generic Device Requirements.

7.3.2 Secure Element Access API requirements

TS26_NFC_REQ_124	Before Android P the device SHALL implement SIMalliance Open Mobile API (using the namespace “org.simalliance.openmobileapi”), but only the transport layer. The service layer SHALL not be implemented. Note 1: this requirement fulfils the generic requirements TS26_NFC_REQ_047, TS26_NFC_REQ_047.1, TS26_NFC_REQ_047.2 TS26_NFC_REQ_047.3 and TS26_NFC_REQ_183. Note 2: see TS26_NFC_REQ_047.3 for the values of EventTypes.

TS26_NFC_REQ_186	This requirement is applicable from Android P onwards. The device SHALL implement GlobalPlatform Open Mobile API using the "android.se.omapi" namespace. Note: this requirement fulfils the generic requirements TS26_NFC_REQ_047, TS26_NFC_REQ_047.1, TS26_NFC_REQ_047.2 TS26_NFC_REQ_047.3 and TS26_NFC_REQ_183.
TS26_NFC_REQ_125	SIMalliance Open Mobile API and GlobalPlatform Access Control SHALL be initialized and ready to use when the BOOT_COMPLETED intent is sent.
TS26_NFC_REQ_125.1	As system components, SIMalliance Open Mobile API and GlobalPlaform Access Control SHALL be independent of the state of the SIM and initialization SHALL be completed even when the SIM is locked by a PIN code.

7.3.3 Multiple CEE support

All generic device requirements are applicable in addition to below specific requirements for Android.

7.3.3.1 VOID

TS26_NFC_REQ_126	VOID

7.3.3.2 Multiple Active CEE model

The following requirements only apply where a device supports Multiple Active CEEs model.

TS26_NFC_REQ_094	When a mobile application is registering an AID-based or non AID-based service (statically or dynamically) it SHALL be able to state the target CEE using an OS mechanism (manifest, API, …).
TS26_NFC_REQ_094.1	The following extension SHALL be supported in the manifest <extensions xmlns:android="http://www.gsma.com" android:description="@string/servicedesc"> <se-ext-group> <se-id name="XXX" /> </se-ext-group> <AID-based>boolean</AID-based> </extensions> Where: - *se-id name* SHALL be set as described in TS26_NFC_REQ_069 and TS26_NFC_REQ_070 ; the uniqueness of the naming SHALL be ensured by the OS as described in TS26_NFC_REQ_144. - For *se-id name* field, following values SHALL be accepted by the device: - "*SIM*" and "*SIM1*" for SIM slot 1 - "*eSE*" and "*eSE1*" for embedded Secure Element 1 - *AID-based* SHALL be set to true in case a SE application is

<table>
<tr><td></td><td>compliant with ISO 7816-4 and false in all other cases
Note: When a mobile application is declaring a se-id name that is not existing on the device the registration shall be ignored.</td></tr>
<tr><td>TS26_NFC_REQ_094.2</td><td>If the extension is not declared in the manifest of the application, then the following default values SHALL apply (for backward compatibility):
- <se-id name="SIM1"/>
- <AID-based>true</AID-based></td></tr>
<tr><td>TS26_NFC_REQ_133</td><td>The device SHALL support an OS mechanism that allows applications to statically register NFC application by list of AIDs.
Note: this requirement fulfils the generic requirement TS26_NFC_REQ_061.</td></tr>
<tr><td>TS26_NFC_REQ_127</td><td>The device MAY implement classes and methods as described below.
Note: Refer to the Javadoc linked to this document for more details.

<table>
<tr><th colspan="2">"com.gsma.services.nfc.NfcController"</th></tr>
<tr><th>Classes</th><th>Methods</th></tr>
<tr><td>nested classes (NfcController.Callbacks)</td><td>All methods except "onCardEmulationMode"</td></tr>
<tr><td>NfcController</td><td>All methods except "disableCardEmulationMode, enableCardEmulationMode"</td></tr>
</table>
<table>
<tr><th colspan="2">"com.gsma.services.nfc.AIDGroup"</th></tr>
<tr><th>Classes</th><th>Methods</th></tr>
<tr><td>AIDGroup</td><td>All methods</td></tr>
</table>
<table>
<tr><th colspan="2">"com.gsma.services.nfc.OffHostService"</th></tr>
<tr><th>Classes</th><th>Methods</th></tr>
<tr><td>OffHostService</td><td>All methods</td></tr>
</table></td></tr>
<tr><td>TS26_NFC_REQ_127.1</td><td>API SHALL also manage "Off-Host" services registered statically through the Manifest.</td></tr>
<tr><td>TS26_NFC_REQ_127.2</td><td>All the services/group created through these APIs SHALL stay persistent (still available after a power off/on of the device).</td></tr>
<tr><td>TS26_NFC_REQ_128</td><td>The device MAY handle a "Long Press" on all "Tap&Pay" menu entries.</td></tr>
<tr><td>TS26_NFC_REQ_128.1</td><td>When a "Long Press" is done on a "Tap&Pay" menu entry, the device SHALL only send an intent to the application that has created this entry.

Note: the primary anticipated usage of the intent is to allow the application to display further information to the user about its services. Other usages are also possible.</td></tr>
<tr><td>TS26_NFC_REQ_128.2</td><td>Device SHALL send intent using Android "startActivity(…) API.</td></tr>
</table>

TS26_NFC_REQ_128.3	Intent SHALL be built as follows: • action: com.gsma.services.nfc.SELECT_DEFAULT_SERVICE • no category
TS26_NFC_REQ_134	The device SHALL provide an additional menu entry in “Settings” in order to enable/disable group of AIDs (as defined by Android) belonging to the category “Other”. A group of AIDs SHALL only be enabled/disabled as a single unit.
TS26_NFC_REQ_134.1	When there is an overflow in the NFC router table, the menu entry SHOULD display: • A banner representing the groups of AIDs belonging to the category “Other” (and optionally the group description) with its current status (enabled/disabled) and a way for disabling/enabling it • A visual indication representing the NFC Controller capacity and showing 1. The space used by the selected group 2. If enablement of the selected group can fit with the remaining space of the NFC Controller routing table
TS26_NFC_REQ_134.2	The menu entry SHOULD be hidden to the end user until the first time a NFC Service cannot be added in the NFC Controller routing table
TS26_NFC_REQ_134.3	When the menu entry is opened, the status of NFC services group displayed by the menu entry SHALL reflect the actual status of the current NFC Controller routing table.
TS26_NFC_REQ_134.4	When there is no overflow in the NFC routing table and if the menu entry is available to the end user, the menu SHOULD not allow the end user to disable the AID groups.
TS26_NFC_REQ_172	The device SHALL consider that non-AID based services are conflicting as soon as they are associated to different Off-Host entities.
TS26_NFC_REQ_170	In case the device detects a conflict between non-AID based contactless services (see TS26_NFC_REQ_172), the device SHALL display a menu entry in “Settings” in order to list impacted contactless services and the user SHALL be directed to the menu.
TS26_NFC_REQ_170.1	The menu entry SHALL present the conflicting services to the end user with an option to select which service(s) to be active. Only one set of service(s) (which are not conflicting with each other) SHALL be active at any one time.
TS26_NFC_REQ_170.2	The menu entry SHOULD be hidden to the end user in case there is no conflicting services.
TS26_NFC_REQ_170.3	VOID
TS26_NFC_REQ_171	VOID
TS26_NFC_REQ_180	If the end user changes the default non-AID based contactless service via the menu described in TS26_NFC_REQ_170, the device SHALL reconfigure the NFC Controller with the contactless parameters configured in the Off-Host linked to the newly activated service.

<table>
<tr><td>TS26_NFC_REQ_135</td><td>When an application is trying to register new AIDs belonging to the category “Other” and there is no automatic solution to solve any routing table overflow (as defined in REQ_143), the device SHALL:
Inform the end user that some NFC Services proposed by the application cannot be used. A message SHALL provide the description of the group(s) of AIDs (android:description) which cannot be activated
Propose the end user should disable some previously installed NFC services using the feature described in TS26_NFC_REQ_134 in order to free some NFC Controller routing table space to be able to register all AIDs needed by the current application
When one AID from a group of AIDs cannot be added in the NFC Controller routing table, the entire group of AIDs SHALL not be enabled.</td></tr>
<tr><td>TS26_NFC_REQ_136</td><td>When a customer is selecting a service from the “Tap&Pay” menu and there is no automatic solution to solve any routing table overflow (as defined in REQ_143), the device SHALL:
Inform the end user that activation of the selected NFC services cannot be performed
Propose the end user should disable some previously installed NFC services using the feature described in TS26_NFC_REQ_134 in order to free some NFC Controller routing table space
If the end user doesn’t disable enough NFC services to allow activation of the selected “Tap&Pay” menu, previous “Tap&Pay” entry SHALL stay active and the end users selection is cancelled.</td></tr>
<tr><td>TS26_NFC_REQ_147</td><td>In the “Tap&Pay” menu, the user selection has precedence and the behaviour of the device is consistent across different handset states.
The following scenarios SHALL be applied.

Note 1: If Android is changing the behaviour then this requirement will change accordingly.

Note 2: GSMA strongly recommend that service providers register the Off-Host AIDs in the Android OS as defined by Android.

<table>
<tr><th>Scenario</th><th>Screen ON (Screen Lock/Unlock)</th><th>Screen OFF</th><th>Switched Off</th></tr>
<tr><td>Default AID route is set to HCE.
No App in the Payment category has been installed.</td><td>For any AID which is not registered in the "Other" category, contactless selection fails with error code '6A82'.</td><td>For any AID which is not registered in the "Other" category, contactless selection fails with error code '6A82'.</td><td>For any AID which is not registered in the "Other" category, contactless selection fails with error code '6A82'.</td></tr>
</table>
</td></tr>
</table>

	Default AID route is set to Off-Host No App in the Payment category has been Installed.	For any AID which is not registered in the "Other" category, APDUs go to Off-Host.	For any AID which is not registered in the "Other" category, APDUs go to Off-Host.	For any AID which is not registered in the "Other" category, APDUs go to Off-Host.
	Default AID route is set to Off-Host The user selected an Off-Host-based service in Tap&Pay menu.	APDUs intended for the selected Off-Host service go to Off-Host.	APDUs intended for the selected Off-Host service go to Off-Host.	APDUs intended for the selected Off-Host service go to Off-Host.
	Default AID route is set to Off-Host The user selected a HCE-based service in Tap&Pay menu.	APDUs intended for the selected HCE service go to HCE.	For APDUs intended for the selected HCE service, contactless selection fails with error code '6A82' OR APDUs go to HCE.	For APDUs intended for the selected HCE service, contactless selection fails with error code '6A82'.
	Default AID route is set to HCE The user selected a HCE-based service in Tap&Pay menu.	APDUs intended for the selected HCE service go to HCE.	For APDUs intended for the selected HCE service, contactless selection fails with error code '6A82 OR APDUs go to HCE'.	For APDUs intended for the selected HCE service, contactless selection fails with error code '6A82'.
	Default AID route is set to HCE The user selected an Off-Host-based service in Tap&Pay menu.	APDUs intended for the selected Off-Host service go to Off-Host.	APDUs intended for the selected Off-Host service go to Off-Host.	APDUs intended for the selected Off-Host service go to Off-Host.
TS26_NFC_REQ_148	The device SHALL not change the default AID route in response to changes in device state (such as screen off, power off).			
TS26_NFC_REQ_148.1	The same behaviour SHALL be implemented when the mobile device is set in Flight Mode with NFC ON.			
TS26_NFC_REQ_149	AID Conflict resolution mechanism defined by Android SHALL be applied to both HCE applications and Off-Host applications (UICC, eSE).			

7.4 UI Application triggering requirements

The same generic requirements are applicable to Android platform with the following requested implementation:

TS26_NFC_REQ_129	Before Android P, a Transaction Event (EVT_TRANSACTION) SHALL be triggered based on the following information:

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

Table: Intent Details for TRANSACTION Event (EVT_TRANSACTION)

TS26_NFC_REQ_096	Before Android P, transaction event data SHALL be set in the following extended field:

com.gsma.services .nfc.extra.AID `ByteArray`	Contains the card “Application Identifier” *[optional]*
com.gsma.services .nfc.extra.DATA `ByteArray`	Payload conveyed by the HCI event “EVT_TRANSACTION”

Table: Transaction Event (EVT_TRANSACTION) data

TS26_NFC_REQ_187	This requirement is applicable from Android P onwards. Transaction Event is provided natively by Android. A Transaction Event (EVT_TRANSACTION) SHALL be triggered based on the following information:

Action	*android.nfc.action.**TRANSACTION_DETECTED***
Mime type	-
URI	*nfc://secure:0/<SEName>/<AID>* - *SEName* reflects the originating SE It must be compliant with SIMalliance Open Mobile API - *AID* reflects the originating UICC applet identifier

Table: Intent Details for *TRANSACTION_DETECTED*

TS26_NFC_REQ_188	This requirement is applicable from Android P onwards. Transaction event data SHALL be set in the following extended field:

android.nfc.extra.AID `ByteArray`	Contains the card “Application Identifier”
android.nfc.extra.DATA `ByteArray`	Payload conveyed by the HCI event “EVT_TRANSACTION” *[optional]*
android.nfc.extra.SECURE_ELEMENT_NAME `String`	Indicates the Secure Element on which the transaction occurred. eSE1...eSEn for Embedded Secure Elements, SIM1...SIMn for UICC, etc.

Table: *TRANSACTION_DETECTED* data

In Android, two mechanisms may be used for receiving this event.

Note: From version 11 of TS.26 the GSMA APIs are deprecated: For later releases “Multicast” may not be possible.

TS26_NFC_REQ_098	Before Android P, the device SHALL support “Unicast” mode for receiving the “EVT_TRANSACTION”.
TS26_NFC_REQ_182	Before Android P, the device MAY support “Multicast” mode for receiving the “EVT_TRANSACTION”.
TS26_NFC_REQ_097	Before Android P, by default, each time the device is powered on, “Unicast” mode SHALL be activated.
TS26_NFC_REQ_099	Before android P, for switching from “Unicast” to “Multicast”, the device MAY support an API as described below. “com.gsma.services.utils.Handset” **Classes** / **Methods** Handset / enableMultiEvt_transactionReception This is considered a sensitive API.
TS26_NFC_REQ_189	This requirement is applicable from Android P onwards. The framework SHALL generate a send broadcast intent toward application signed with an allowed hash in access rules.

[6] Note: Refer to the Javadoc linked to this document for more details.

7.4.1 Unicast – Reception by only one activity

TS26_NFC_REQ_150	Before Android P, in “Unicast” mode, “EVT_TRANSACTION” event SHALL be received by only one activity.
TS26_NFC_REQ_150.1	Before Android P, when several activities registered for the “EVT_TRANSACTION”, the framework SHALL follow the priority scheme described below: 1. “Android:priority” level defined in the Intent Filter SHALL be compared 2. First installed application package (APK) has a higher priority Note: *Only priority <0 are possible for non privileged application, as per*

	Android AOSP code (https://android.googlesource.com/platform/frameworks/base/+/5585dd971e265cbb3a515bb759710646c5bc6f63)
TS26_NFC_REQ_150.2	Before Android P, the framework SHALL call "`startActivity(…)`"on a Context instance (Activity, Service etc.) using an explicit Intent (as defined by Android and with content as per TS26_NFC_REQ_129 and TS26_NFC_REQ_096) in order to launch the selected activity and inform it of the event.

7.4.2 Multicast – Reception by several application components (activity, services…)

TS26_NFC_REQ_151	Before Android P, the framework SHALL use "`BroadcastReceiver`" mechanism to inform the registered activities of the event.

7.5 Remote Management of NFC Services

No specific requirement, see Generic Device Requirements.

7.6 Security

TS26_NFC_REQ_130	Before Android P, the Android Framework SHALL define the following permissions for handling Open Mobile API and Transaction events: - Permission-group "**com.gsma.services.nfc.permission** " SHALL be defined in Android framework to host all NFC-related permissions as following: permission-group android:name = com.gsma.services.nfc.permission android:label = @string/permgrouplab_NFC = "NFC - Near Field Communication" android:icon = @drawable/perm_group_network android:description = @string/permgroupdesc_NFC = "Access various NFC features" android:priority = 270 - Permission "**org.simalliance.openmobileapi.SMARTCARD**" SHALL be defined in Android framework as following: android:name = org.simalliance.openmobileapi.SMARTCARD android:permissionGroup = com.gsma.services.nfc.permission android:label = @string/permlab_SmartcardService = "Access to SIM Card" android:description = @string/permdesc_SmartcardService = "Allows application to communicate with the SIM card" - Permission "**com.gsma.services.nfc.permission.TRANSACTION_EVENT**" shall be defined in Android framework as following: android:name = com.gsma.services.nfc.permission.TRANSACTION_EVENT android:permissionGroup = com.gsma.services.nfc.permission android:label = @string/permlab_nfcReceiveEvent = "NFC

	Transaction awareness" android:description = @string/permdesc_nfcReceiveEvent = "Allows application to receive information about NFC transaction performed"
TS26_NFC_REQ_130.1	Before Android P, protection level of all the permissions described in TS26_NFC_REQ_130 SHALL be set to to the same level as “android.permission.NFC”. Note: Before Marshmallow version: android.permission.NFC is set to “dangerous”. From Marshmallow version: android.permission.NFC is set to “normal”.
TS26_NFC_REQ_130.2	Before Android P, the Android Framework SHALL provide the permission-group and permissions defined in TS26_NFC_REQ_130 in the local language of the device.
TS26_NFC_REQ_190	This requirement is applicable from Android P onwards. The permission SHALL be set as following: - OMAPI does not require any specific permissions - TRANSACTION_EVENT SHALL be o android.permission.NFC_TRANSACTION_EVENT

7.6.1 Access API & Secure Element Access Control requirements

No specific requirement, see Generic Device Requirements.

7.6.2 NFC Event & Access Control requirements

The same generic requirements are applicable to Android platform with the following requested implementation:

Android permissions

TS26_NFC_REQ_131	Before Android P, the device SHALL ensure that the application has the following permission before forwarding a Transaction event to the application:

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

Table: EVT_TRANSACTION Permissions

TS26_NFC_REQ_191	This requirement is applicable from Android P onwards. The device SHALL ensure that the application has the following permission before forwarding a Transaction event to the application:

NFC Controller	android.permission.NFC
Transaction Event	android.permission.NFC_TRANSACTION_EVENT

Table EVT_TRANSACTION Permissions

Access control

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 API.

TS26_NFC_REQ_152	The NFC stack SHALL therefore use internal "*Access Control"* API 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.
TS26_NFC_REQ_152.1	If an application is registered to any "EVT_TRANSACTION", by omitting the AID in the Intent, it SHALL receive the events of any applets to those accessible using the "*Access Control"*.
TS26_NFC_REQ_152.2	If no application is authorised as per "*Access Control"* check, then the event SHALL be discarded by the framework.

7.6.3 Protecting sensitive APIs

TS26_NFC_REQ_156	If GSMA API is implemented the device SHALL authorize a mobile application to use GSMA sensitive APIs if the mobile application is: • Signed with a certificate referenced in the "Certificate Directory File" (CDF) of the PKCS#15 structure of the UICC as defined in chapter 0. Note: This requirement may be subject to modification if another solution is found for TS26_NFC_REQ_153.
TS26_NFC_REQ_153	VOID
TS26_NFC_REQ_153.1	VOID

7.7 SCWS support

No specific requirement, see Generic Device Requirements.

7.8 Card Application Toolkit Support

No specific requirement, see Generic Device Requirements.

7.9 Platform Dependent Properties

TS26_NFC_REQ_132	The device MAY implement classes and methods as described below: "com.gsma.services.utils.Handset" **Classes** / **Methods** Handset / getProperty, getVersion Note: this requirement fulfils the generic requirement TS26_NFC_REQ_092.

8 VOID

TS26_NFC_REQ_100	VOID
TS26_NFC_REQ_101	VOID
TS26_NFC_REQ_102	VOID
TS26_NFC_REQ_103	VOID
TS26_NFC_REQ_104	VOID

TS26_NFC_REQ_105	VOID
TS26_NFC_REQ_106	VOID

9 VOID

10 VOID

11 Android Wear Operating System

Requirements for Android Wear will be added in a later version of this document.

Annex A Implementation help of REQ 94.1 for multi eSE on Android

OffHost Service definition in Android Manifest

```
<service android:name=".MyserviceOffHost"
          android:exported="true"
         android:permission="android.permission.BIND_NFC_SERVICE" >
   <intent-filter>
      <action android:name="android.nfc.cardemulation.action.OFF_HOST_APDU_SERVICE"/>
      <category android:name="android.intent.category.DEFAULT"/>
   </intent-filter>
<meta-data android:name="android.nfc.cardemulation.off_host_apdu_service"
android:resource="@xml/offhost_aid"/>
<meta-data android:name="com.gsma.services.nfc.extensions" android:resource="@xml/nfc_se"/>
</service>
```

Note: the bold line is a GSMA extension.

com.gsma.services.nfc.extensions = see REQ 094.1

nfc_se XML file content example

```
<extensions xmlns:android="http://www.gsma.com" android:description="@string/servicedesc">
   <se-ext-group>
      <se-id name="XXX"/>
   </se-ext-group>
<AID-based>boolean</AID-based>

</extensions>
```

XXX can be : SIM/SIM1, SIM2, eSE/eSE1, eSE2, … (as per requirements TS26_NFC_REQ_070 and 071)

AID-based is set to:

- true for Application defining service using AID based (compliant with ISO 7816-4)
- false for Application defining service using non AID based (i.e. Mifare, Felica, …)

Annex B Document Management

B.1 Document History

Ver.	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
5.0	12th March 2014	Implemented updates to: Title, Introduction, Abbreviations, Terms of Definitions, References, Doc. structure, Figures, Requirements across the document and API specifications. Added a new numbering scheme and provided editorial updates. Marked requirements still work in progress. Removed Annexes and requirements/references as applicable.	Terminal Steering Group	K.Jordan (DT)
6.0	21 July 2014	Implemented updates to: Introduction, Abbreviations, Definitions of Terms, References, Figures, Core Required NFC Features, Secure Element Access & Multiple Secure Elements Management, UI Application triggering requirements, UICC Remote Management, Security, SCWS support, Card Application Toolkit Support, Platform Dependent Properties, Android and Blackberry OS specific sections. Removed majority of yellow markings where req. could be	Terminal Steering Group	K.Jordan, G.Printemps, DT

		confirmed. Removed Android API details and provided updated API details as separate Javadoc with Readme file. Implemented further formatting and quality updates.		
7.0	23 March 2015	Extra document release to address 2 critical issues: 1) ambiguity of the requirement 114 and 2) full AID routing table issue when installing new application on Android devices.	Terminal Steering Group	Radomír Věncek, DT
8.0	13 October 2015	General update based mainly on feedback from the NFC Handset Test Book, changes across the whole document. Includes also initial changes related to transport industry – accepted NFC Forum specifications as a baseline for device testing. Added new requirements as well as modified existing ones.	Terminal Steering Group	Radomír Věncek, DT
9.0	01 April 2016	General update fixing some known issues, removing empty OS sections (Blackberry, Windows Phone), adding few new requirements.	Terminal Steering Group	Radomír Věncek, DT
10.0	05 December 2016	General update amending some existing requirements and adding few new requirements.	Terminal Steering Group	Ruben Rico, Vodafone
11.0	12 June 2017	General update, adding eSE and Wearables as part of the document. GSMA API is marked as deprecated from this version.	Terminal Steering Group – TSG#28	Anders Olsson, Sony Mobile
12.0	05 December 2017	General update, next step in deprecating GSMA APIs. Adding clarifying requirements for Dual SIM and eSE.	Terminal Steering Group – TSG#30	Anders Olsson, Sony Mobile
13.0	04 June 2018	This version includes the following changes: ▪ **Android P** impacted requirements have been updated and some new requirements have been introduced to support the Android P implementation. ▪ **Single CEE** (Card Emulation Environment) requirements have been Voided as this is not relevant anymore.	Terminal Steering Group	Anders Olsson, Sony Mobile

		▪ **Multiple UICC "slots"** – 1 updated requirement on NFC capability for multiple UICC/eUICC slots. ▪ **GSMA APIs**: Requirements updated to reflect deprecated GSMA APIs have been made optional and changed from SHOULD to MAY.		

Other Information

Type	Description
Document Owner	GSMA TSG
Editor / Company	Anders Olsson (Sony Mobile), Paul Gosden (GSMA)

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