TRUSTED CONNECTIVITY ALLIANCE

Mobile Near Field Communication (Mobile NFC) Stepping Stones

Version 1.0.0

Published by simalliance now Trusted Connectivity Alliance

June 2012

Table of Contents

Figure index

1. Introduction

The Mobile Near Field Communication (Mobile NFC) technology is a short-range wireless communication technology allowing the exchange of data between a Mobile device and another NFC device. This powerful enabler defines new important business scenarios of identification, payments, secure communication where the USIM card plays his important role of secure element at the hearth of the telecommunication ecosystem.

This is strengthened by the secure and reliable GSM network to offer highly flexible services such as contactless public transport ticketing and payment systems or contactless bonus systems for customers.

The introduction of new services involves new players coming in the Telecommunication arena: payment, identification and service providers may create new added value on the USIM. New players means new opportunities but also more partners involved, and hence the need of Interoperability in the Mobile NFC ecosystem is even higher than the past. SIMalliance Interoperability Working Group aims by this document to assist the usage of the Card components in an Interoperable way.

The present document is based upon the latest ETSI and GlobalPlatform specifications, main enablers for this kind of technology.

The target audience of this guide is Network Operators, Wireless Service Providers and anyone interested in interoperable UICC service development.

1.1 Acknowledgements

A lot of people have been contributing the document, but a special thank goes to the people active in the SIMalliance Interoperability Working Group, including:

Eric Laffont (Comprion), Laurence Bringer, Christophe Dubois, Basile Fourcade (Gemalto), Michael Schnellinger, Nils Nitsch (G&D), Amedeo Veneroso (Incard STMicroelectronics), René Huxol, Marc-Andre Wiese (Morpho), Claudio Restante, Gaetano Esposito (Movenda), Fabien Cordier (Oberthur Technologies)

1.2 Reference Documentation

Entity	Reference	Title
ISO	[1] **ISO/IEC 7816-3**	"Information technology – Identification cards – Integrated circuit(s) cards with contacts – Part 3: Electronic signals and transmission protocols"
	[2] **ISO/IEC 7816-4**	"Identification cards — Integrated circuit cards — Part 4: Organization, security and commands for interchange"
	[3] **ISO/IEC 13239**	"Information technology – Telecommunications and information exchange between systems – High-level data link control (HDLC) procedures"
	[4] **ISO/IEC 14443-2**	"Identification cards – Contactless integrated circuit(s) cards – Proximity cards – Part 2: Radio frequency power and signal interface"
	[5] **ISO/IEC 14443-3**	"Identification cards – Contactless integrated circuit(s) cards – Proximity cards – Part 3: Initialization and anti-collision"
	[6] **ISO/IEC 14443-4**	"Identification cards – Contactless integrated circuit cards – Proximity cards – Part 4: Transmission protocol"

Entity	Reference	Title
	[7] **ISO/IEC 15693**	"Identification cards – Contactless integrated circuit(s) cards – Vicinity cards"
	[8] **ISO/IEC 18092**	"Information technology – Telecommunications and information exchange between systems – Near Field Communication – Interface and Protocol (NFCIP-1)"
ETSI	[9] **ETSI TS 102 127**	"Smart Cards; Transport protocol for CAT applications; Stage 2"
	[10] **ETSI TS 102 221**	"Smart Cards; UICC-Terminal interface; Physical and logical characteristics"
	[11] **ETSI TS 102 223**	"Smart Cards; Card Application Toolkit (CAT)"
Reentered	[12] **ETSI TS 102 224**	"Smart Cards; Security mechanisms for UICC based Applications – Functional requirements".
	[13] **ETSI TS 102 225**	"Smart Cards; Secured packet structure for UICC based applications" (OTA)
	[14] **ETSI TS 102 226**	"Smart Cards; Remote APDU structure for UICC based applications" (OTA)
	[15] **ETSI TS 102 241**	"Smart Cards; UICC Application Programming Interface (UICC API) for Java Card™"
	[16] **ETSI TS 102 312**	"Near Field Communication Interface and Protocol-2 (NFCIP-2) "
Reentered	[17] **ETSI TS 102 483**	"Smart Cards; UICC-Terminal interface; Internet Protocol connectivity between the UICC and terminal"
	[18] **ETSI TS 102 588**	"Smart Cards; Application invocation Application Programming Interface (API) by a UICC webserver for Java Card™ platform"
See 6.1	[19] **ETSI TS 102 600**	"Smart Cards; UICC-Terminal interface; Characteristics of the USB interface"
	[20] **ETSI TS 102 613**	"Smart Cards; UICC – Contactless Front-end (CLF) Interface; Part 1: Physical and data link layer characteristics"
	[21] **ETSI TS 102 622**	"Smart Cards; UICC – Contactless Front-end (CLF) Interface; Host Controller Interface (HCI)"
	[22] **ETSI TS 102 694-1**	"Smart Cards; Test specification for the Single Wire Protocol (SWP) interface; Part 1: Terminal features"
	[23] **ETSI TS 102 694-2**	"Smart Cards; Test specification for the Single Wire Protocol (SWP) interface; Part 2: UICC features"
	[24] **ETSI TS 102 695-1**	"Smart Cards; Test specification for the Host Controller Interface (HCI) Part 1: Terminal features "
	[25] **ETSI TS 102 695-2**	"Smart Cards; Test specification for the Host Controller Interface (HCI) Part 2: UICC features "
	[26] **ETSI TS 102 695-3**	"Smart Cards; Test specification for the Host Controller Interface (HCI) Part 3: Host Controller features "

Entity	Reference	Title
	[27] **ETSI TS 102 705**	"Smart Cards; Contactless API for Java Card(TM) for the UICC platform"
Java Service Requests (Java Community Process)	[28] **JSR 000177**	"Security and Trust Services API for J2ME(TM) " (SATSA)
	[29] **JSR 000257**	"Contactless Communication API"
Java Card Specs	[30] **Java Card 3.0.1**	
	[31] **Java Card 2.2.2**	
GlobalPlatform	[32] **GlobalPlatform 2.2.1, Core specification**	(Including "Errata and precision list" Version 0.2.)
	[33] **GlobalPlatform 2.2, Amendment A**	"Confidential Card Content Management"
	[34] **GlobalPlatform 2.2, Amendment B**	"Remote Application Management over HTTP"
	[35] **GlobalPlatform 2.2, Amendment C**	"Contactless Services"
	[36] **GlobalPlatform UICC configuration**	"UICC Configuration"
new	[37] **GlobalPlatform Card Specification v2.1.1**	(March 2003)
3GPP	[38] **3GPP TS 31.115**	"Core Network & Terminals: Secured packet structure for (Universal) Subscriber Identity Module (U)SIM Toolkit applications
	[39] **3GPP TS 31.116**	"Core Network & Terminals: Remote APDU Structure for (U)SIM Toolkit applications
	[40] **3GPP TS 23.040**	"Core Network and Terminals; Technical realization of the Short Message Service (SMS)"
	[41] **3GPP TS 24.090**	"Group Core Network and Terminals; Unstructured Supplementary Service Data (USSD)"
OMA	[42] **OMA TS Smartcard Web Server v1.1.1**	
SIMalliance Steppingstones	[43] **SteppingStones_R7_v1.0.0**	Gives an overview based upon the ETSI Release 7 framework that references GlobalPlatform 2.1 of the Mobile Near Field Communication (Mobile NFC) technology.
SIMalliance SCWS Stepping Stones	[44] **SteppingStones_SCWS_v.1.0.0**	Analyzes and collects all information related to SCWS services and their remote management.
IETF	[45] **RFC 2616**	Hypertext Transfer Protocol -- HTTP/1.1
	[46] **RFC 4279**	Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)
	[47] **RFC 2246**	The TLS protocol

1.3 Abbreviations

ACT	Activation Protocol
AID	Application Identifier
APDU	Application Protocol Data Unit
API	Application Programming Interface
APSD	Application Provider Security Domain
ATR	Answer To Reset
ATS	Answer To Select
BIP	Bearer Independent Protocol
CAT	Card Application Toolkit
CB	Chaining Bit
CGM	Cumulative Granted Memory
CLF	Contactless Front End
CLT	Contactless Tunnelling
CRC	Cyclic Redundancy Check
EOF	End of Frame
EPOS	electronic point-of-sale
FWI	Frame Waiting Time Integer
FWT	Frame Waiting Time
GP	GlobalPlatorm
HCI	Host Controller Interface
HCP	Host Controller Protocol
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
J2ME	Java 2 Mobile Edition
JCRE	Java Card Runtime Environment
JSR	Java Specification Request
ME	Mobile Equipment
MNO	Mobile Network Operator
NFC	Near Field Communication
OTA	Over The Air
PICC	Proximity Card
PSK-TLS	PreShared Key Transport Layer Security
RAM	Remote Application Management
RF	Radio Frequency
RFM	Remote File Management
SAK	Select Acknowledge
SCWS	Smart Card Web Server
SHDLC	Simplified High Level Data Link Control
SIM	Subscriber Identity Module
SFGI	Startup Frame Guard Time Integer
SFGT	Startup Frame Guard Time
SOF	Start of Frame
SWP	Single Wire Protocol
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TSM	Trusted Service Manager
UICC	Universal Integrated Circuit Card
WTX	Waiting Time eXtension

2. General architecture

Mobile NFC is a field which offers a huge amount of opportunities and use cases where interoperability is a key point for success. This chapter gives an introduction to solutions and possibilities regarding the use of mobile NFC. It is followed by an overview of mobile NFC technology.

A mapping between the different terminologies used at different specification authorities is located at the end of this chapter. This document shall help to establish a common understanding of the mobile NFC technology, which is described in a variety of different specifications and standardisation bodies.

2.1 Benefits of Mobile NFC

Customers can benefit from many technologies which can be enhanced with mobile NFC. This includes but is not limited to:

- Secure access to existing payment infrastructure, see 2.1.1 Banking Service Scenario
- Secure access to transportation infrastructure, see 2.1.2 Transportation Service Scenario
- Secure customer verification token, see 2.1.3 Loyalty Service Scenario
- Easy data exchange with security, see 2.1.4 Information exchange Scenario

All use cases are based on a NFC capable mobile device implementing a CLF which is offering SWP support. The UICC using SWP to communicate with the CLF plays the role of a secure element. The following technical requirements are valid for all use cases:

- The Mobile Device shall include a CLF.
- The CLF shall offer card emulation and/or card reader ability according to TS 102 622.
- The UICC shall work as a secure element hosting NFC services, providing a user interface when required by the service.
- The UICC shall support Java Card 3.0.1 classic and GlobalPlatform 2.2.
- The UICC and the Mobile Device shall at least be compliant to ETSI SCP Release 7 Specifications, despite with devices compliant to ETSI SCP Release 9 Specifications, new OTA services and a better user interface are available

2.1.1 Banking Service Scenario

In this scenario the customer can directly perform payment transactions using the UICC card and his NFC capable handset. To achieve this goal the UICC card is hosting one or more applications of one or more banking service supplier like MasterCard, Visa or others.

The features introduced in Release 9 and GlobalPlatform 2.2 offers a significant enhancement in the security of remote service management, which makes the usage of banking easier and reliable.

For the following use cases an application is needed, residing on the UICC. This application must be provided and deployed in a secure way even in third party networks. The GlobalPlatform 2.2, Amendment A defines a mechanism for confidential and secure load, install and personalisation of provided applications. According to this, application distribution can be done by the Mobile Network Operator, for example an Over-The-Air platform operator. For details please refer to chapter 5.

(a) Technical requirements

UICC card requirements:

The UICC card must be equipped with one or more banking applications.

According to the GP2.2 specification there must be a security domain installed, which is able to separate the network provider, banking supplier and user access rights.

Network Provider requirements:

The Network Provider provides the UICC.

Partner requirements:

The banking service provider must deploy the applications for the UICC and the terminals which grant the data exchange between the mobile front end and the payment background system with the user interface.

Customer requirements:

The customer needs one or more accounts at the banking service

(b) Realisation

For balancing a bill the customer taps the mobile phone to the NFC enabled EPOS which verifies his identity to the banking service provider. Instead of choosing one of the customers banking cards the user can now select the account for debit within the mobile phone or directly at the banking Interface.

User interactions on the mobile phone can be requested to complete the transaction.

2.1.2 Transportation Service Scenario

A classical transportation service is the Underground-Train-Scenario. The service-access can be realised on a pre- or post-paid basis. Here the scope of the transportation service is to give the customer a secure, easy and fast way to pay the daily fees or to authorise for passing.

A user interaction during the payment is not intended within this scenario. The transportation application can be enhanced by mobile based information services to give the user more transparency on the prior performed transactions and the possibility for online booking.

Using Release 9 features this service can be enhanced with Reader Mode capabilities.

A possible use case is to give the customer the possibility to choose special features by tapping the mobile onto smart stickers. This allows e.g. easy and secure booking of additional features, without the need of expensive equipment on the transportation provider side.

A further enhancement is using the Writer mode to book multiple tickets with one account and deploy the tickets within a group afterwards.

(a) Technical requirements

UICC requirements:

The UICC must be equipped with an application of the transportation provider which grants physical access and also access to the payment services.
The UICC is capable of using the full or low power mode.

Network Provider requirements:

The Network Provider provides the UICC.

Partner requirements:

The transportation service provider must deploy the applications for the UICC card and the terminals which grant the access to the customer database of the transportation provider.

Customer requirements:

The user has to register to the customer service system of the transportation provider.

The conditions of use according to pre- or post-paying apply.

(b) Realisation

After the customer is registered to the payment service there are two possibilities for the customer to use the mobile phone for payment.

According to the possible power states of a mobile phone there are following scenarios can occur:

Full Power mode:

The Full Power mode applies when the mobile phone is switched on. If the customer now taps the mobile phone to the payment terminal the mobile phone is fully functional and the user is instantly authenticated by the UICC to the customer service system of the transportation provider.

Low Power mode:

The low Power mode applies when the mobile phone is in energy savings mode or is switched off and the battery is able to power the CLF and UICC. The user authenticates to the transportation provider when tapping the mobile phone onto the payment terminal without switching it on. In the Low Power mode no user interaction is possible.

Battery OFF mode:

The Battery OFF mode is an optional mode and applies when the battery of the mobile phone is discharged or dismounted. In case the customer taps the mobile phone onto the payment terminal the CLF powers on the UICC card in Battery OFF mode. The UICC card is now able to authenticate the user to the customer service system of the transportation provider.

In the battery OFF mode no user interaction is possible.

Please see chapter 3.3.5 for details.

2.1.3 Loyalty Service Scenario

The scope of a loyalty service is to bind the customer to the Network Provider and the partners. The second aim is to get information about the customer's needs and interests of products and services. The customer is attracted to use this service by offering free products or services. The amount of free or discounted offers may be proportional to the volume of sales the customer achieved in the past.

(a) Technical requirements

UICC requirements:

The UICC must be equipped with an application of the loyalty partners.

Network Provider requirements:

The Network Provider provides the infrastructure for collecting and transferring the user's data within the wireless network.

Partner requirements:

The partner companies apply a function to exchange data via mobile NFC between the electronic point-of-sale (EPOS) system and the mobile phone. In addition the personalised list of purchased goods has to be sent to a database to analyse the data for the statistics.

Customer requirements:

The customer has to register to the system. The mobile phone may also be offered at registration to attract the customer and to assure that the mobile phone of the customer is capable of NFC.

(b) Realisation

In the partner shops the customer taps his mobile phone onto the NFC reader. The user is now authorised via NFC and the data for the loyalty service (e.g. shop name, date, time, amount of spend money…) are exchanged between EPOS and UICC.

The user can access his account information using e.g. SCWS or S@T.

The using of achieved loyalty units can also be realised at the EPOS or with a terminal where the user can authorise himself with mobile NFC to get access to his data.

2.1.4 Information exchange Scenario

The UICC is used as a secure storage device for account access data. These accounts are used on a laptop. In case there is an access to such an account needed the mobile phone is tapped onto the NFC reader of the laptop and the access will be granted with the data from the UICC. The short distance and the applicable encryption of the NFC connection are strengthening the security.

Account data can also be synchronised or stored as backup using secured OTA.

For short range synchronisation a mobile phone to mobile phone connection is used. Here one phone acts as NFC Tag (in Card Emulation Mode) and the other one as NFC Reader/Writer.

For the future, a peer to peer connection shall be possible. Here the two devices are alternately establishing the RF field. But currently this is out of scope of this document.

(a) Technical requirements

UICC card requirements:

The UICC must be equipped with an application handling the account data storage and exchange.

Network Provider requirements:

The Network Provider provides the UICC and the infrastructure to exchange the data.

Customer requirements:

The customer has to purchase the related applications for the UICC and the laptop.

(b) Realisation

The application on the card is managing the connection to the Laptop and is sending the account data when it is requested. So the user does not have to enter his usernames and passwords in an un-secure environment. Access to the securely stored data in the UICC can be granted using the mobile phone e.g. with PIN entry or with features presented on the laptop e.g. using biometrical data like a fingerprint or a smartcard using the PC/SC interface.

2.2 Mobile NFC overview

This overview starts with the structure in which NFC components are organised. These components may reside in different hardware used for mobile NFC. It describes the needs of the UICC and the mobile phone.

It is followed by a brief description of the hardware which is able to communicate with the mobile NFC device.

2.2.1 Mobile NFC Components

The NFC concept for the mobile environment consists of a certain number of components and their interaction.

NFC components inside a mobile device are the CLF the Terminal Host and the UICC Host. These Components are forming a hub and spoke network with the CLF in its centre. The Hosts and the CLF form the Host Controller Network with the Host Controller being a part of the CLF. The CLF will also manage all RF-communication.

The hosts and the CLF are communicating with each other via the Host Controller using the Host Controller Interface (HCI). The HCI consist of a collection of Gates the HCP messaging mechanism and HCP routing.

Figure 1: Interworking of NFC hardware shows the interfaces between the hosts and the CLF in a Mobile Device. Between the UICC Host and the CLF the SWP will be applied, the Terminal Host will use a proprietary interface to the CLF. External NFC Devices like Contactless Reader or Tags will communicate via the contactless communication handled by the CLF. The following chapters will describe these components more detailed.

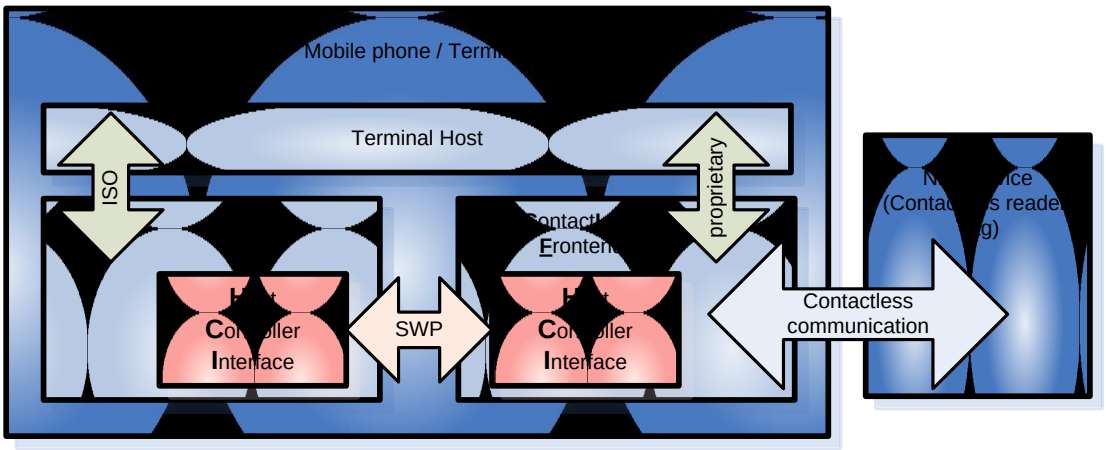


Figure 1: Interworking of NFC hardware

(a) UICC with HCI

The UICC is a multi interface device inside the mobile equipment. Equipped with the NFC features defined by ETSI and GlobalPlatform the UICC is able to communicate through the CLF of the mobile. The UICC will communicate, using the HCI. For this communication the HCI provides a number of logical Gates ([21])

» Management Gates will give access to the management of the Host Network.
» Generic Gates will provide a logical pipe to the Terminal Host.

(b) CLF with HCI

The CLF will handle all RF-communication with external NFC devices. On the physical layer the single hosts will not directly communicate with external devices or each other but will exchange messages using the HCI. The CLF will then forward these messages accordingly.

» The CLF will provide an interface between the mobile phone and a background system.
» The CLF contains the Host Controller, managing and providing communication between Gates on particular hosts.
» The CLF may also provide power for communication if the battery off mode is supported.

For communication between the CLF and the UICC Host the HCI offers the HCP which makes use of the SWP. For more details on this communication refer to chapter 2.2.2.

(c) Terminal Host

The Terminal Host is the baseband controller of the mobile equipment. Between the Terminal Host and the UICC Host a logical pipe can be established for data exchange and event triggering. This logical pipe defines a service between the hosts and a generic Gate on each side provides the entry point.

2.2.2 NFC Communication

In chapter 2.2.1 four high level communication interfaces are mentioned:

– ISO Interface
– Proprietary Interface
– Contactless Communication
– SWP

The ISO/IEC 7816 Interface [2] is the universally deployed communication interface between UICC and the baseband controller.

The proprietary interface enables the mobile phone to communicate between the Mobile Host and the CLF. The used protocol can differ between the different mobile phone vendors.

The contactless communication is the actual a wireless NFC communication.

The SWP interface is the physical line on which the UICC and the CLF in the mobile phone communicate. Data exchanged on this line is formatted in the HCP format.

The data is prepared for transmission by splitting it and adding control information for the HCP routing. After this the package will be embedded into the format for the SWP. The SWP consists of the logical link protocol, the MAC layer and finally the physical layer. For more details refer to chapter 3.3).

Figure 2 shows this communication for the different layers of abstraction. Each communication takes place between two logical Gates.

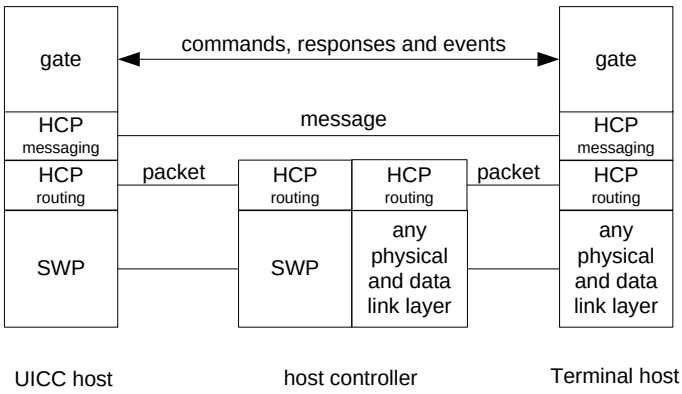


Figure 2: SWP components and their communication layers (ETSI TS 102 622)

This communication structure applies for all communications between the UICC Host and the CLF. If the opposing Gate is located on the Terminal Host, the communication on HCP and SWP layer will be routed by the CLF.

2.2.3 Mobile NFC operation modes

Within mobile NFC several operation modes are described from which side a NFC connection is initiated and how the data transfer works.

This paragraph is structured according to ISO/IEC standards where NFC devices can be categorized in three different operating modes, where one of the NFC components has either an active or passive role.

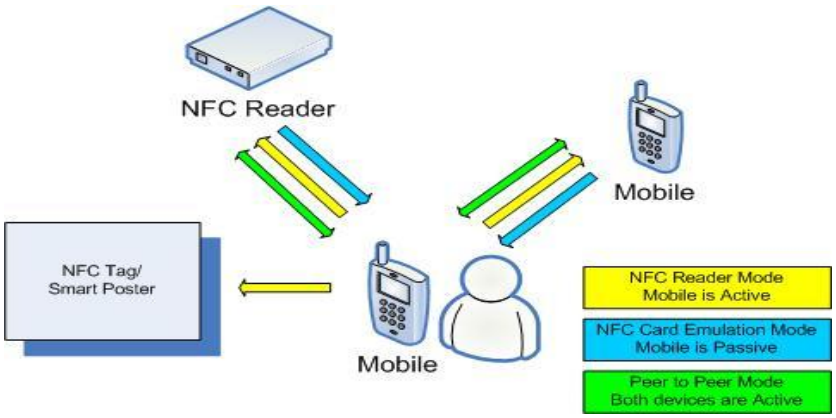


Figure 3: Operating modes for NFC devices

(a) Card Emulation Mode

The mobile NFC device is in passive mode and can accept a connection when entered into an RF field of an external NFC Reader (active).

This mode is standardized in ISO/IEC 14443 ([4], [5], [6]) and is applicable for use cases like contactless payment or transport applications.

(b) Reader/Writer Mode

The mobile NFC device is initiating its own RF field as standardized in ISO/IEC 14443 ([4], [5], [6]). This enables the mobile NFC device to act like a NFC Reader (active)

This mode is applicable for use cases like reading NFC Tags or NFC Smartposter. It is also possible to interact with another UICC in Card Emulation Mode.

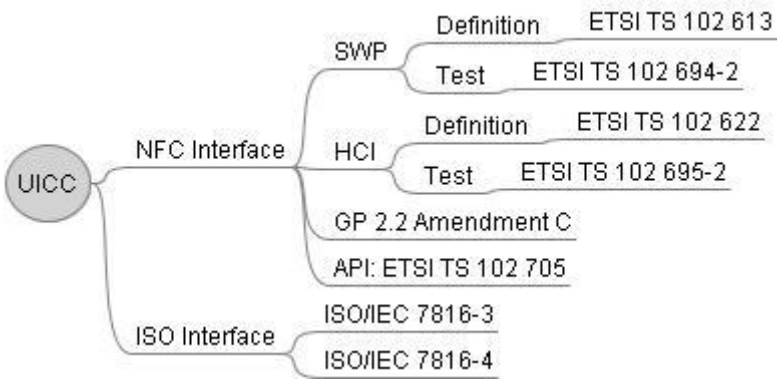
(c) Peer-to-Peer Mode (P2P)

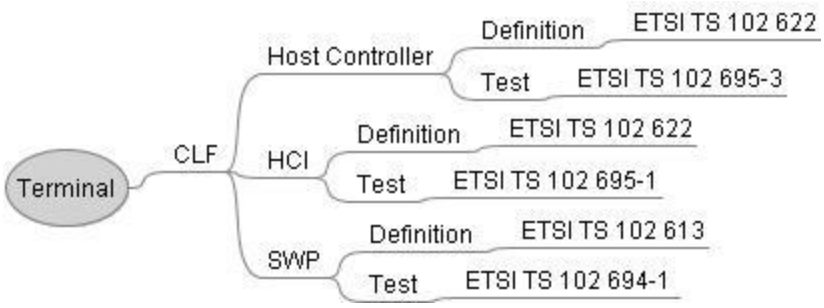
The P2P-Mode standardized in [8] ISO/IEC 18092 is defined as two active devices which are alternately establishing a RF field for communication.

The P2P Mode is not in the scope of this document.

2.3 Specifications

This chapter gives an overview on important specifications regarding NFC and the related topics.





2.3.1 New main features in R9

The Release 9 introduces new features but also implements corrections and specification updates.

The evolution from Release 7 to Release 9 includes also a change of the referencing to other specifications. Table 1: Overview of references in R7 and R9 below gives a brief overview.

Table 1: Overview of references in R7 and R9

	Release 7 references to	Release 9 references to
Java Card	2.2.2	3.0 Classic
GlobalPlatform	2.1.1	2.2 Amendment A, 1.0 Amendment B, 1.0 Amendment C, 1.0
Reader Mode		HCI API 4.2
Connectivity service		HCI API 4.2
Card content management	GlobalPlatform 2.1.1 Chapter 4 Chapter 6	GlobalPlatform 2.2 Chapter 4 Chapter 9 Amendment A Confidential card content management • Link platform provider • Application provider • Application Provider Security Domain
Application Management	GlobalPlatform 2.1.1 Chapter 6	GlobalPlatform 2.2 Chapter 9 Amendment B • RAM over HTTP
Contactless services		GlobalPlatform 2.2 Amendment C (Contactless services) • CRS Application • New Lifecycle (availability state)

In the migration from GP 2.1.1 to GP 2.2 the definition of Install data for applets has changed.

The Amendment A (Confidential card content management) introduces some new features:

- The Link platform provider is responsible for Installing Applets and giving Ownership.
- The Application provider can send requests to install to the platform provider
- Creation of Application Provider Security Domain (APSD) is now possible
- This includes 2 APIs:
- API for Applet confidential personalisation implements an interface for applets which is usually invoked by Application Security Domain
- API for Controlling authorities implements an interface usually implemented by the application provided by the Controlling Authority

GlobalPlatform Amendment C (Contactless services) ([35]) introduces a new Lifecycle (featuring the availability state), the CRS Application and some further features.

2.4 Mapping of terms from different specification authorities

The terms and phrases within specifications may vary due to different specification authorities. This can have different reasons like where the origin business of the authority is situated.

This is the case when comparing ISO/IEC 14443 and NFC Forum.

Examples of differences:

- Types of Contactless smartcards
  Contactless smartcards of type A or B as defined by ISO/IEC 14443-3/4 can be classified into the NFC-Forum category of tag 4A or 4B provided their file structure follow the one defined for these tags and as long as they also follow the ISO-DEP which is equal to the ISO/IEC 14443.
- Different names for same topic
  In some cases the wording between the 2 committees for a similar item is different like the ISO 14443 defines "anti-collision" when the NFC Forum talks about "Single device detection".
- Different capabilities definitions
  Also the technical capabilities can be specified different like that the ISO/IEC 14443 defines an 848 kbps maximum data rate whereas the NFC Forum only goes up to 424 kbps.

This list is not exhaustive.

2.4.1 Overview of NFC-Forum Tag-types

In the following chapters several different types and modes for NFC communication will be mentioned. This chapter contains a data sheet, listing these types and the according technical details for comparison.

Table 2: Overview on NFC Tag-types

Tag type	Type 1	Type 2	Type 3	Type 4A	Type 4B
Technology	NFC-A	NFC-A	NFC-F	NFC-A	NFC-B
Frequency range	13,56 MHz	13,56 MHz	13,56 MHz	13,56 MHz	13,56 MHz
Bit rates	106-212-424 kbps	106-212-424 kbps	212-424 kbps	106-212-424 kbps	106-212-424 kbps
Range	< 0.1 m	< 0.1 m	< 0.1 m	< 0.1 m	< 0.1 m
Contactless standard	ISO-DEP = ISO 14443	ISO-DEP = ISO 14443	ISO 18092	ISO-DEP = ISO 14443	ISO-DEP = ISO 14443
Poll Mode Coding	Modified Miller ASK 100% modulation	Modified Miller ASK 100% modulation	Manchester ASK 10% modulation	Modified Miller ASK 100% modulation	NRZ-L ASK 10% modulation
Poll Mode waveform	See table below				
Listen Mode coding	Manchester with OOK carrier submodulation	Manchester with OOK carrier submodulation	Manchester ASK 10% modulation	Manchester with OOK carrier submodulation	NRZ-L with BPSK modulation
Listen Mode waveform	See table below				
Tag Structure	Memory block	Memory block	Memory block	EF-Type ISO 7816-4	EF-Type ISO 7816-4
Block size	8 bytes	4 bytes	16 bytes	N.A.	N.A.
Memory sizes	Static : 120 bytes Dynamic : 256-2048 bytes	Static : 64 bytes Dynamic : 128-1024 bytes	depends on memory size	depends on memory size	depends on memory size
Data Access	- Read/write - Read-only	- Read/write - Read-only	- Read/write - Read-only	C-APDU	C-APDU

Tag type	Type 1	Type 2	Type 3	Type 4A	Type 4B
Command set	- RALL - READ - READ8 - WRITE-E - WRITE-E8 - WRITE-NE - WRITE-NE8 - RSEG	- READ - WRITE - SECTOR SELECT	CHECK - UPDATE - BLOCK LIST	- SELECT (file or application) - READ BINARY - UPDATE BINARY	- SELECT (file or application) - READ BINARY - UPDATE BINARY
Data structure (Tag-technology independent)	Messages using NDEF = NFC Data Exchange Format (TLV-based)				
NDEF Messages	Contains 1 or more NDEF records, each carrying a payload of arbitrary type				
Possible payloads in NDEF	- URI (RFC 3986) - RTD like smart posters - MIME media type construct messages (RFC 2046) - NFC specific formats like type identifiers as on the right:		*** URI (RFC 2717) *** Image / Jpeg (RFC 2046) *** Message/ http (RFC2616) *** XML document (RFC 3023)		

For different types of NFC communication there are different standards regarding the RF-communication. These differences have to be taken into account for the switch between NFC modes. This affects the RF-transmitter most which interfaces with the CLF.

Table 3: Poll Mode Waveforms

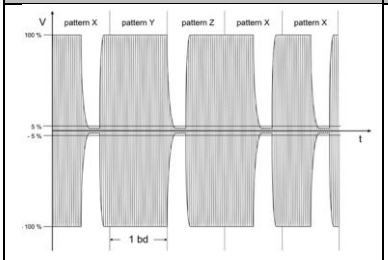
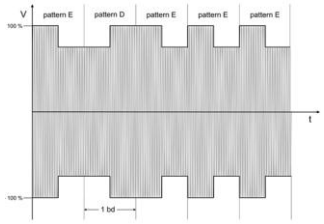
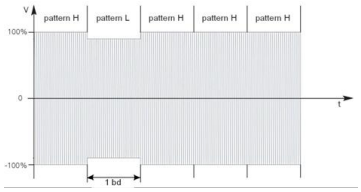
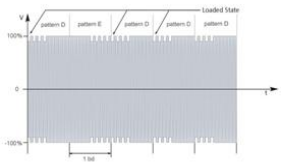
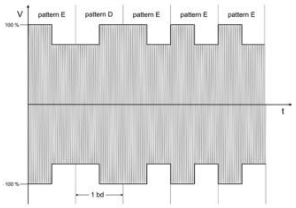
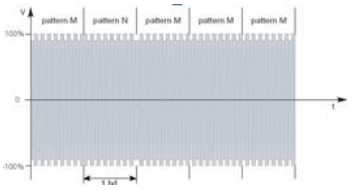
Modified Miller ASK 100% modulation	Manchester ASK 10% modulation	NRZ-L ASK 10% modulation
		

Table 4: Listen Mode Waveforms

Manchester with OOK carrier submodulation	Manchester ASK 10% modulation	NRZ-L with BPSK modulation
		

3. UICC SWP and Interaction with handset

3.1 Introduction

This section describes the Single Wire Protocol (SWP) which defines the message frames that are exchanged on the SWP-Interface between a Contactless Frontend (CLF) controller and the UICC.

Later in the section the payload of the SWP-frames is introduced, which is mainly the Host Controller Protocol (HCP) with commands and events from the Host Controller Interface (HCI).

With the HCI-Commands the CLF and UICC are able to exchange messages, configure each other and communicate with APDUs to an external NFC terminal.

3.2 Specifications

This list of specifications mainly used for SWP and HCI:

- ETSI TS 102 613 (Single Wire Protocol) (see [20])
- ETSI TS 102 622 (Host Controller Interface) (see [21])

3.3 The SWP protocol

3.3.1 Structure of a SWP Frame

The following figure splits up the different Layers of the SWP Communication Stack and their role in building together the final SWP Frame, which is sent on the Physical Line. The following figure will describe a single SWP frame with a (non fragmented) HCP message:

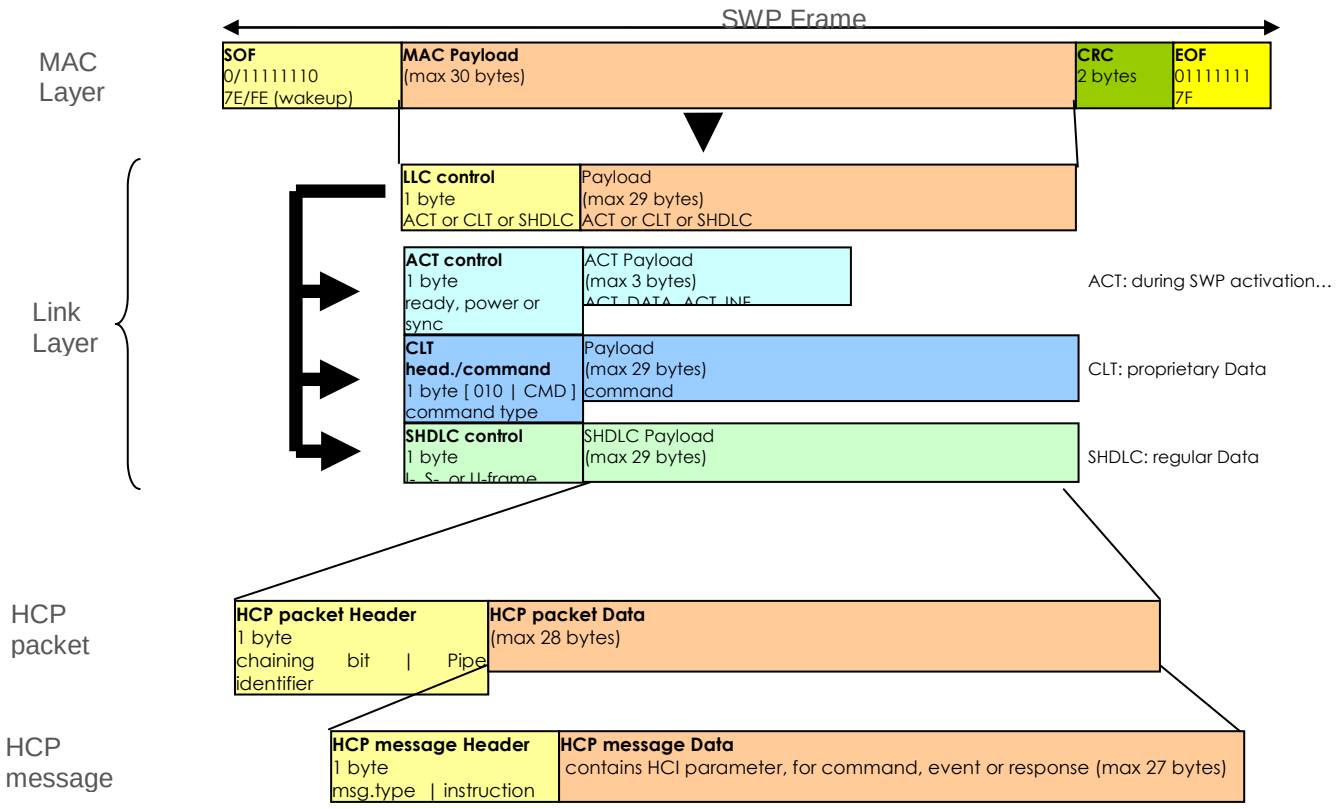


Figure 4: Structure of the communication items in the protocol layers of the SWP stack

Description of the different layers shown in the above figure:

MAC Layer:

A received SWP Frame will be evaluated by the 'Medium Access Control' Layer, which detects the SOF, EOF and checks the CRC Checksum.

Link Layer:

The Link Layer evaluates the LLC-Control byte to determine if the message type is either:

- ACT (mandatory): handles the activation Protocol during SWP activation, or
- CLT (optional): Contactless Tunnelling, used for proprietary transport mechanisms, or
- SHDLC (mandatory): Simplified High Level Data Link Control, used for regular data exchange)

HCP Packet Layer:

In a regular SHDLC Payload the Header-Byte will indicate Chaining and the Pipe-ID.

HCP message Layer:

The HCP message Header contains the message type (command, event or response) and the corresponding HCI Instruction/Event. The Parameters will follow in the Data part.

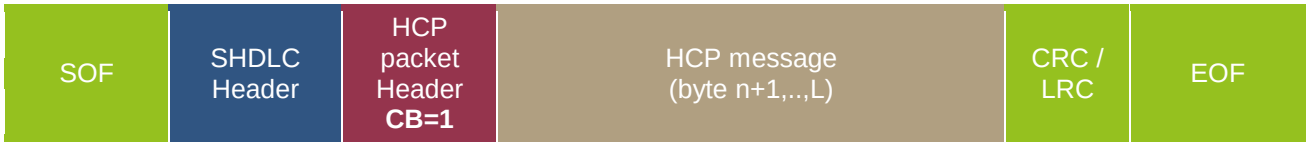
3.3.2 HCP Message Fragmentation

Due to the fact that the Payload of an HCP Message has a maximum size of 27 bytes, it is necessary that longer messages are fragmented into several frames.

Following diagram shows a message with length = L and its fragmentation using two frames. The fragmentation is indicated in the Chaining Bit (CB) of the HCP Packet Header. Only the last frame of a fragmented message will have the Chaining Bit set to value 1. Non-fragmented messages will have the Chaining Bit always set to 1. The HCP-message header only appears in the first frame of the fragmented message.

Frame 1:

Frame 2 (= last Frame):



When exchanging APDUs via SWP, each APDU is mapped to one HCI Exchange command, so a fragmented HCP message belongs to one APDU only.

3.3.3 Indication of SWP support

The UICC and the Mobile Device both indicate their capability for SWP support.

(a) UICC

» ATR

According to ETSI TS 102 221 V7.10.0 (see [9]) the UICC-CLF support is coded in the optional Global Interface bytes, which are present after the T=15 indication in the ATR. The content of the first Tbi (i>2) after T=15 is defined as:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	-	1	-	-	-	-	-	UICC-CLF interface is supported as defined in ETSI TS 102 613

» SELECT MF response

If the UICC needs to know the TERMINAL Capabilities regarding CLF support, the UICC may request the Terminal-Capabilities command from the Mobile in the UICC's Proprietary Information Template (tag 'A5'), which is responded by the selection of the MF. There the tag '87' for Supported System Commands has to indicate to the Phone:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
-	-	-	-	-	-	-	1	TERMINAL CAPABILITY is supported
-	-	-	-	-	-	-	0	TERMINAL CAPABILITY is not supported

(b) Mobile

When the UICC has indicated the request for a TERMINAL CAPABILITY commands (see above), the mobile shall send the APDU command "TERMINAL CAPABILITY" to the UICC. Within its template tag 'A9' the mobile indicates an Additional Interface Support with tag '82', coded as:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
-	-	-	-	-	-	-	1	UICC-CLF interface according to ETSI TS 102 613 supported
-	-	-	-	-	-	-	0	UICC-CLF interface according to ETSI TS 102 613 not supported

3.3.4 Optional components (CLT)

As NFC cards, Tags and Terminals are commercially used all over the world, some of them are implementing APDU based protocols according to ISO/IEC 14443-3. Many commercial applications are using a proprietary mechanism which is application/company specific, e.g. such as MIFARETM or FELICATM.

Those proprietary protocols are using "non-APDU" formatted messages between NFC-Reader and NFC-Tag, which cannot be handled by the APDU-based SHDLC layer defined in ETSI TS 102 613.

In order to route such non-APDU messages to the SIM, ETSI TS 102 613 defined the CLT mechanism (Contactless Tunnelling).

ETSI TS 102 613 specifies the CLT Mode for card emulation mode protocols according to following standards:
- ISO/IEC 14443-3 Type A (e.g. MIFARETM)
- ISO/IEC 18092 Type F (212/424 kbps passive mode) (e.g. FELICATM)

The UICC can check the CLT-support in the CLF-registry of the appropriate RF-Gate (Type A or Type F).

For ISO/IEC 14443-3 Type A protocol:

The UICC indicates the CLT Mode in the SAK (Select Acknowledge Type A) value, which is set in the CLF's Type A RF-Gate.

The SAK value is defined in ISO/IEC 14443-3 (see [5]) and indicates to the NFC-Reader, if the emulated Proximity Card (PICC) supports only ISO/IEC 14443-3 (non APDU based protocols like MIFARETM) or includes support for ISO/IEC 14443-4 (APDU based protocols).

Following SAK values can be set by the UICC to the CLF:
- 0x00 PICC not compliant with ISO/IEC 14443-4 (CLT Mode)
- 0x20 PICC compliant with ISO/IEC 14443-4 (standard Mode, APDUs are sent via SHDLC)

For more details about SAK please see page 29.

For ISO/IEC 18092 Type F protocol:

This non-APDU based protocol must be used with CLT mode, so the UICC just has to create a dynamic pipe to the CLF's Type-F Gate.

The following table gives an overview of commercially available protocols / products and gives feedback about the possibility to map them on the existing infrastructure standardised by ETSI TS 102 613 and ETSI TS 102 622.

ISO standard	known applications	SWP / HCI support
ISO/IEC 14443-4 Type-A (SAK = standard)	various, e.g. banking, loyalty cards, transport	full support by SHDLC and HCI
ISO/IEC 14443-3 Type-A (SAK = non standard)	e.g. MIFARE™	only supported with Link-Layer in CLT-Mode and proprietary higher Layer (note 1)
ISO/IEC 14443-4 Type B	various, e.g. transport, ticketing	full support by SHDLC and HCI
ISO/IEC 14443-4 Type B prime	e.g. Calypso™	full support by SHDLC and HCI
ISO/IEC 18092 Type F	e.g. Felica™ (Sony™)	only supported with Link-Layer in CLT-Mode and proprietary higher Layer
ISO/IEC 15693-3	Vicinity Cards	not supported by current HCI specification (CLF vendors may offer proprietary pipes to map nonAPDU ISO/IEC 15693-messages to HCI)

Note 1:
The MIFARETM Protocol is a proprietary protocol using hardware based stream-cipher for very fast processing. Due to the overall encryption of the full communication and the strong timing requirements, the MIFARETM messages have to be tunnelled through the CLF using the CLT-Mode on the Link-Layer.

Developer tip:
CLT support is usually bound to a specific application

Interoperability note:
UICC decides support of different gates as does Terminal, e.g. by declining pipe creation

Interoperability note:
A Release 9 UICC shall always support Type A and Type B protocol.

3.3.5 Power Modes

Power Modes from the UICC perspective:

The UICC is informed by the CLF about the actual Power-Status with the ACT-Frame (Activation Protocol) at the beginning of a SWP Activation or Re-Activation as defined in ETSI TS 102 613.

The ACT-Frame indicates following two power conditions to the UICC:

- Low Power Mode
  In the Low Power Mode the mobile is switched off, the standard ISO/IEC 7816-4 and USB interface (ETSI TS 102 600) are not activated by the mobile, just the SWP-interface is active. In that case the UICC has a maximum power-limit of 5 mA.
- Full Power Mode
  In the Full Power Mode the mobile is switched on, it activates all available interfaces, e.g. ISO/IEC 7816-4 or ETSI TS 102 600.

Battery Off Mode:

CLF manufacturers may design the CLF to work even without support of the mobile's battery (e.g. if it is nearly empty or even not installed). In that case the RF-field has to power the CLF and the UICC.

The support of "Battery Off Mode" is optional to the CLF and not standardised. The UICC cannot distinguish "battery off mode" from "low power mode".

In case of Low-Power Mode the Applet Developer should be careful with possible tears or sudden power loss which can happen if the RF-field is weak or if the user leaves the RF-field during applet operation.

The power mode can be requested with HCIDevice.getPowerMode()

Applet Developer Tip:
Even if the applet indicates that a display is required it is triggered in low power mode.

3.3.6 SWP State Management

In ETSI TS 102 613 there are three states defined for SWP:

- ACTIVATED: S1 and/or S2 are sending bits
- SUSPENDED: S1 = High, S2 = Low (initial state of SWP at interface activation)
- DEACTIVATED: S1 = Low, S2 = Low

(S1 is the communication signal from CLF to UICC via the voltage, S2 is the communication signal from the UICC to CLF via the current)

Figure 5 shows the three states and its transitions.

Figure 5: SWP states and transitions

The SUSPENDED and ACTIVATED state must be supported by all CLFs for regular functionality.

The DEACTIVATED State is also a "Power-Save Mode" and may be optionally supported by the CLF.

It depends on following conditions:
- indication from higher layer that no more activity on SWP is expected (see HCI-event EVT_OPERATION ENDED in section 3.4.1(e))
- the CLF is currently not in a RF-field and there was no SWP activity for more than 15 milliseconds

The (Re-)Activation is done by the CLF upon following conditions:
- the CLF is detecting a RF-field
- the UICC requests the interface activation by using the ACTIVATE CAT-Command as defined in ETSI TS 102 223 (see section 3.3.7)

3.3.7 ACTIVATE toolkit Command

In the previous section it is described, that the UICC cannot communicate to the CLF if the SWP interface is in the deactivated state.

If the UICC wants to communicate with the CLF it first has to send the card application (CAT) toolkit command ACTIVATE to the mobile on the ISO/IEC 7816 interface (see ETSI TS 102 223 [11]). The mobile's baseband controller will then trigger the CLF to re-activate the SWP interface for short period of time and the UICC has the chance to exchange the HCI messages with the CLF.

The ACTIVATE toolkit command has a parameter 'Activate descriptor tag' which indicates the 'UICC-CLF interface' that needs to be activated by the mobile.

3.3.8 Speed

The SWP signal includes the clock indication (self-synchronising code).

The UICC indicates its operating speed in the Extended-Capabilities Byte during the first ACT-SYNC frame (see Example in Annex A) The CLF may then apply the extended speed in the ACT-INFORMATION frame which is sent back to the UICC.

It operates at two speed options:
- Default Speed: 200 kHz – 1 MHz
  This speed is used at the start of the SWP interface.
- Extended Speed : 100 kHz – 1,69 MHz
  This speed is optional; its support has to be indicated by the UICC.

3.3.9 Sliding Window Size

The concept of a sliding window is used to send multiple frames before receiving confirmation that the first frame has been received correctly. This means that data may continue to flow in situations where there may be long "turnaround" time lags without stopping to wait for an acknowledgement.

The sliding window size is negotiated during SHDLC session establishment. The validity of the negotiated window size starts with completing a successful session establishment and ends with the interface deactivation or with a new SHDLC session re-establishment.

The sliding window size may be lower than the default value due to limited resources. In consequence, an endpoint may want to ask the other endpoint to lower the sliding window size.

The RSET frame may carry a configuration field in order to change the sliding window size (down to 2). If the default size (in case of an RSET command without configuration field) or the size provided is too large at a RSET frame reception, the receiver shall not acknowledge it. Instead, the receiver shall send a RSET frame with an appropriate sliding window size (which is lower than the window size offered by the other endpoint).

This mechanism can be seen in the Sample-Trace 1 (see Example1: Typical Full SWP/HCI Session Init), Where the CLF sends a SHDLC-Reset frame with the default windows size of four. The UICC doesn't acknowledge and sends back a SHDLC-Reset frame with its windows size of two, which is then acknowledged by the CLF.

A higher sliding window size allows a higher data throughput, e.g. if messages get lost during communication, but this requires fast hardware resources on the SHDLC layer of both SWP-hosts. Therefore on UICCs the minimum window size of two may be chosen, which is fast enough for the NFC use case and does not block the communication in case of one temporary un-acknowledged SHDLC-frame.

3.4 The HCI protocol

The HCI protocol is defined in ETSI TS 102 622 and is used to establish logical links between the physical entities which are connected via SWP. The HCI commands and events are transported in the HCP message layer of an SHDLC frame.

The usual SWP entities in a mobile are:

- Contactless Frontend (Host Controller)
- UICC (Host)
- Baseband-Controller (Host)

The physical connection between the mobile's baseband and the CLF is not necessarily SWP, it can also be another interface (e.g. I2C) and depends on the CLF/mobile design.

The following figure shows the logical HCI connections of the Host-Controller with the other Hosts.

For simplicity, Figure 6 is just an example and does not contain all available gates of the Hosts.

Figure 6: Logical connections in the HCI network

Description of the logical Components as shown in above Figure:

Host:
A Host in the HCI Infrastructure is identified by the Host-ID (1 byte). Beside several Hosts the system contains always one Host-Controller, which is the Contactless Frontend in the Mobile-NFC.

Gate:
A Gate is an entry point to a service that is operated inside a Host, identified by a Gate-ID (1 byte). Management Gates are needed for the management of the Host Network, Generic gates are just generally defined by HCI and may extend the access to the Host's services.For simplicity the figure does not include Identity Management and Loop-Back Gate (one for each Host).

Pipe:
A pipe is a logical communication channel between two gates, identified by a Pipe-ID (1 byte). Static Pipes have fixed Pipe-IDs and are always available, they do not need to be created and cannot be deleted. Dynamic Pipes have Pipe-IDs allocated by the Host-Controller, they can be created and deleted.

Interoperability Note:
The maximum number of opened pipes and the maximum number of buffered HCI commands depend on the UICC and CLF implementation (no. of pipes).

Interoperability Note:
It can not be guaranteed that commands or events can be sent on one pipe while waiting for a response on another pipe. It is highly recommended to avoid this. Additionally no command or response can be sent on a pipe before a response was received on that pipe.

3.4.1 Indication of HCI features in the Terminal Profile

The Terminal Profile command is transferred from the mobile to the UICC at start-up of a Toolkit session (power up) and gives information about various supported features, which can be checked by applet developers.

For the Host Controller Interface following two features are reflected in the Terminal Profile Data:
- Byte 25, Bit 6: mobile supports Envelope(Event-ID: HCI Connectivity)
- Byte 30, Bit 5: mobile supports toolkit command ACTIVATE

If those bits are not set or missing in the Terminal Profile command, HCI without those features can still be supported between the UICC and CLF.

(a) HCI Gates supported by CLF/UICC Hosts

Gates, which are typically used by the CLF/UICC for CardEmulation Mode:
- Link-Management gate
- Admin gate
- Connectivity gate
- CardEmulation RF-type: A, B
- Optional: B', F gate

Gates, which are supported by the Hosts but not necessary for CardEmulation or Reader Mode operation:
- Identity Management gate
- Loop back gate

Gates, which are optionally supported, depending on the configuration of the CLF/UICC:
- CardReader RF-type: A, B

Interoperability Note:
The Link-Management gates of the CLF- and UICC-Host hold an error-counter that counts invalid or lost frames on the data link layer.
There is no standard how to implement this counter. So SIMalliance members do not have the same implementation of this counter.

(b) HCI Registry

For each RF-type the CLF holds independent registries, which contain configuration settings that are specific to the corresponding RF-technology.

When the UICC triggers a SWP session initialisation, it has to set those RF-parameters for the required RF-type, which should be available for NFC communication. The RF-parameters are stored in the CLF persistently so that a new session initialisation can reuse the previously set parameters. Therefore the UICC and CLF manage the SYNC-ID and Session-ID values (please refer to section 3.5).

The following table shows the RF-types and their registry information. Some values are read-only, others can be configured by the UICC-Hosts.

Registry Information for the CardEmulation Gates:

Identifier	RF-Type ISO-A	RF-Type ISO-B	RF-Type ISO-B'	RF-Type ISO-F
1	Mode (RW)	Mode (RW)	Mode (RW)	Mode (RW)
2	UID_REG (WO)	PUPI_REG (WO)	PAT_IN (RW)	SPEED_CAP (RO)
3	SAK (RW)	AFI (RW)	DAT_OUT (RW)	CLT_SUPPORT (RO)
4	ATQA (RW)	ATQB (RW)		
5	Appl.-Data (RW)	Higher-Layer-Response (RW)		
6	FWI, SFGI (RW)	Datarate_Max (RW)		
7	CID-Supp.(RW)			
8	CLT_Supp. (RO)			
9	Datarate_Max (RW)			

Table 1: HCI registry parameters for CardEmulation gates

Registry Information for the Reader Gates:

Identifier	RF-Type ISO-A	RF-Type ISO-B
1	Datarate_Max (RW)	Higher-Layer-Response (RO)
2	UID (RO)	AFI (RW)
3	SAK (RO)	PUPI (RO)
4	ATQA (RO)	Appl.Data (RO)
5	Appl.-Data (RO)	Higher-Layer-Data (RW)
6	FWI, SFGI (RO)	

Table 2: HCI registry parameters for Reader Mode gates

(c) Usage of the Mode-Bit

The Mode-Bit is a switch in each of the CLF's RF-registries which is used to enable or disable the CardEmulation mode functionality of the CLF for the corresponding RF-technology.

It can be used by the UICC to switch the RF-technology globally on or off, e.g. to prevent APDU-exchange during a change of registry parameters.

This mechanism allows the UICC to disable e.g. the RF-type ISO-A CardEmulation mode by resetting the Mode-Bit in the ISO-A registry. In this case the CLF will not use anymore the parameters set by the UICC on the contactless interface and will not forward ISO Type-A APDU to the UICC.

A later enabling of the Mode-Bit on the same UICC will immediately re-establish the ISO-A availability because the CLF registry parameters are already set.

(d) Impact of HCI-settings to applications

Some of the CLF registry parameters are set by Install Parameters of NFC-applets (see 5.6.6, 5.6.7 , 5.6.9). During SWP Session initialisation those parameters are set by the UICC into the corresponding RF-type registry of the CLF.

The following values may have impact on the user experience of the corresponding UICC application, therefore their value need special attention:

» FWI:
The FWI (Frame Waiting time Integer) is defined in ISO/IEC 14443-3 (see [5]) and is coded in 4 high-nibble bits of the FWI/SFGI byte. It is used to calculate the maximum Frame Waiting Time (FWT) for the CLF to start its RF-response after the end of a received RF-frame, e.g. FWI=4 results in FWT= app. 4,8ms.
If the UICC needs longer than this time for an APDU response, then the CLF needs to send Waiting Time Extensions (WTX) on the RF-interface in order to keep the RF-communication with the NFC-Terminal alive. On the other side a high FWT value will force the NFC-Reader to wait a long time until it interrupts or repeats a communication (e.g. due to RF-problems).

» SFGI
The SFGI (Start-up Frame Guard time Integer) is defined in ISO/IEC14443-4 (see [6]) and is coded in the 4 low-nibble bits of the FWI/SFGI byte.
It is used as a multiplier value to calculate the Start-up Frame Guard Time (SFGT), which defines the guard time needed by the CLF before it is ready to receive the next frame after it has sent the ATS, e.g. a SFGI=14 means: SFGT is app. 4949ms. The NFC-Terminal will wait for this time until it sends out the first command.
As faster the UICC is ready to receive data, as smaller the SFGI value should be chosen.

The right FWI/SFGI values may depend on different factors like e.g. the physical coupling/interworking of NFC-Reader and CLF-antenna, or specified ranges from organisations like EMVCo. The right values will be a trade-off between fast or stable NFC-communication.

» SAK
The SAK (Select Acknowledge Type A) value, which is set in the CLF's Type A RF-Gate. The SAK value is defined in ISO/IEC 14443-3 and indicates to the NFC-Reader, if the emulated Proximity Card (PICC) supports only ISO/IEC 14443-3 (non APDU based protocols like MIFARETM) or includes support for ISO/IEC 14443-4 (APDU based protocols).

The SAK value can be set by the UICC to the CLF with the following coding:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
-	-	1	-	-	-	-	-	PICC compliant with ISO/IEC 14443-4 (standard Mode, APDUs are sent via SHDLC)

-	-	0	-	-	-	-	-	PICC not compliant with ISO/IEC 14443-4 (CLT Mode)

Developer tip:
Setting bit 6 does not mean that CLT mode is not used.
Having bit 6 not set does not mean that UICC supports CLT mode.

(e) Connectivity Gate

The Connectivity Gate is of special interest for applet developers, because it allows UICC/applet internal call-backs, interaction with applications executed in the mobile, or commands related to SWP Power-Management. From the UICC perspective the intended command and events are addressed to the connectivity gate of the terminal Host. The communication between CLF and terminal host depend on their design and must not necessarily be based on SWP and HCI.

Following commands and events are specified for the Connectivity Gate:

Command:

» PRO_HOST_REQUEST
This command allows a host (e.g. UICC) to request the terminal host to request SWP interface activation of other hosts. But it does not apply to request the activation of the host controller (CLF) or terminal host itself.

Interoperability Tip:
PRO_HOST_REQUEST is currently not supported by all the SIM Alliance members.

Events:

» EVT_CONNECTIVITY
This event triggers the terminal host to send an Envelope "HCI connectivity event" as defined in ETSI TS 102 223 (see [11]) to the UICC on the ISO/IEC 7816 [2] interface.

» EVT_TRANSACTION
This event triggers the terminal host to launch an application (J2ME Midlet) which is intended to the terminal. The Midlet must have registered itself for a certain AID, using the JSR257 (see [29]) API.
This AID and an optional additional byte array of 256 bytes length are parameters for this event.

» EVT_OPERATION ENDED
This event notifies the terminal host that the SWP interface activation from the previous PRO_HOST_REQUEST command is no longer used.

Interoperability Tip:
EVT_OPERATION_ENDED is currently not supported by all the SIMalliance member.

For the terminal host the support of the Connectivity Gate is optional. If supported only some of the events may be supported.

Usually the UICC opens the "Connectivity Pipe".

(i) Connectivity Event

The following figure shows the mechanism of the Connectivity Event:

1) An applet on the UICC uses the API of ETSI TS 102 705 by calling `ConnectivityService.prepareAndSendConnectivityEvent()`.

The UICC will send the HCI-Event to the CLF's Connectivity Gate.

2) The CLF will forward the Connectivity-Event from the UICC to the Terminal-Host (if it supports this event)

3) The Terminal will send a Toolkit-Envelope with the Event-ID "HCI Connectivity" to the UICC

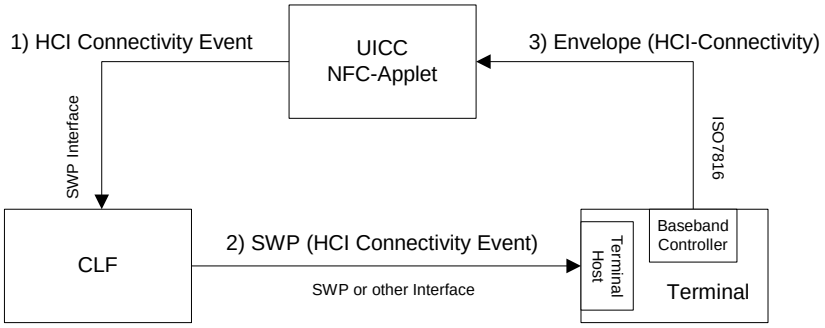


Figure 7: Communication flow between Terminal, UICC and CLF

Applet Developer Tip:
This Connectivity Event can be used by a NFC-applet during the execution in the process() method, in order to get triggered immediately via the processToolkit() method, e.g. to send a proactive command. Note that in this case all applets registered to this event get triggered in the processToolkit() method.

3.5 CLF – UICC Synchronisation

The full SWP session initialisation as seen in the sample trace 1 (see Appendix A) can consist of various RF-registry settings depending on the configuration of the UICC. For performance reasons most of the settings are stored persistent in the CLF and in the UICC in order to speed up the following start up phase between the same CLF and UICC. Therefore, a synchronisation mechanism was established where each entity – CLF and UICC – can detect a previous full session init, using the following identifiers:

- SYNC_ID (identifies the UICC)
- Session ID (identifies the CLF)

The first frame sent by the UICC to the CLF after SWP interface activation is called ACT_SYNC frame, which contains a SYNC_ID value.

The CLF will compare the received SYNC_ID with the internally stored SYNC_ID from the last SWP session. If it doesn't match, the CLF will later respond with a Session-ID different from previous one (e.g. default Session-ID 'FF….FF'), which is an indication for the UICC to issue the CLEAR_ALL_PIPES command and fully reconfigure the CLF.

This CLEAR_ALL_PIPES command contains a new generated random SYNC_ID from the UICC.

After the UICC has finished the CLF-configuration (open dynamic pipes and set registry values), the UICC generates a Session-ID (random value) which is sent to the CLF and stored in the CLF's non volatile memory.

If the SWP interface is reactivated and the CLF recognises the SYNC_ID from the last SWP-session, the CLF will respond with the last Session-ID. The UICC then compares it with its

reference value from the last SWP session and if it matches, the CLF and UICC are both ready for NFC communication.

Note that the Session-ID is not always requested by the UICC, e.g.:

- If it is in low power mode
- It is in Full Power mode but an ACTIVATE was already received before in the same session (i.e. the session is already initialized)

4. NFC Application development

4.1 Java Card APIs

Applications using ETSI Release 9 APIs and programming model can take benefits from all the services provided by the SWP/HCI protocols.

However, also applications compliant to ETSI Release 7 specifications can provide services over the SWP/HCI interface, with some limitations indicated in the following.

4.1.1 Backward compatibility (Release 7)

Card Emulation mode:
In Release 7, only Card Emulation mode is supported by Java Card APIs.
ETSI Release 7 refers to Java Card 2.2.2 and to GlobalPlatform 2.1.1.
Card Emulation Mode Applications are interoperable in this context.

See SteppingStones_R7_v1.0.0 [43]: section 7.12.2 Java Card 2.2.2 Contactless API

In Release 7, Reader Mode and Connectivity service were not defined for Java Card context. Consequently such APIs were not supported or were supported in a proprietary way.

Reader Mode:
In Release 7, NFC Applications using Reader Mode are considered as proprietary (using proprietary APIs)

Connectivity Service:
In Release 7, NFC Applications using Connectivity Service are considered as proprietary (using proprietary APIs)

See SteppingStones_R7_v1.0.0 [43]: section 7.12.4 Connectivity mechanism for more information of such mechanism in the context of the Release 7.

Applet Installation:
Install parameters (e.g. protocol parameters, etc.) are not defined in GlobalPlatform Card Specification v2.1.1 [37]; support of those parameters is proprietary.

4.1.2 JC 3.0.1 classic

Java Card(TM) Classic 3.0.1 is the evolution of previous Java Card Version. The main difference with the previous version concerns contactless features. It clarifies the use of Interface Reset Behaviour and management of transient data.

See Java Card 3.0.1 Classic [11]: section 3.6 Power loss and Reset

In the context of a contactless communication, Transient data are no more cleared if the Card is still powered (i.e. contact I/O is still available, i.e. Card operating in "Battery on").

Java Card Classic 3.0.1 APIs could still be used for NFC Applications in CardEmulation Mode (APDU processing).

Developer Tip:
The Java Card Classic VM is a single thread virtual machine and it is shared between all the interfaces. SIMalliance members agree that when one application is executing on the JCVM (e.g. executing an APDU on the ISO interface) and another message targeting another Java Card application is received on the other interface (e.g. an APDU

is received on the HCI), execution of the later may be delayed after the completion of the execution of the first.

4.1.3 Update of the GlobalPlatform Card Specifications v2.2 with Amendment C and Amendment A

Contactless and user-interface parameters are set in the Card Registry and cannot be changed by Java Card application by means of APIs; however, by using RAM protocol it is possible to manage those parameters via OTA.

(a) Update of the Application Life Cycle (Amendment C)

A new life cycle state has been introduced in the application life cycle to manage the specificities of a contactless application: the Availability state.

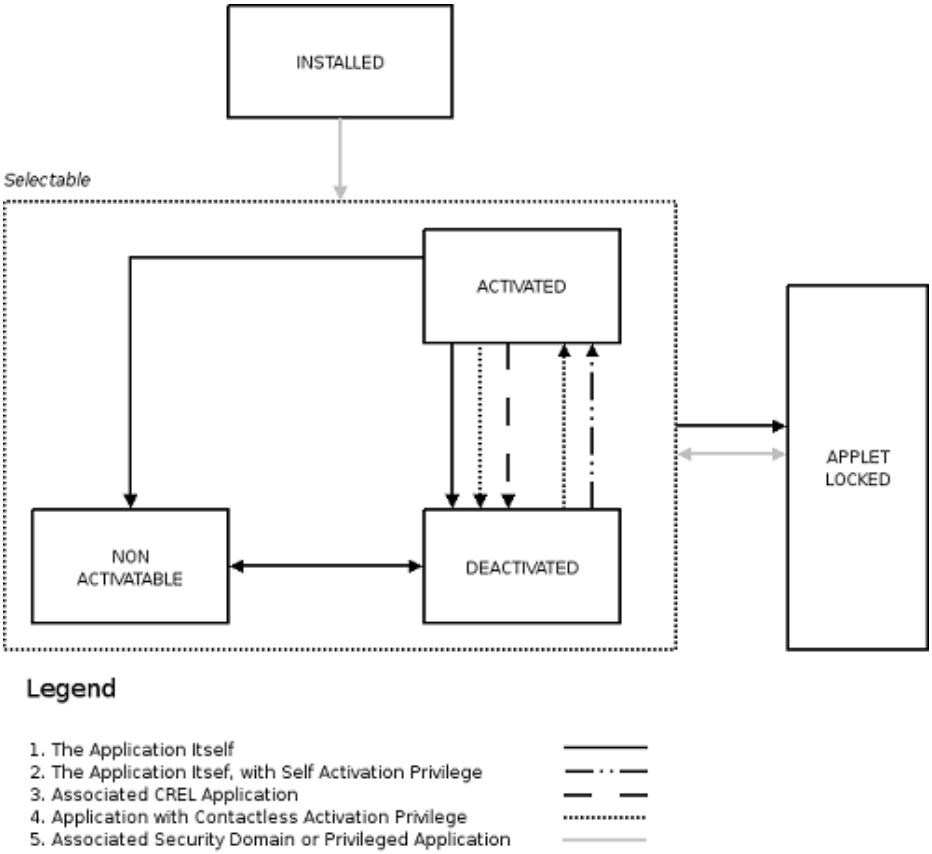


Figure 8: Contactless Application states as defined by GlobalPlatform 2.2, Amendment C [35]

When an Applet is ACTIVATED, it is available on the contactless interface. It implies that its associated protocol parameters are reflected on the HCI RF gates of the CLF.

When the Applet is installed, its state is inherited from the default state defined by its Security Domain. But, the state can be updated later by the Applet itself (with the associated Contactless Self Activation privilege) or by an Application having the Contactless Activation Privilege (i.e. the CRS application).

If an Application has the Contactless Self Activation privilege, the CRS Application will not be involved in the activation of the Application, although the CRS Application will be notified of this activation. For all others cases, the CRS Application will process Activation requests.

Example:

```
public interface GPCLRegistryEntry
            byte setCLState(byte state)
```

This API allows, for example, an Application to request to be activated on the contactless interface (state changed to ACTIVATED).

(b) The CRS Application:

The CRS Application is the Application which manages the Contactless Registry Service (only one per Secure Element). This Application shall implement the CRSApplication Interface.

This service provides means for:

- Administrates the Availability state of Applications
- Switching the ISO/IEC 14443 RF Interface
- Retrieving and prioritizing all Applications
- Resolving conflicts with ACTIVATED Applications

The following figure shows an example of the communication flow of the different entities.

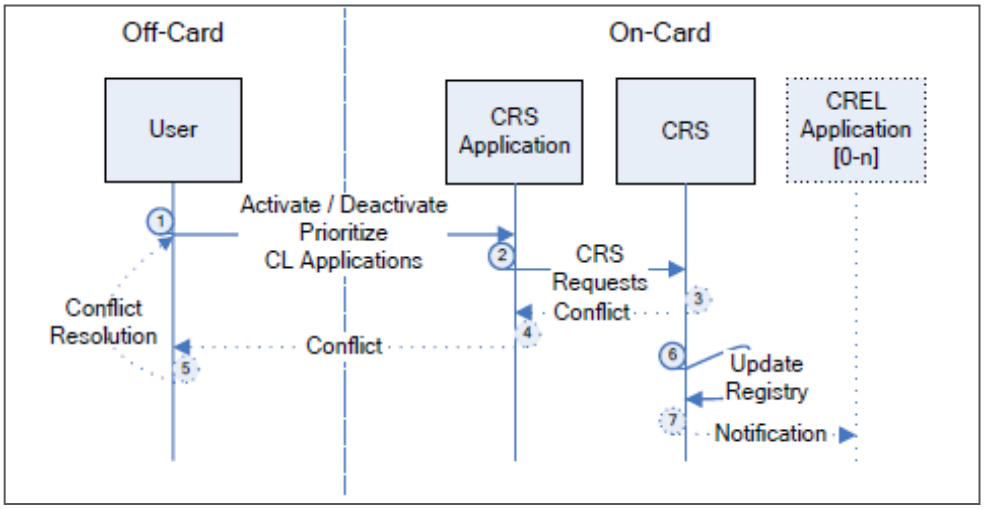


Figure 9: Example of communication flow

The use of the CRS Application could induce interactions between the CRS and Applications in a different manner depending of the interfaces they exposed, which allow us to identify:

- The Contactless Applications (interface CLApplet):

The Contactless Applications are the targets to be administrated by the CRS Application, they will be notified for the events that affects their registry entries.

- The Contactless Registry Event Listener (interface CRELApplication):

The CREL Applications are Listeners that could be registered to any Contactless Applications to be notified on any events that affects the registry entries of these Applications.

The CRS application should provide a "user-friendly" way (e.g. by Toolkit means) for the user to benefit of the services that could be provided by its Contactless Applications.

When the user wants to activate a specific Contactless Application, the CRS Application will process the request and warn the user of the result:

- If there are no conflicts with ACTIVATED Contactless Applications, the user will be able to immediately use it.
- If conflicts are detected with existent ACTIVATED Contactless Applications, the CRS Application will provide to the user a way to deactivate all conflicting Contactless Applications. Then, the newly ACTIVATED Application will be available on the Contactless interface.

Example of useful APIs to be used in the CRSApplication in order to implement specific business policies

```
org.globalplatform.contactless
Class GPCLSystem
```

`static GPCLRegistryEntry getGPCLRegistryEntry(AID oAID)`
Gets a reference to a GPCLRegistryEntry interface. Could be used to access related Objects and methods of the specified entry.

`static void setCommunicationInterface(short sInterface, boolean onOff)`
This method allows switching ON or switching OFF the ISO 14443 interface at GlobalPlatform card level.

`static void setVolatilePriority(GPCLRegistryEntry oEntry)`
Sets up or discards the volatile priority of a GPCLRegistryEntry .

```
org.globalplatform.contactless:
public interface GPCLRegistryEntry extends GPRegistryEntry
```

`byte getCLState()`
This method returns the contactless life cycle state of application (ACTIVATED, DEACTIVATED, NON_ACTIVATABLE)

`byte setCLState(byte state)`
Changes the contactless life cycle state of the applet associated with this GPCLRegistryEntry.

`GPCLRegistryEntry getNextConflictingApplication(GPCLRegistryEntry oEntry)`
This method allows iteration over the currently activated contactless applets that would conflict if the applet associated with this entry was activated.

```
Org.globalplatform.contactless
public interface CRSApplication extends CRELApplication
```

`void processCLRequest(GPRegistryEntry requester, GPCLRegistryEntry target, short event)`
Called by the OPEN for each activation requested by an Application that does not have the Self-Activation Privilege. This method is a unique entry point for all requests for LifeCycle and registries management.

```
org.globalplatform.contactless
public interface CLApplet
```

`void notifyCLEvent(short event)`
Applet receives notification on event affecting its registry entry. Each applet can take specific actions depending on the changes on their registries.

Note for Developers:
Concerning notification mechanisms:

- SIMalliance members agree that the notifiers are notified in the scope of the API calling. For example, with the method `setCLState()`: the notifiers are called before the method returns.
- Developers should be aware that applications may be notified again in case of power loss.
- Developers should be aware of the risk of creating notification loops by changing a state within call (see §3.10.1 of GlobalPlatform 2.2 Amendment C [35]).).

Interoperability issue:
In case of several notifiers, there exists no defined calling order.

(c) Confidential card content management (Amendment A)

In the contactless environment, the provider of the contactless application may want to download and manage its applications in a confidential manner with regard to the network operator that provides the OTA channel.

The GlobalPlatform 2.2 Amendment A [33] defines mechanisms for an Application Provider to confidentially load, instantiate and personalize its application using an untrusted transport link. More precisely:

- An Application Provider shall be able to load confidential applications using an untrusted transport link, able to instantiate and personalize them.
- An Link Platform Operator (e.g. OTA operator) shall be able to create on-card component to load and manage confidential application and transfer the ownership to an Application Provider

To do this, a Security Domain called Application Provider Security Domain (APSD) needs to be personalized (i.e. Key creation) in trustworthy way. There are several schemes to achieve that, some of them use an on-card Controlling Authority.

New APIs have been defined in GP 2.2 Amendment A [35] section 5:

- An API for applications that needs to be personalized over this "confidential" channel
- An API that should be implemented by the Controlling Authority Security Domain when this model is used

(d) APIs for Applet confidential personalization:

`org.globalplatform interface Personalization`
Shall be implemented by an Applet which needs to be personalized through a "confidential" channel (i.e. through another Security Domain).

`org.globalplatform interface Personalization extends shareable:`
Usually this interface is invoked by the Application Security Domain to forward the personalization data to the Applet when an INSTALL (for PERSONALIZATION) and STORE DATA sequence is.

`short processData (byte[] inBuffer, short inOffset, short inLength, byte[] outBuffer, short outOffset)`
This method allows the Applet to retrieve its personalization data from another Security Domain on the card,
This method allows also the applet to send back personalization information data (i.e. personalization receipt).

(e) APIs for the Controlling Authority:

`org.globalplatform interface Authority extends shareable`
This interface is responsible of signing and verifying messages with a key implicitely known (by the concerned parties). This interface is usually implemented by the application provided by the Controlling Authority.

`Void init(byte theMode)`
Initialize the Signature object with the appropriate Key.

`Short sign(byte[] inBuff, short inOffset, short inLength, byte[] sigBuff, short sigOffset)`
Generate the signature of all/last input data.

`Void update(byte[] inBuff, short inOffset, short inLength)`
Accumulate a signature of the input data.

`Short verifyKey(byte[] inBuff, short inOffset, short inLength, byte[] outBuff, short outOffset)`
Verify and extract a key.

4.2 Use of ETSI TS 102 705

4.2.1 Application model

This APIs is based on event/listener model, it is closer to HCI communication level (than `process(APDU)` use).

This specification provides APIs for the following modes/services.

(a) Card Emulation mode

When using the Card Emulation mode, data could be exchanged either through the `process(APDU)` method or the OnCallBack() method. These two ways of exchange are mutually exclusive within the same transaction. The advantage of `OnCallBack()` method is the support of APDU based but also non-APDU based Applet (see section (d)). The advantage of `process(APDU)` method is obviously the legacy support of existing Applets.

To receive data through the OnCallBack() method, the API event EVENT_ON_SEND_DATA shall be activated (`activateEvent()` method). Otherwise, APDUs will be received through the process(APDU) method.

The application can also subscribe to the event "EVT_FIELD_OFF".

> **Note for developers:**
> Only a selected applet will receive the EVT_FIELD_OFF. Considering this, the event will mainly be forwarded in case of abnormal end of transaction when the RF reader has not deselected yet the applet from RF perspective (e.g field tearing, without RF "DESELECT" frame)

Each API event activation is independent of the others subscribed events.

This is shown in the example below with APDU management through process() method and the EVENT_FIELD_OFF activated.

Example for a Card Emulation Applet with the EVT_FIELD_OFF subscription:

```
package CardEmulation_TestApplet;
// Import Java Card and ToolKit
// Import HCI APIs
import uicc.hci.framework.HCIDevice;
import uicc.hci.framework.HCIException;
import uicc.hci.framework.HCIMessage;
import uicc.hci.framework.HCIService;
import uicc.hci.services.cardemulation.CardEmulationListener;
import uicc.hci.services.cardemulation.CardEmulationMessage;

public class CardEmulation_TestApplet extends Applet implements
        CardEmulationListener, MultiSelectable {

    public HCIService myCardEmulationService;

    public CardEmulation_TestApplet() {

        // Allocation of Memory
        // Registration of the Applet

        // Retrieves and register service and activate EVT_FIELD_OFF
        myCardEmulationService =
HCIDevice.getHCIService(HCIDevice.CARD_EMULATION_SERVICE_ID);
        myCardEmulationService.register(this);

myCardEmulationService.activateEvent(CardEmulationListener.EVENT_FIELD_OFF);
    }

    public void process(APDU apdu) throws ISOException {

        //Manage APDUs

    }

    public void onCallback(byte event, HCIMessage message) {
        switch (event) {
            // RF Transaction failed
            case CardEmulationListener.EVENT_FIELD_OFF:
            //
            // Take specific Action
            //
            break;

            default:
                break;
        }

    }

    public void deselect(boolean appInstStillActive) {
        // If RF ? =>  Transaction successful
        //
        // Take specific Action
        //
    }
}
```

(b) Reader mode: standardized way of reading targets

The reader mode can only be managed in an interoperable way using the HCI API defined in ETSI TS 102 705 [27]. In this case, the Applet shall implement the ReaderListener interface.

The ReaderMode service is retrieved through the method `HCIDevice.getHCIService(short service)`.

The Applet shall first start the discovery procedure and look for any NFC target in the field (i.e a NFC TAG or a NFC Smartposter, supported by ETSI TS 102 622 [21]). To start the discovery, the Applet shall be ACTIVATED and activate the API event `ReaderListener.EVENT_TARGET_DISCOVERED` using the method `HCIService. activateEvent(byte event)`.

If a target is discovered, the Applet will be notified through its `HCIListener.onCallback(byte event, HCIMessage message)` method.

Then, to start reading the target, the Applet shall call the method `ReaderMessage.prepareAndSendWriteXchgDataCommand()`.

When several targets are in the field, the event EVENT_TARGET_DISCOVERED with MULTIPLE_TARGET_STATUS is generated and no data could be exchanged.

If this status is received by the Applet, the user must be warned so he could move away the conflicting NFC Tags and the Applet can start again the discovery procedure using the method `restartReaderModeProcedure()`.

In any communication error case at RF level, the Applet can invoke the method `restartReaderModeProcedure()` to restart the discovery procedure.

The Applet can use the method `prepareAndSendGetParameterCommand(byte paramid)` for retrieving information in the CLF Gates Registries.

Developer tip:
Reader Mode Applets are not selected by the JCRE when performing NFC Tags reading. A Java Card context switch occurs during the invocation. This implies that clear-on-deselect arrays owned by the applet are not cleared and cannot be accessed during Reader Mode operations.

Example for a Reader Mode Applet based on ToolKit interaction:

```
package ReaderMode_ToolKit_TestApplet;
// Import Java Card and ToolKit
// Import HCI APIs
import uicc.hci.framework.HCIDevice;
import uicc.hci.framework.HCIException;
import uicc.hci.framework.HCIMessage;
import uicc.hci.framework.HCIService;
import uicc.hci.services.connectivity.ConnectivityService;
import uicc.hci.services.readermode.ReaderListener;
import uicc.hci.services.readermode.ReaderMessage;
import uicc.hci.services.readermode.ReaderService;

public class ReaderMode_ToolKit_TestApplet extends Applet implements
    ReaderListener,MultiSelectable, ToolkitConstants, ToolkitInterface {

    private ToolkitRegistry myToolKitReg;
    private HCIService myReaderService;
    private ConnectivityService myConnectivityService;

    public ReaderMode_ToolKit_TestApplet() {

        // Allocation of Memory
        // Registration of the Applet
        // ToolKit registration to launch the Applet

        //Get the ReaderService
        myReaderService =
(ReaderService)HCIDevice.getHCIService(HCIDevice.READER_SERVICE_ID);
        myReaderService.register(this);
    }
```

```
    public void processToolkit(short event) throws ToolkitException {
        switch(event){
            case EVENT_MENU_SELECTION:
            // Possible sending of ACTIVATE Command
            // Start the polling

myReaderService.activateEvent(ReaderListener.EVENT_TARGET_DISCOVERED);
            break;

            case EVENT_EVENT_DOWNLOAD_HCI_CONNECTIVITY:
            //Clear HCI Connectivity Event
            myToolKitReg.clearEvent(EVENT_EVENT_DOWNLOAD_HCI_CONNECTIVITY);
            //
            //Process Received Data
            //
            break;

            default: return;
        }
    }

    public void onCallback(byte event, HCIMessage myHCIMessage) {

        // First: Reception of Datas
        short offset = myHCIMessage.getReceiveOffset();
        short length = myHCIMessage.getReceiveLength();
        byte[] IncomingBuffer = myHCIMessage.getReceiveBuffer();

        switch(event){

            //Target Discovered !!!
            case ReaderListener.EVENT_TARGET_DISCOVERED:

                //Activation of relevant events

myReaderService.activateEvent(ReaderListener.EVENT_WRITE_EXCHANGE_DATA_RESPONS
E);

myReaderService.activateEvent(ReaderListener.EVENT_GET_PARAMETER_RESPONSE);

                // Check if status is OK
                if (IncomingBuffer[(short)(offset) ] ==
(ReaderMessage).SINGLE_TARGET_STATUS){

                    // Copy your Datas into IncomingBuffer
                    // and send the first message

((ReaderMessage)myHCIMessage).prepareAndSendWriteXchgDataCommand((byte) - 1,
IncomingBuffer, (short)0, (short)OutDataLength);

                    // Or retrieves informations on the Target discovered
                    ((ReaderService)myReaderService).getReaderRFType();

((ReaderMessage)myHCIMessage).prepareAndSendGetParameterCommand(ReaderMessage.
PARAM_ID_TYPE_B_READER_AFI);

                }else{
                // If status is equal to MULTIPLE_TARGET_STATUS

((ReaderMessage)myHCIMessage).restartReaderModeProcedure();
                // Warn the user
                }
                break;

            //Response received !!
            case ReaderListener.EVENT_WRITE_EXCHANGE_DATA_RESPONSE:

                //First check if response is a Success
                if(myHCIMessage.getInstruction() ==
(byte)(myHCIMessage.RESP_ANY_OK)){

                    //Copy your Datas into IncomingBuffer
```

```
                    // and send your next message

((ReaderMessage)myHCIMessage).prepareAndSendWriteXchgDataCommand((byte) - 1,
IncomingBuffer, (short)0, (short)OutDataLength);

                    // Or you have all the datas and you want to process them

myToolKitReg.setEvent(ToolkitConstants.EVENT_EVENT_DOWNLOAD_HCI_CONNECTIVITY);
                    myConnectivityService =
(ConnectivityService)HCIDevice.getHCIService(HCIDevice.CONNECTIVITY_SERVICE_ID
);
                    myConnectivityService.prepareAndSendConnectivityEvent();

myReaderService.deactivateEvent(ReaderListener.EVENT_TARGET_DISCOVERED);

                }else{
                    //Or restart the procedure

((ReaderMessage)myHCIMessage).restartReaderModeProcedure();
                }
                break;

            case ReaderListener.EVENT_GET_PARAMETER_RESPONSE:
            default:
                break;
        }
    }
}
```

(c) Connectivity service:

Interaction with Terminal with the event HCI_CONNECTIVITY (Toolkit event).
This mechanism allows the UICC to be triggered immediately by the Handset.
Associated API:
`ConnectivityService.prepareAndSendConnectivityEvent()`

Developer tip:
The main goal of this HCI event is to be triggered as soon as possible when the proactive handler is available. The mechanism is equivalent as registering to the PROACTIVE_HANDLER_AVAILABLE event, except that calling the API `prepareAndSendConnectivityEvent()` allows the Applet to be triggered immediately (There is a delay for being triggered on proactive handler availability)
Furthermore, registering only the PROACTIVE_HANDLER_AVAILABLE could be sufficient for such purpose as the HCI_CONNECTIVITY event is triggered following it.

Interaction with a MIDlet with the event HCI_TRANSACTION (see section 4.3.1(b))
This mechanism allows the UICC to trigger an associated MIDlet.
Associated APIs:

```
ConnectivityService. prepareAndSendTransactionEvent(byte[]
parameters, short parametersOffset, short parametersLen)
```

```
ConnectivityService.prepareAndSendTransactionEvent(byte[] aid,
short aidOffset, short aidLen, byte[] parameters, short
parametersOffset, short parametersLen)
```

(d) Generic APIs features

The underlying link layer of the HCI protocol is transparent and error free (i.e. SHDLC) from its perspective.

In case of link layer problem, a reset could be used to recover from this state, then both parts (UICC and CLF) are responsible to sent again partially sent messages or discard partially received messages.

However, the HCI might need a bigger buffer bandwidth than the SHDLC to transmit or receive very long messages. For this reason or any other unexpected reasons, and depending on implementation, the HCI layer might not be able to insure the success of such a recovery.

In order to be warned in case of data transmission errors over the SWP/HCI interface, the Applet can subscribe to the following API Events for any specified services:
`HCIListener.EVT_HCI_RECEPTION_FAILED`
`HCIListener.EVT_HCI_TRANSMISSION_FAILED`

The method `requestCallBackNotification()` is used to be triggered within the `OnCallBack()` method in order to initiate an HCI message.

Note for Developer:
To invoke successfully the `requestCallBackNotification()` method, the Applet does not need to be selected nor ACTIVATED

Note for Developer:
SIMalliance Members agree that the RF-Error-Indicator byte (last byte of each HCI-message triggered upon the API events CardEmulationListener. EVENT_ON_SEND_DATA and ReaderListener. EVENT_WRITE_EXCHANGE _DATA_RESPONSE, which is sent from CLF to UICC) is counted by the method `getReceiveLength()` according to ETSI TS 102 622 [21]. Even if the byte has no practical relevance in CardEmulation Service as messages with this error byte set are ignored by the HCI, this byte has to be checked by the Applet in ReaderMode Service in order to be able to warn the user or react accordingly (e.g send again the command, restart the procedure, etc…) .

In case of use of `process()` method, the APDU length do not include the last RF error indicator byte.

(e) Interface with JCRE

Rules for Select/deselect/Life Cycle state etc… are bound to GlobalPlatform 2.2 for Card Emulation (and Reader Mode when applicable)

Non-APDU based Applications support is allowed by ETSI TS 102 705 [27] using the `OnCallBack()` method.

As Java Card 3.0.1 and GlobalPlatform 2.2 do not reference it explicitly, some Card Manufacturers might not fully support it.

Interoperability issue:
In order to support legacy "non-APDU" applications, some Card Manufacturers might implement proprietary mechanisms to support these applications (e.g. by modifying the frames into APDUs to be called in `process()` method) . In such configuration, conflicts could exist between the card implementation and a NFC Application using the `OnCallBack()` method.

4.3 Interaction with the end user and data presentation

A contactless application is generally split in an application managing the business logic (i.e. the interaction on the contactless interface) and an application managing the interaction with the end user (i.e. the User Interface or UI).

The User Interface can be managed in two different ways:

- management through an application in the device (e.g. a MIDlet)
- or management through an application in the UICC i.e. using the SCWS (e.g. static pages and servlets)

4.3.1 User Interaction using a device application

The user interface can be managed using an application in the device, for example a MIDlet on Java devices.

On Java devices there are two JSRs that are useful in the context of contactless applications:

- JSR 177 Security and Trust Services API (SATSA) [28]
- JSR 257 Contactless Communication API [29]

(a) JSR 177

This specification [27] allows MIDlets to support smart card communication, generation of digital signatures, and low level cryptography operations.

It could be useful for use-cases such as protection for content management, secured smart card communication for Mobile Payment or Personal information management, etc…

The API in the JSR 177 specification [28] is defined in four optional packages that can be implemented independently:

- SATSA-APDU optional package defines an API to support communication with smart card applications using the APDU protocol.
- SATSA-JCRMI optional package defines a Java Card RMI client API that allows a J2ME application to invoke a method of a remote Java Card object as defined in Java Card RMI (Java Card 2.2 Platform Specification)
- SATSA-PKI optional package defines an API to support application level digital signature signing (but not verification) and basic user credential management. This package provides interaction with the WIM application.
- SATSA-CRYPTO optional package defines a subset of the J2SE cryptography API. It provides basic cryptographic operations to support message digest, signature verification, encryption, and decryption.

For the contactless use case, we are mainly interested in the ability to use the JSR 177 [28] for a MIDlet to send APDU to the UICC and thus to an applet. This API is defined in the package `javax.microedition.apdu`, which includes the interface `APDUConnection` to support APDU exchanges.

Note for developers:
After a Contactless transaction, data might need to be exchanged in a secure way between the Contactless Application and the user by the mean of a MIDlet (e.g. PIN)

(b) JSR 257

This specification [29] and the optional package javax.microedition.contactless provide Contactless communication APIs for MIDlet. It allows interaction between a MIDlet and contactless targets such as RFID tags and bar codes.

But, the JSR 257 [29] is also useful when the user interface of a contactless application is implemented using a MIDlet. Indeed it provides a mechanism for a card application to wake up a MIDlet for example at the end of a contactless transaction.

The MIDlet shall implement the interface `TransactionListener` as defined in JSR 257 [29].

When the applet invokes the method `ConnectivityService.prepareAndSendTransactionEvent()` as defined in section 3.3.1.3, the HCI Event EVT_TRANSACTION will be sent to the device (ref [20]). Then, the device will wake up the MIDLET through the method: `TransactionListener.externalReaderDetected()`

Interoperability issue:
JSR 257 and the management of the HCI EVT_TRANSACTION have to be implemented by the CLF and the device.

Note for developers:
Data exchanged through this HCI event are not secured, thus it shall not be used in a sensible context. However, it could be used to just wake up a MIDlet after a sensible transaction has been performed by the UICC on the contactless interface.

(c) Launch application

This mechanism allows an applet to wake up an application in the device (e.g. a MIDlet). The device first informs the UICC of the available applications in the device using the ENVELOPE (TERMINAL APPLICATIONS) defined in ETSI TS 102 223 [11]. Then; an application in the UICC can trigger this application using an OPEN CHANNEL (related to Terminal Server Mode).

This Launch application procedure and the related commands are detailed in Stepping Stones Release 7 [40].

Interoperability issue:
This procedure is only available if the device supports class 'e' and class 'k'.

4.3.2 User Interaction using the SCWS

A Smart Card Web Server (SCWS) is an HTTP server that is implemented in a smart card, residing in the mobile device. It allows the use of HTTP protocol in order to provide a Rich User Interface to card services.

For a detailed explanation of the potentiality of the SCWS and the deployment of SCWS services, see SteppingStones - Smart Card Web Server [44].

Developer tip:
The toolkit command LAUNCH BROWSER can be used by an applet to launch the browser on a URL handled by the SCWS in order to wake up the User Interface implemented through the SCWS.
The LAUNCH BROWSER command is only available if the device supports class 'c'.

Interoperability issue:
The UICC needs to be SCWS enabled. The device needs to be SCWS enabled (class 'e' with BIP Server Mode implemented, etc), see SteppingStones - Smart Card Web Server [44] for details.

4.3.3 Interaction with Handset/CLF

(a) Control of the SWP protocol by the applet

The power supplying of the CLF is managed by the device itself. For user privacy protection or power saving purpose, the CLF could be deactivated. So, the availability of the SWP line may not guaranteed. However an applet can request the enabling of the SWP line using the proactive command ACTIVATE.

Developer tip:
This mechanism is mainly useful in case of Reader Mode transactions as in Card Emulation mode transactions the CLF is woken up by the RF field. The activation of the SWP line will be needed also to update the CLF HCI registries in order to have a consistent system prior any RF transaction.

Interoperability issue:
Application developers shall not rely on the UICC framework to send the ACTIVATE command each time it would be needed.

Developer tip:
As SWP line enabling is needed to propagate HCI registries update and as these updates will follow any user action modifying the CRS, the CRS Application implementing ToolKit interface is a good candidate to send the ACTIVATE command if UICC does not send it automatically

(b) Control of the RF interface by the applet

An applet can enable/disable the RF capabilities of the CLF using the method `org.globalplatform.contactless.GPCLSystem.SetCommunicationInterface()` defined in GP 2.2 Amendment C [35]

This API allows an applet to enable/disable the RF capabilities of the CLF, provided the applet has the CONTACTLESS_ACTIVATION privilege (i.e. the CRS Application), see section 7 of GP 2.2 Amendment C [35].

Note for developers:
The device may also ask the UICC to enable/disable the RF interface using the command ENVELOPE(EVENT DOWNLOAD – Contactless state request) defined in ETSI TS 102 223 [11].
Upon reception of this ENVELOPE, depending on UICC Os and configuration, either the UICC is responsible to enable/disable the RF interface, or the CRS Application (due to CONTACTLESS_ACTIVATION privilege for calling accordingly the method `SetCommunicationInterface()`).

When the `GPCLSystem.SetCommunicationInterface()` method is invoked by an applet or when the ENVELOPE(EVENT DOWNLOAD – Contactless state request) is received from the device, the UICC behaviour will be the following at HCI level:

» Send a HCI command SET_PARAMETER to change the value of the MODE parameter for each opened pipe in Card Emulation mode.

» If any EVT_READER_REQUESTED was sent before, send an event EVT_END_OPERATION on one of the opened pipe in Reader Mode. This step is only performed when a disabling of the RF interface is requested.

Note for developers:
All SIMalliance members agree that as soon as the RF interface is disabled, any data received on this interface will not be forwarded to the JCRE and thus to any selected Applet. In that case, any selected Applet on the Contactless interface will be explicitly deselected, independently of the reception of HCI events EVT_FIELD_OFF/EVT_CARD_DEACTIVATED.

(c) Control of the Reader Mode (RF field generation)

In Reader mode, the UICC asks for RF field generation to the CLF (requesting the discovery of a target) with the event EVT_READER_REQUESTED and notifies its end with the event EVT_END_OPERATION. However, as the Terminal is "power supplying" this CLF action, it might implement some restrictions in time/duration. There are no mechanisms to inform the UICC in this case. Consequently, the applet will not be aware if the Terminal has turned off the RF field if no target is discovered.

Developer tip:
Applet should implement internal timer mechanisms to re-launch the target discovering procedure if necessary.

5. Remote management

OTA management in UICC services has always been a key element in the development of the UICC ecosystem. Since the very beginning secure remote applet management and file management as well as remote interaction with applications stored on the card has been part of the core set of UICC specifications.

With the Contactless environment the need of remote file and application management grows: a typical scenario in contactless is the addition of a new payment services after card issuance and this is typically performed via OTA. This scenario requires the possibility of management of contactless features as well as the capability of the UICC to use fast bearers. Already starting since Release 6, ETSI and GlobalPlatform specifications have evolved to meet those new scenarios.

5.1 Evolution of OTA protocols

With the Release 6, it was introduced the BIP mechanism and CAT-TP protocol as defined in ETSI TS 102 223 [11] and in ETSI TS 102 127 (see [9]), allowing fast bearers to be used to perform RFM and RAM management; in Release 7 support for applications using the CAT-TP has been introduced.

With the ETSI TS 102 226 Release 6 [14], the main changes introduced was the Remote Commands in Expanded Data Format supported by Remote Management Applications, adding more flexibility to the script execution engines. In Release 7 is these commands go beyond APDUs by also including proactive commands. Another feature introduced by Release 7 is Data Download via USSD, specified in 3GPP 31.115 [38] that specifies the use of USSD application mode, enabling the transparent transport of data between an application residing in the network and a UICC based application.

The Release 7 hence allows the adoption of fast applet downloading and files management via OTA, by using the BIP protocol and the CAT-TP. The Release 7 OTA protocols have been described in Stepping Stones Release 7 (ref. [43], §15, §18).

In Release 9 OTA protocols have continued their evolution to take into account new requirements from the NFC ecosystem.

The ETSI TS 102 226 Release 9 [14] introduces the RFM and RAM over HTTPs protocol, referenced in to the GlobalPlatform 2.2 Amd B [34]: the RAM over HTTPs is the mechanism for an Application Provider to perform Remote Application Management (RAM) according to TS 102 226 of its application i.e. to load, install and personalize using the HTTP protocol (see [45] RFC 2616) and PSK-TLS security Over-The-Air. Remote File Management via HTTPs is, also, specified by above mentioned ETSI TS by adding, in Annex B, a RFM scenario to the RAM scenario described in [34]. Another important update is the reference to the Amendment C of the GlobalPlatform 2.2 [35]: this specification introduces the managements of Parameters for Contactless Applications. It defines mechanisms, parameters, and interfaces to set-up and maintains the configuration of applications and controls their access to system resources like communication interfaces and memory and focuses on parameters and mechanisms required for Applications in card emulation mode.

5.2 Technical and test specifications

General Overview of standard specifications:

5.2.1 ETSI and 3GPP and GlobalPlatform: General overview of the specifications

The ETSI Technical Specifications ETSI TS 102 225 [13] and ETSI TS 102 226 [14] specify Over The Air management protocol of the card. In particular, the ETSI TS 102 225 is applicable to the exchange of secured packets between an entity in a network and an entity in the UICC. On the other hand the ETSI TS 102 226 defines the remote management of the UICC based on any of the secured packet structures specified in ETSI TS 102 225.

UICC
ETSI TS 102 225 : Smart cards, Secured packet structure for UICC based applications
ETSI TS 102 226 : Smart cards, Remote APDU structure for UICC based applications
ETSI TS 102 223 : Smart cards; Card Application Toolkit (CAT)
ETSI TS 102 127 : Smart cards; Transport protocol for CAT applications; Stage 2
ETSI TS 102 483 : Smart cards; UICC-Terminal interface; Internet Protocol connectivity between UICC and terminal
GP2.2 : Global Open Platform Card Specification, Version 2.2 (plus Amendment A – B – C)

(U)SIM
3GPP TS 24.090 : Unstructured Supplementary Service Data (USSD)
3GPP TS 23.040 : Technical realization of the Short Message Service (SMS)
3GPP TS 23.041 : Technical realization of Cell Broadcast Service (CBS)
3GPP TS 31.115 : Secured packet structure for (U)SIM Toolkit applications (SMS-PP, SMS-CB, USSD)
3GPP TS 31.116 : Remote APDU Structure for (U)SIM Toolkit applications (RFM and RAM)

5.3 Transport layers

With the adoption of the Release 9, there are now several standardized ways to download data to the UICC:

- Short Message Service Point-to-Point (SMS-PP) – single and concatenated
- Short Message Service Cell Broadcast (SMS-CB) – single and concatenated
- Unstructured Supplementary Service Data (USSD)
- Card Application Toolkit Transport Protocol (CAT_TP)
- HTTP protocol (RFC 2616 [2]) and PSK TLS security Over-The-Air.
- ETSI TS 102 226 over TCP/IP

These mechanisms can be used to trigger a Toolkit Application or to manage the card remotely (RAM and RFM).

Interoperability issue:
Some services may offer similar user experience and value for the operator; it is possible that on some SIMalliance cards those features will not be present in order to save resources, e.g. on card supporting RAM over HTTPs, CAT-TP could also be disabled.

5.3.1 Short Message Service Transport Layer

(a) Short Message Point-To-Point (SMS-PP)

The protocols and protocol layering for Short Messages Point-To-Point (SMS-PP) are defined in 3GPP TS 23.040 [40]. If the incoming message is secured (e.g. the TP-UD contains a data structure according to 3GPP TS 31.115 [38] and ETSI TS 102 225 [13]) then we are speaking of a formatted Short Message. Otherwise the term is unformatted. See SteppingStones R7 (ref. [43], § 15.1.1) for more details.

(b) Short Message Service Cell Broadcast (SMS-CB)

A Short Message Cell Broadcast is sent from the base station to all mobiles which are located in the same cell. Normally this service is used for localized information but it can be used to download dato to SIM by suing signalling channel. See SteppingStones R7 (ref. [43], § 15.2) for more details.

5.3.2 Unstructured Supplementary Service Data (USSD)

The specification 3GPP TS 31.115 [38] introduces the USSD as a new secured packet. USSD is generally associated with real-time or instant messaging type phone services (see and 3GPP TS 24.090 [41]). There is no store-and-forward capability, such as is typical of other short-message protocols (in other words, an SMSC is not present in the processing path). Response times for interactive USSD-based services are generally quicker than those used for SMS. See SteppingStones R7 (ref. [43], § 15.2) for more details.

5.3.3 Card Application Toolkit Transport Layer (CAT_TP)

The CAT_TP protocol is defined in the ETSI TS 102 127 (see [9]). This protocol uses the Bearer Independent Protocol mechanism to exchange data between the ME and the card. The BIP mechanism is described in specified in ETSI TS 102 223 [11]. See SteppingStones R7 (ref. [43], § 14) for more details about BIP and CAT-TP.

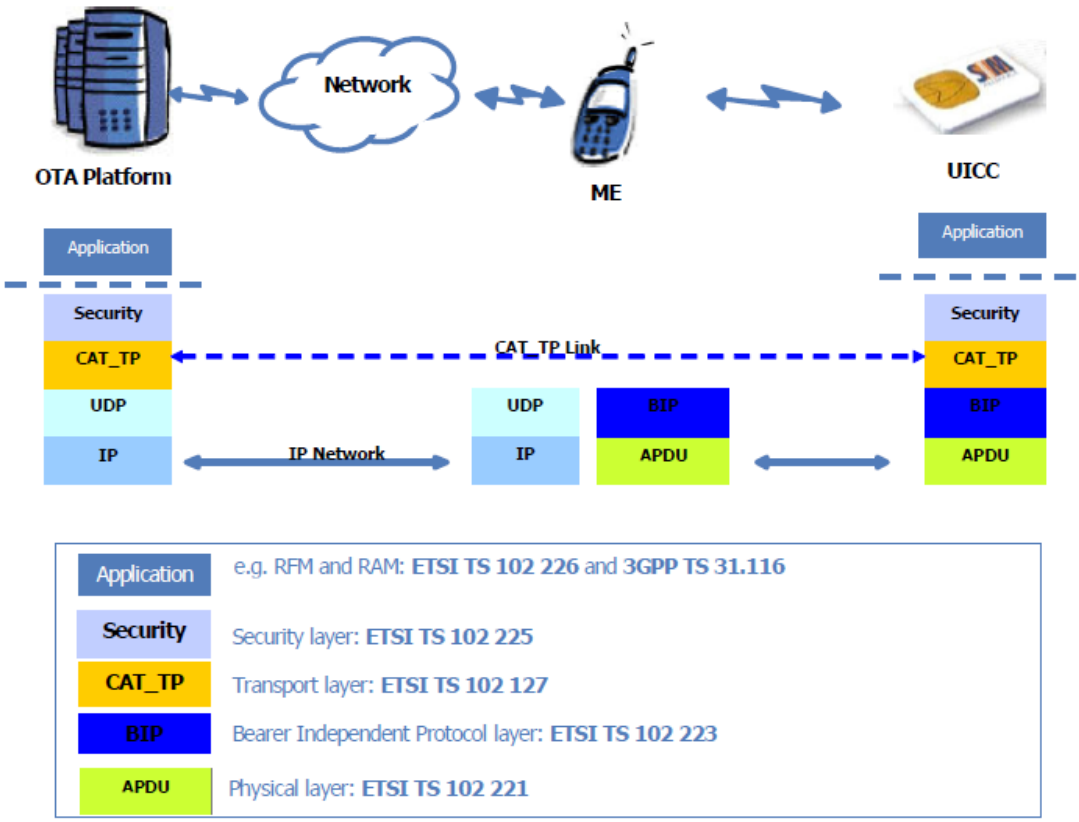


Figure 10: Overview CAT_TP protocol layer with associated layers in OTA platform, UE & UICC

5.3.4 HTTPS Transport Layer for Remote Management (HTTP/TLS_PSK)

This new protocol has been introduced in ETSI TS 102 226 Release 9 and GlobalPlatform 2.2 Amendment B; it smoothly integrates with existing Smart Card Web Server administration infrastructure and it can be used to perform remote file and applet management on the UICC.

In this document an overview of the protocol is given, but a more detailed description will be provided in a future work.

The HTTP protocol is used on top of TLS to provide encapsulation of the data and information about the receiving entity. TCP/IP transport, from SIM to ME, is provided by the Bearer Independent Protocol specified in ETSI TS 102 223 [11] or by a direct IP connection as specified in TS 102 483 [17]. This Transport Layer can be used by an Application Provider to perform Remote Application Management (RAM) of its application i.e. to load, install and personalize using the HTTP protocol (see RFC 2616 [45]) and PSK-TLS security (see RFC 4279 [46]) over TCP/IP network. ETSI TS 102 226 [14] Release 9, adds RFM support to HTTPS Transport protocol.

To start an administration session the Security Domain/File System Manage in the SIM needs to be triggered in order to start the administration session. The remote administration server, or a delegated authorized entity, shall trigger the administration session. The triggering of the Security Domain may result from:

- an external event, for example a message sent by a remote entity or by an off-card entity,
- an internal event, for example a timer,
- an application using a dedicated API method.

The Security Domain shall receive a triggering message. The Security Domain will handle the administration session, using its own PSK TLS keys for the communication security. It is assumed that the Security Domain knows all parameters needed to establish a connection or to handle its security. These parameters can be parameters of the triggering message or the parameters of the Security Domain itself. See [34] § 4.7 about Administration session triggering parameters. Once started, the same session can be used also to carry SCWS administration commands. For a description of SCWS Full Administration protocol, see SCWS_SteppingStones (ref.[44], § 11) for more details

In the GlobalPlatform scenario, a third party communication network may be used if the Application Provider has no OTA capability. In this case third party shall not be able to access clear text of any confidential data and code belonging to the Application Provider and this requirement is assured by TLS Protocol that cipher communication between Application Provider and Remote Card Entity.

5.4 Security Layers

As the bearers mentioned above are only used for transport purposes, it is recommended to use the security protocol defined in ETSI TS 102 225 [13] on top of them; for a description of ETSI TS 102 225 security, refer to SteppingStones R7 (ref. [43]). Only for Remote Management over HTTP, security layer ETSI TS 102 225 should be strengthened by TLS-PSK Security Layer according to [47] (TLS Protocol) and [46] (TLS-PSK) specifications.

5.4.1 Security layer TLS_PSK (RFC 4279)

TLS (Transport Layer Security) provides a secure and reliable transport mechanism between two communicating parties. It provides confidentiality and integrity protection for the transport used. It can also provide unilateral or mutual authentication depending on the implementations. TLS works in a client-server model, where the initiator is called the Client and the responder is called the Server. In most cases, a TLS client can authenticate a TLS server using a public key certificate. Mutual authentication is possible using public key certificates or with pre-shared keys using PSK-TLS.

The security in RAM and RFM scenarios, for data exchange over TCP, is provided by TLS with pre-shared keys using TLS-PSK, even if ETSI TS 102 225 shall be applied to command APDU in accordance with ETSI TS 102 226 Release 9 specifications. The processing rules for messages that are protected using HTTPS are specified in Amendment B of the GlobalPlatform Card Specification v 2.2 [34].

If a TLS connection with the receiving entity is not already established, the sending entity shall send a triggering message as specified in Amendment B of the GlobalPlatform Card Specification v 2.2 [34] to the security domain handling the TLS connection for itself or for an associated application.

The Security Domain processes the PSK TLS over this communication channel to enable mutual authentication, confidentiality and integrity, using one of the following cipher suites:

For TLS 1.0 and TLS 1.1:
- TLS_PSK_WITH_3DES_EDE_CBC_SHA, as defined in RFC 4279 [46].
- TLS_PSK_WITH_AES_128_CBC_SHA, as defined in RFC 4279 [46].
- TLS_PSK_WITH_NULL_SHA, as defined in RFC 4785

For TLS 1.2:
- TLS_PSK_WITH_AES_128_CBC_SHA256, as defined in RFC 5487
- TLS_PSK_WITH_NULL_SHA256, as defined in RFC 5487

The PSK TLS key version and key id to be used to initiate the PSK TLS session are read in the triggering parameters. See [34] § 4.7 about Administration session triggering parameters.

Interoperability Note:
Support of a version of TLS implies support of previous versions, e.g. a card supporting TLS 1.2 is also supporting TLS 1.0.
A configuration supporting only TLS 1.2 and not TLS 1.0 may lead to backward compatibility issues.

5.5 Applicative Layers

The ETSI TS 102 226 [14] specification defines the Card Remote Management by OTA including the Remote File Management and the Remote Applet Management. This specification first defines the data formats used to send a remote management request in a Command Packet and to retrieve the result of this request or card data in the Additional Response data of a Response Packet. This specification then defines the commands available for the Remote File Management and for the Remote Applet Management. Following paragraphs are intended only to provide an overview of RAM and RFM applications; more details will be provided in future SteppingStones.

5.5.1 Remote Management Application data formats

Two different data formats are supported by Remote Management Applications. These two format was introduced by ETSI TS 102 226 [14] from the Release 6 onward. The Compact Data Format allows the Remote Management Application to execute several commands but can send back to the server only one response to an APDU command, i.e. only one outgoing command can be included in the script.

While the Expanded allows the card to send back to the server several responses to the APDU commands, i.e several outgoing commands can be included in the script. The Compact and Expanded data formats are distinguished by different TAR; in other words the same Remote Application can have two TARs: one for Remote Management by using Compact Data format and one for Remote Management by using Expanded Data format.

(a) Compact Remote Management Application data format

In the Compact Data format, each command packet contains a sequence of commands. Commands are concatenated in the command packet according to the following format:

CLA (1 byte)	INS (1 byte)	P1 (1 byte)	P2 (1 byte)	P3 (1 byte)	DATA (Opt. – length is specified in P3)

To retrieve the Response parameters/data of a case 4 command (Data in input to the card and response with data from the card) the GET RESPONSE command has to be issued following the case 4 command. The GET RESPONSE and any case 2 command (no data in input to the card but data in output form the card) shall only occur once in a command string and, if present, must be the last command in the string. See Stepping Stones R7 (ref. [43], § 19.1) for more details. If a proof of Receipt is required by the sending entity, the Additional Response Data sent by the Remote Management Application shall be formatted according to the following table (ref. [14] – v9.2.0 - §5.1.2).

Length	Name
1	Number of commands executed within the command script (see note)
2	Status bytes 'r '61'xx' procedure bytes of last executed command/GET RESPONSE
X	Response data of last executed command / GET RESPONSE if available (i.e. if the last command was a case 2 command or a GET RESPONSE)
NOTE: This field shall be set 'o '01' if one command was executed within the command scrip', '02' if two commands were executed, etc.	

Table 7: Format of Additional Response Data in Compact Data Format

(b) Expanded Remote Management Application data format

With the Expanded Remote Application data format it is possible to chain two or more scripts using Script Chaining TLVs packed in a BER-TLV (ref. [14] – v9.2.0 - §5.2.1). Two variants exist for the expanded remote command structure:

- The Command Scripting template is a BER-TLV data object as defined in TS 101 220 [5], i.e. it uses definite length coding (see Table 8).
- The Command Scripting template is a BER-TLV data object which uses indefinite length coding as defined in ISO/IEC 8825-1 (see Table 9).

Length in bytes	Name
1	Command Scripting template tag for definite length coding
L	Length of Command Scripting template= A+B+…C
A	Command TLV
B	Command TLV
...	...
C	Command TLV

Table 8: Expanded format of Remote Management application command "secured data" - definite length coding

Length in bytes	Name
1	Command Scripting template tag for indefinite length coding
1	Indicator for indefinite length coding (val'e '80')
A	Command TLV
B	Command TLV
...	...
C	Command TLV
2	End of content indicator (val'e '00'00')

Table 9: Expanded format of Remote Management application command "secured data" - indefinite length coding

A Command TLV can be one of the following:
- A C-APDU, containing a remote management command;
- An Immediate Action TLV, containing a proactive command or another action to be performed when it is encountered while processing the sequence of Command TLVs (this type of command was introduced by ETSI TS 102 226 [14] from the Release 7 onward);
- An Error Action TLV, containing a proactive command to be performed only if an error is encountered in a C-APDU following this TLV.
- A Script Chaining TLV as first Command TLV.

See Stepping Stones R7 (ref. [43], § 19.1) for more details.

(c) Implementation for TCP/IP

The variant of Expanded Data Format with indefinite length coding is recommended to be used for RAM/RFM over HTTPS.

5.5.2 Remote Applet Management

Remote Application Management on a card includes the ability to load, install and remove applications. It is performed using commands defined in the GlobalPlatform Specification (ref. [32] §11) that are summarized in Stepping Stones R7 (ref. [43], § 21.2). With these commands, Remote Administration Server is able to do Remote Load File loading, Application installation, Load File removal, Application removal, Application locking/unlocking, Application information retrieval.

(a) Parameters assigned to an application during installation

The INSTALL [INSTALL] command is able create an instance on card of an application previously downloaded. With this command, several parameters should be added in order to specify functionalities used by the application. These parameters can be different in dependence of the application type (Java Card Application, SIM Toolkit Application, USIM Toolkit Application, Contactless Application and so on). For a (U)SIM Toolkit Application, for example, one should assign security features (Access Domain, Minimum Security Level), execution priority inside the Java Card Runtime Environment (Priority Level), maximum number of ME timers used by the application (Maximum Number of Timers), number of items shown in SET UP MENU handled by the application (Menu Entries) and so on. These parameters are detailed in Stepping Stones R7 (ref. [43], § 21.2.3). GP2.2 Amd. C adds some parameters in order to configure Contactless Applications during installation (ref. §4.6)).

(b) Implementation for TCP/IP

When using remote APDUs to perform RAM over HTTPS, the header values defined in GlobalPlatform 2.2 – Amendment. B [34] applies. The RAM / HTTP communication flow is illustrated in the relevant GlobalPlatform specification (ref. [34] - Annex A5).

5.6 OTA management of contactless applications

5.6.1 Introduction: GlobalPlatform 2.2 Amd C

Amendment C to GlobalPlatform Card Specification v2.2 addresses how end-users can manage contactless services on a secure element and easily select which application they would like to activate.

The technology outlines how the mobile phone interface can be tailored by the card issuer to meet its needs and those of service providers, including the storage of commercial logos to enable simple brand recognition on a mobile phone screen.

This document also enhances the deployment and lifecycle management of contactless applications by detailing how new data can be controlled within this environment, such as contactless protocols parameters, logo, application family and end user priority.

With the finalization of Amendment C, GlobalPlatform has completed the first phase of its activity to enhance and support NFC technology.

When this amendment is used together with GlobalPlatform Card Specification v2.2 Amendment A, entitled Confidential Card Content Management, a service provider can:

- Manage their applications over-the-air on an end-users secure element
- Present their service visually on a mobile phone screen
- Pilot the behaviour of their applications within an NFC mobile phone, infrastructure, and the interaction between the secure element and a reader.

Today's process of choosing a payment card from a wallet will be replaced by the selection of a contactless service via the screen of a mobile handset.

5.6.2 What's new with GlobalPlatform 2.2 Amd C.

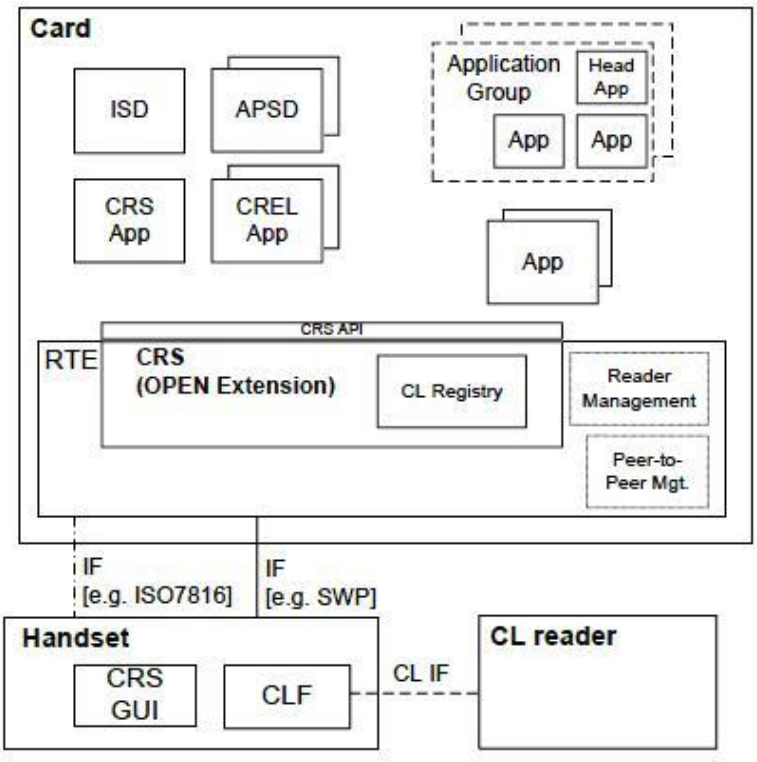


Figure 11: High level architecture overview

With the GP2.2 Amd C, we can find additional elements in the architecture that will be explained later in details:

CL Registry
The Contactless Registry adds functions to the GlobalPlatform Registry Services for managing Contactless Applications

CRS (OPEN Extension)
The Contactless Registry Service (CRS) is an extension of the OPEN to manage and provide access to contactless registry parameters

CRS Application
The CRS Application is an optional component designed for the management of Contactless Applications by the end user

CREL Application
A Contactless Registry Event Listener (CREL) Application is an Application interested in being notified of the changes occurring to one or more Contactless Application

CRS GUI
This handset-side component allows the end-user to interact with the on-card CRS Application

CL Reader
The Contactless Reader represents the card external contactless communication interface allowing on-card and off-card Contactless Applications to interact with each other.

5.6.3 Cumulative features

The GP2.2 Amendment C also introduces cumulative features. In particular, we find a Cumulative Granted Memory and a command, the Cumulative Delete, used to delete a complete Security Domain hierarchy.

(a) Cumulative Granted Memory

Amendment C introduces a new install parameter that may be used to set up Cumulative Granted Memory (CGM) for a Security Domain and its sub-hierarchy.

Using the Install Parameter for cumulative granted memory, it is possible to specify two types of Granted Memory: Volatile and Non-Volatile.

The INSTALL Command 'EF' has two Optional parameters: '82' and '83' whose values define respectively the amount of Cumulative Granted Volatile and Cumulative Granted Non-Volatile Memory.

Tag	Length	Value Description	Presence		
'EF'	Var	**Parameters**			
		Tag	**Length**	**Value Description**	
		…	…	Other Parameters	
		'82'	2 or 4	Cumulative Granted Volatile Memory	Optional
		'83'	2 or 4	Cumulative Granted Non-Volatile Memory	Optional

Interoperability Note:
The implementation of this feature may be limited, or otherwise impacted, by the specific runtime environment. Specifically, implementations of CGM for Volatile memory on the Java Card™ runtime environment may not be interoperable.

The Cumulative Granted Memory (CGM) specifies the exact amount of memory granted to a Security Domain, to its associated Applications and its entire sub-hierarchy.

This cumulative memory must be considered as reserved and allocated to the Security Domain, this means that the sum of the memory of all the applications and the SD itself must not exceed this amount but at the some time it is exclusively reserved and available to the Applications.

The scenarios that have to be considered for the Cumulative Memory management are related to

- Installation
- Deletion
- Extradition (from a Security Domain to another)
- Update

Application Installation:

When an Application is installed, the allocation of the Memory performs according to the following rules:

Precondition	Type of Memory Allocated
Memory reserved to the Application	Reserved Memory
Reserved Memory Exhausted	Cumulative Granted Memory
No Reserved Memory Specified	Cumulative Granted Memory
Memory reserved to the Application greater than Memory Quota	
Memory reserved to the Application greater than remaining amount of Resrved Memory and remaining amount of Cumulative Granted Memory	

Application Deletion:

When deleting an Application, the greater amount between Reserved Memory (if any was assigned to the Application) or the total amount of memory allocated by the Application, is credited back to the Cumulative Granted Memory.

Application Extradition:

When an Application is extradited, the allocation of the Memory performs according to the following rules:

Precondition	Type of Memory Allocated
the target Security Domain is assigned a CGM amount	the greater between Reserved Memory (if any was assigned to the Application) and the total amount of memory allocated by the Application

Precondition	Type of Memory Allocated
the target Security Domain is in the scope of a CGM amount	the greater between Reserved Memory (if any was assigned to the Application) and the total amount of memory allocated by the Application
the origin Security Domain (associated with the Application) was assigned a CGM amount	the greater between Reserved Memory (if any was assigned to the Application) and the total amount of memory allocated by the Application
the origin Security Domain (associated with the Application) is in the scope of a CGM amount	the greater between Reserved Memory (if any was assigned to the Application) and the total amount of memory allocated by the Application

CGM Update:

The policy for updating the CGM amount is configuration dependent.

When decreasing CGM, the new value shall not be lower than the total amount of memory allocated or reserved by the hierarchy.

(b) Cumulative Delete

Cumulative Delete is an option for a Security Domain with the Global Delete privilege (e.g. an Issuer) to delete a complete Security Domain hierarchy in case of technical problems or according to changing business relationships.

According to GP2.2, the deletion of Content may fail, for example, when other Applications or other Packages present in the card maintain references to the Package.

Even if these limitations can be considered useful, because they guarantee that the UICC always remain in a coherent status, they represent an obstacle for the UICC Issuer/Owner to fully control the card.

For example, an application provider could install a malicious application on its APSD. In this case, none could remove it using the simple DELETE Command.

This situation, a in a GP2.2 Authorized Management scenario, it would be considered unacceptable for the MNO. Now, with the Cumulative Delete command, the Issuer/Owner, using its Security Domain is able to force the deletion of single applications or a whole APSD.

Considering the list of possible failure causes, a Cumulative Delete process provides a solution to delete the root of a Security Domain hierarchy and all associated Packages or Applications.

The GlobalPlatform and the underlying operating system dependencies are taken into account by logically deleting Packages and Applications with open references, but internally keeping the Packages and Applications until the references and relations are resolved.

This allows also extradited Applications to continue using the code of its Package.

According to GP2.2 (9.1.3.4), if the command issuer has a Global Delete Privilege, it is possible to issue a Cumulative Delete command.

This privilege (see section 9.1.3.4 of GP2.2) provides the capability to remove any Package or Application from the card even if the Package or Application does not belong to this Security Domain (see section 9.5 of GP22).

When the Cumulative Delete command is issued, security checks are performed then, for each Application and Packages, the OPEN checks references to and from other Applications and Packages.

In case of Application deletion, if other Applications or Packages, not part of the same hierarchy, maintain references to any data within this Application, then the OPEN resets these references to null, while, in case of Package deletion, if other Applications or other Packages that are part of the same hierarchy make references to this Package then the OPEN will delete the referencing Applications, the referencing Packages and the Package itself.

If other Applications or other Packages not part of the same hierarchy make references to this Package, then the OPEN will mark this Package as Logically Deleted with References.

Packages which are marked as Logically Deleted with References remain physically in the Mutable Persistent Memory and in the GlobalPlatform Registry.

The GET STATUS command (see section 11.4 GET STATUS (Logically Deleted with References) Command) can be used to retrieve a list of all Logically Deleted with References marked Packages. The OPEN will return this list to any Security Domain with the Global Registry privilege.

5.6.4 Contactless parameters management in Amd C

A Contactless card usually hosts many applications from several service providers. Many of those applications, however, have their requirements in terms of the NFC parameters that are configured at the HCI initialization time (see Annex A). If two applications require different values for the same parameters, they are in conflict and they cannot be available together on the contactless interface. As an example, if two applications ISO 14443-A require different values for the UID, the HCI framework does not know which is the right one to set in the CLF parameter. So, one of the two application has to take precedence over the other.

However establishing which application takes precedence depends on several factors: as an example, if an user has two different credit cards applications that generates a conflict, the user shall decide which is the one to be used for the payments. In this case, a user interface is required to decide which has the higher priority. In other scenarios, the operator is in charge of setting the proper priority. As an example, when abroad some applications could be automatically deactivated due to agreement between parties.

As a consequence, the priority of applications depends on the business logic and on the policy that the operator establishes for the UICC. In order to allow the MNOs to define the business logic and to manage the Contactless application, a new concept has been introduced that is the CRS application, which is a privileged application (only one per UICC) that is able to activate and deactivate the visibility of the other applications by using specific API.

Developer tip:
As the CRS application is one per Secure Element, it is expected that it is directly provided and controlled by the MNO.

5.6.5 Contactless activation state

An application that is not available over the contactless interface might also be required to be available over other interfaces; as an example, even when a Contactless payment application is disabled, so contactless payments are not operative, the user could be interested in accessing information of the payment application such as the credit balance or the last operations.

To distinguish between the traditional Selectable state and the availability on the Contactless application, three new "sub-state" of the state SELECTABLE have been defined; the states are:

- ACTIVATED – meaning that the application is visible on the contactless
- NON_ACTIVATABLE – meaning that the application is disabled and hence is not visible
- DEACTIVATED– meaning that the application is disabled and hence it is not visible

DEACTIVATED and NON_ACTIVATABLE look very similar, but they differ in the concept that who's the entity enabled to reactivate the application. In case of NON_ACTIVATABLE, the application itself must authorize the reactivation by changing status to DEACTIVATED. In case of DEACTIVATED, also other privileged applications and the CRS application can change the status to ACTIVATED.

Every operation is subject to specific application privileges that are required to perform the operations. The required privilege depends also on the application role of the application in the meaning that:

- CRS Application that has the "Contactless Activation" privilege can perform any operation
- CREL applications are identified by the presence of the TLV list "CREL" in the GPCLRegistry
- Application requires the "Contactless Self-Activation" privilege to switch to ACTIVATED

In all the above states, however, the application is SELECTABLE, in the meaning that it is allowed to:

- Receive and process incoming messages over the ISO interface
- Receive and process incoming toolkit events, including OTA messages

Developer tip:
Deciding to move to NON_ACTIVABLE or DEACTIVATED depends on whether the application wants to control its re-activation.

5.6.6 Contactless parameters

The contactless parameters are the parameters by which the UICC configures the CLF. There is at most one set of parameters for each protocol and for each mode, which means that the set of parameters are the following:

- Card Emulation Mode Type A
- Card Emulation Mode Type B
- Card Emulation Mode Type F
- Reader Mode Type A
- Reader Mode Type B

The install parameters to specify Card Emulation Mode Types are specified in GlobalPlatform 2.2 Amendment C (see [35]), while the ones to specify Reader Mode Types are specified in ETSI TS 102 226.

Management of the two sets of parameters is quite different.

5.6.7 Card Emulation Mode parameters

Those parameters are configured at applet installation time and can be updated when the INSTALL [for Registry Update] command is executed.

For each technology (Type A, B or F), the parameters are created by merging the contactless parameters of all the applications on the card that:
- have that specific technology Contactless parameters in the INSTALL x INSTALL
- are ACTIVATED (and hence SELECTABLE)

The merging rules are quite complex and depend on how the applications are installed. For each parameter, two fields are indicated:
- the parameter itself
- the mask byte, that indicates the "flexibility" by which different parameters are accepted

Depending on the parameter, there are different types of masks and of rules to identify conflicts:
- In case of fixed value parameters, the rules are indicated by a mask saying which are the bits that, if required different by other applications, generate a conflict
- In case of "string" parameters, there are both masks on the length and on the value itself of the parameters

The check for a possible conflict is performed every time the status or the contactless parameters of a contactless application change.

This includes:
- INSTALL [For Make Selectable], INSTALL [For Install and make selectable] that instantiate a new set of contactless parameters
- DELETE of an application that remove such a set
- INSTALL [for Registry Update] that may change the contactless parameters
- SET STATUS that can move from ACTIVATED to NON_ACTIVABLE, etc.
- CRS API that may affect status of the applet
- CREL Notifier operations that may affect the status of the applet
- Applet operations that may affect the status of the applet

5.6.8 Card Emulation Mode application selection

In Card Emulation Mode Java Card applications are selected in order to process the APDUs that are incoming over the HCI interface. Applications are available for selection even if there are not the relevant install parameters, e.g. if an application does not have the Type A parameters and a SELECT by AID targeting the application is received on Type A, the application is selected anyhow.

On the other side, there are other parameters that may block application selection: a new TLV (Communication Interface Access Configuration) has been introduced to forbid application availability over a specific interface (ISO7816 or SWP).

Developer tip:
For security reasons, it is suggested to configure NAA and related applications (UICC, SIM, USIM) as not available on the SWP interface.

5.6.9 Reader Mode parameters

Card Emulation Mode applications are passive applications, in the meaning that the card is waiting for an APDU coming from an external entity, Reader mode applications, on the other side, are active applications, as they are in charge of sending APDUs and receiving response.

As a consequence, only one Reader Mode application can be ACTIVATED for each technology and only if it is in ACTIVATED it is allowed to search for targets by registering itself to the event EVENT_TARGET_DISCOVERED. So there is no possibility of conflicts in Reader Mode applications as only one application is ACTIVATED at any time.

To manage that, the HCI Framework may change the parameters of the Reader Mode at any registration or de-registration of the EVENT_TARGET_DISCOVERED. The parameters that are set are the one indicated in the installation parameters "Additional Contactless Parameters" TLV object (tag 'B0') that is specified in ETSI TS 102 226 (see [14]).

Tag	Length	Value	Presence
'86'	1	Reader mode protocol data Type A	Optional
'87'	N+2	Reader mode protocol data Type B	Optional

Where Type A parameters are:

Parameter	Value	Length
DATARATE_MAX	Maximum data rate supported as defined in TS 102 622 (see [21])	1

And Type B parameters are:

Parameter	Value	Length
AFI	Application family identifier as defined in TS 102 622 (see [21])	1
HIGHER_LAYER_DATA_LENGTH	Length of HIGHER_LAYER_DATA	1
HIGHER_LAYER_DATA	Higher layer data as defined in TS 102 622 (see [21])	N

Rules to migrate to ACTIVATED state for the Reader Mode applications are the same as for Card Emulation Mode, with the additional requirement that only one Reader Mode applet can be ACTIVATED for each technology. In order to present the Reader Mode applications to the User, usually the User Interaction Parameters are present for Reader Mode application.

5.6.10 The User Interaction parameters

The Contactless parameters are used by the CRS Application to understand which applications are in conflicts and to choose among the applications the ones to deactivate in order to solve the conflict.

In GlobalPlatform 2.2 Amendment C, the CRS Application can use specific information in the contactless registry to create a menu to present to the user, in order to allow the user choosing which the applications to activate are.

That information is named "User Interaction parameters" and may be present for all the applications on the card. They include:
- A textual name for the application (e.g. "Credit Issuer A")
- An Image (e.g. a JPEG) used e.g. if the CRS Application supports the SCWS interface
- An URL to a resource that can be either local (provided by the SCWS) or remote
- An Application Family, one byte coded as indicated in ISO/IEC 14443-3 (e.g. 0x10 means "Transport", 0x20 means "Financial", etc.)

When the CRS application is invoked by the user or by any event, it may create and show a menu displaying all the applications with the above information to allow the user to choose which applications should be activated or deactivated.

5.6.11 Group of Applications

Some services are composed by several applications linked together; as an example, a banking service may offer applications for:
- A Credit Card
- A Debit Card
- A Loyalty Card

All those services are linked each other, in the meaning that if the user request the activation of one of them, it actually wants to activate all the linked applications as well.

To achieve that, it is introduced the concept of *Application Group*.

An Application Group is composed by one and only one Head Application and one or more Member Applications. When an operation of Activation/Deactivation is performed on the Head Application, it is performed also on all the Member Applications; the CRS will then show to the user only the Head Application as all the actions are propagated on the Members Application.

An Application shall not be a member of more than one Application Group.

Not all the Applications require being part of a group: an Application that is neither a Head Application, nor a member of an Application Group, is a Standalone Application. At any time, a Standalone Application may register itself as a Head Application, hence defining a new Application Group, or as the member of an Application Group.

The Head Application is associated with a Group Authorization List defining the possible membership of the group as a list of AIDs. Applications referenced (by AID) in this list may request to join the group.

5.6.12 Updating of application information and notifications

CRS Application is then heavily accessing the GPCLRegistry to retrieve information on application status and contactless parameters and to show user information to ask user to change the activated applications; however those parameters may change: all the parameters are set in INSTALL [for INSTALL] and hence can be updated by the command INSTALL [for REGISTRY UPDATE]; moreover, specific GlobalPlatform APIs allow the Application and other authorized Applications to update the parameters content. As those information are used by the CRS Application, it must be notified by any change in those parameters; those change include:
- Installation of a new application
- Changing in the status, deletion of an existing application
- Update of any User or Contactless Parameter

– Joining / leaving a group, or becoming a Head Application

A complete list of events is indicated in the CLAppletEvent interface.

Also the Application itself and specific applications defined to monitor those events (Contactless Registry Event Listener (CREL) Applications) can be triggered by the above events.

The Application in order to be triggered has to implement the interface `CLAppletEvent` as an extension of the `Applet` class.

The CREL Applications has to implement the `CRELApplication` interface as an extension of the Applet class; moreover, the list of Applications monitored by a specific CREL Application is indicated in the `GPCLRegistry` information and so controlled by means of INSTALL commands or by the `GPCLRegistry` APIs.

Developer tip:
Contactless parameters are changed in the CLF as soon as they are updated in the registry and the SWP/HCI protocol allows the change, i.e.:
– The SWP line must be ACTIVATED
– In case of card emulation no transaction must be ongoing; if a transaction is ongoing, the EVT_FIELD_OFF notifies the UICC when the transaction is over
– In case of reader mode, parameters should be updated before RF field is originated, however some CLFs could support parameters update when searching for a target.

5.6.13 INSTALL for Registry Update Parameters

As above mentioned, a way to update contactless parameter of an applications is to execute an INSTALL [for REGISTRY UPDATE] command. With this kind of INSTALL it is possible to:

– modify the **Implicit Selection Parameter**: this parameter specify is to be implicitly selected on one (or more) specific card interface(s) for one (or more) specific logical channel(s) (see GP2.2 [32] - §11.1.7);

– modify the **Global Service Parameters**: these parameters may associate the Application to one or more service names. Each service name is coded on two bytes. The first byte identifies the service family, the second the service id within that family (see GP2.2 [32] - §8.1.3);

– modify the **Restrict Parameter**: this parameter is applied to the APSD referenced in the Application AID passed with the command. When this application AID is not present, the Restrict Parameter applies to the ISD. With this parameter it is possible to lists the functionalities that shall be disabled (see GP2.2 [32] - §11.5.2.3.7);

Moreover, according to Contactless services amendment C of GlobalPlatform 2.2, it is possible to use the INSTALL [for REGISTRY UPDATE] command to update these contactless parameters:

– modify the **Contactless protocol parameters**. These parameters are TLV structured; inside this structure it is possible to assign/update following functional parameters:
  » **Assigned Protocols for implicit selection** (see GP2.2 – Amd. C [35] - §6.6);
  » **Initial Contactless Activation State** (see GP2.2 – Amd. C [35] - §8.2);
  » **Contactless Protocol Parameters Profile** that is composed of two sub-parameters:
  - **Profile Value for Protocol** (see GP2.2 – Amd. C [35] - §11.2.2);

- **Contactless Protocol Profile Identifier** (see GP2.2 – Amd. C [35] - §11.2.2);
  - » **Recognition Algorithm** (see GP2.2 – Amd. C [35] - §6.5);
  - » **Continuous Processing** (see GP2.2 – Amd. C [35] - §6.4);
  - » **Communication Interface Access Parameters** (see GP2.2 – Amd. C [35] - §5.2);
  - » **Protocol Parameters for Type A (Card Emulation Mode)** (see GP2.2 – Amd. C [35] - §4.6);
  - » **Protocol Parameters for Type B (Card Emulation Mode)** (see GP2.2 – Amd. C [35] - §4.7);

– modify the **User Interaction Parameters**. These parameters are TLV structured; inside this structure it is possible to assign/update following functional parameters:
  - » **Display Control Template** (see GP2.2 – Amd. C [35] - §3.2);
  - » **Head Application** (see GP2.2 – Amd. C [35] - §3.7.2);
  - » **Add to the Group Authorization List** (see GP2.2 – Amd. C [35] - §3.7.5);
  - » **Remove from the Group Authorization List** (see GP2.2 – Amd. C [35] - §3.7.6);
  - » **Add to the CREL List** (see GP2.2 – Amd. C [35] - §3.8.3);
  - » **Remove from the CREL List** (see GP2.2 – Amd. C [35] - §3.8.4);
  - » **Policy Restricted Applications** (see GP2.2 – Amd. C [35] - §3.3);
  - » **Application discretionary data** (see GP2.2 – Amd. C [35] - §3.4);
  - » **Application Family** (see GP2.2 – Amd. C [35] - §3.5);
  - » **Display Required Indicator** (see GP2.2 – Amd. C [35] - §3.6);

– modify the **Additional Contactless Parameters**. These parameters are specific for UICC and are specified in ETSI TS 102 226 R9 (see ETSI TS 102 226 R9 [14] - §8.2.1.3.2.8). Inside the TLV structure of these parameters, it is possible to assign/update following functional parameters:
  - » **Reader mode protocol data Type A** (see ETSI TS 102 226 R9 [14] - §8.2.1.3.2.8.1)
  - » **Reader mode protocol data Type B** (see ETSI TS 102 226 R9 [14] - §8.2.1.3.2.8.2)

– modify the **Cumulative Granted Volatile Memory** (see GP2.2 – Amd. C [35] - §9);

– modify the **Cumulative Granted Non Volatile Memory** (see GP2.2 – Amd. C [35] - §9);

5.7 Main features added in ETSI Release 7 and ETSI Release 9 releases

From the NFC interoperability point of view, the most important changes on the ETSI TS 102 225 / 226 came from the R7 and later with R9 releases.

5.7.1 Summary of all features introduced by Release 7 and Release 8/9 of ETSI specifications

ETSI TS 102 225 [13] / 3GPP 31.115 [38]	
Main features introduced by Release 7	Main features introduced by Release 8 and 9
Introduction of secured data download for USSD (ref. [38] – v7.0.0 - §6, Annex A)	Secured message structure for HTTP (ref. [38] – v7.0.0 - §8)
	Addition of the capability to multiplex RAM and RFM sessions over a single CAT_TP link in a BIP session. (ref. [13] – v8.0.0 - §6)
	Addition of AES for encryption/decryption and cryptographic checksum (ref. [13] – v8.1.0 - §5.1.2, §5.1.3)
	Deprecate the use of single DES (ref. [13] – v8.1.0 - §5.1.2, §5.1.3)
	Addition of data download over IP (ref. [13] – v8.2.0)
	Update to GlobalPlatform Card Specification v2.2 (ref. [13] – v8.2.0)
	Addition of HTTPS (ref. [13] – v9.0.0 - §8)

ETSI TS 102 226 [14] / 3GPP 31.116 [39]	
Main features introduced by Release 7	Main features introduced by Release 8 and 9
Mandatory support of responses to push commands (ref. [14] - v7.0.0 - §9)	Automatic application data format detection (ref. [14] – v8.0.0 - §5.3)
Reservation of an application specific P1 value in the PUSH command (ref. [14] - v7.1.0 - §9.2)	Update to GlobalPlatform Card Specification v2.2 (ref. [14] – v8.1.0)
Add new response TLV in case of a bad formatted C-APDU (ref. [14] - v7.2.0 - §5.2.2)	Modification of Install(Install) for Applets registered to the SCWS using Toolkit resources (ref. [14] – v8.2.0 - §8.2.1.3.2.2.1)
Action TLVs in Command Scripting Template (ref. [14] – v7.3.0 - §5.2.1.2)	Support of Confidential Card Content Management (ref. [14] – v9.0.0 - §10)
	Remote Management over HTTPS (ref. [14] – v9.2.0 - §7.4, §8.3, Annex B)
	Indefinite length coding for remote command and response structures (ref. [14] – v9.2.0 - §5.2.1)
	Script chaining for RAM (ref. [14] – v9.2.0 - §5.2.1.4)
	Addition of push mechanism for TCP (ref. [14] – v9.2.0 - §9.1.4)
	Support of Constrained Load File list in the STORE DATA Command (ref. [14] – v9.2.0 - §8.2.1.8)
	Addition of confidential setup of security domains (ref. [14] – v9.2.0 - §10.3)
	Addition of install parameters for contactless applications (ref. [14] – v9.2.0 - §8.2.1.3.2.8)

6. Testing NFC & SWP/HCI

6.1 Overview of terminal & card conformance test benches

In the below schematic diagram are represented the major interfaces concerned in a Telecommunication Contactless device and their corresponding simulations and test benches.

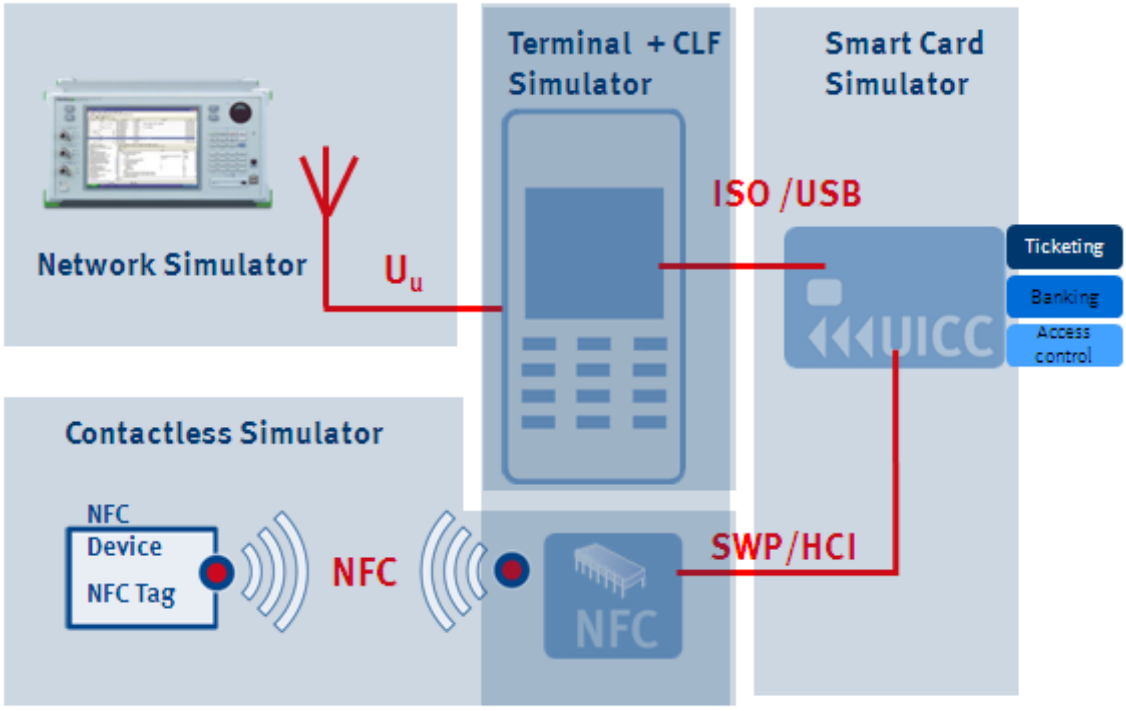


Figure 12: A Contactless Terminal & its interface testing

- The Uu interface (radio interface between the mobile and the radio access network) is specified by 3GPP and is out of the scope of this document. ("Uu" means UTRAN-UE interface, with UTRAN =UMTS Terrestrial Radio Access Network, and UE =User Equipment)
- The ISO/USB interface is specified by the ETSI
- The SWP/HCI is specified by the ETSI, see section 5.3.7
- The NFC interface is specified by the ISO/IEC, ETSI, ECMA, NFC Forum see the details in this section :

6.2 Existing Contactless interfaces

One can distinguish three different Contactless interfaces, the latter one being out of our scope:

- The NFC Devices defined in the ECMA 3–0 - ISO/IEC 18092)
- The Proximity card & Reader (defined in the ISO/IEC 14443)
- The Vicinity card & Readers (defined in the ISO/IEC 15693, longer operating distance 1-1.5 metres, out of our scope)

Below are explained the main technical differences between **NFC** and **Proximity Contactless**:

Table 10: NFC versus Contactless

	ISO/IEC 18092		ISO/IEC 14443*	
	NFC devices		Contactless Smart Card systems	
	Initiator	Target	Reader/writer	Contactless smart cards
Communication principle	Active communication mode: both initiator and target generate RF field		Reader generates RF field and card answers using load modulation	
Initialization	Passive communication mode : Initiator generates RF field and target answers using load modulation		Initialization and anticollision	
Speed at initialization [kbps]	106, 212, 424		106 for ISO 14443-A 212, 424 for FeliCaTM*	
Communication protocol	NFC IP-1 data exchange protocol		ISO 14443 Transmission protocol MIFARER : fixed command set, FeliCaTM : fixed command set	

** Note: FeliCa™ does not follow all the ISO/IEC 14443*

6.3 Contactless interface testing

In the below table are recapped the core and test standards applicable to the major Contactless applications:

Table 11: Core specifications and test specifications

Committees & Standards	Test Method standards
ISO/IEC 14443: Identification cards -- Contactless integrated circuit cards -- Proximity cards	**ISO/IEC 10373-6:** Identification cards -- Test methods -- Part 6: Proximity cards
ICAO (ePassport) **ICAO DOC 9303** Machine Readable Travel Document (MRTD), based on ISO/IEC 14443 and ISO 7816-4	**ICAO**: RF Protocol and Application Test Standard for e-Passport – **part 2: Tests for air interface, initialisation, anticollision, and transport protocol**. **ICAO**: RF Protocol and Application Test Standard for e-Passport – **part 3: Tests for application protocol and logical data structure**. **ICAO**: RF Protocol and Application Test Standard for e-Passport – **part 4: E-passport reader tests for air interface, initialisation, anticollision, and transport protocol**.
ECMA 340– -- ISO/IEC 18092: Information technology– -- Telecommunications and information exchange between systems– -- Near Field Communicati– -- Interface and Protocol (NFCIP-1) **ETSI TS 102 190:** Near Field Communication (NFC) IP-1; Interface and Protocol (NFCIP-1)	**ECMA-35–6—ISO/IEC 22536:** Information technology– -- Telecommunications and information exchange between systems– -- Near Field Communication Interface and Protocol (NFCIP-– -- **RF interface test methods** **ECMA-–2--ISO/IEC 23917**: Information technolo– -- Telecommunications and informati– -- Exchange between systems (NFCIP-1– --**Protocol Test Methods**

Committees & Standards	Test Method standards
EMVCo (Banking Terminals) EMV Contactless Communication Protocol Specification v2.0 Level 1 Test Equipment Specificatio–s - PICC Manual, PCD Manual, CMR Manual, Gerber Files (not public, reserved to EMVCo subscribers) EMV Contactless Specifications for Payment Systems- Entry Point Specification v1.0 EMV Contactless Specifications for Payment Systems – Framework for Contactless Evolution – v1.0	**Level 1** Type Approval process tests compliance with the **electromechanical characteristics, logical interface, and transmission protocol** requirements defined in the EMV Specifications **Level 2** Type Approval tests compliance with **the debit/credit application** requirements as defined in the EMV Specifications
NFC Forum NFC Data Exchange Format (NDEF) Technical Specification NFC Forum Type 1 Tag Operation Specification NFC Forum Type 2 Tag Operation Specification NFC Forum Type 3 Tag Operation Specification NFC Forum Type 4 Tag Operation Specification NFC Record Type Definition (RTD) Technical Specification NFC Text RTD Technical Specification NFC URI RTD Technical Specification NFC Smart Poster RTD Technical Specification NFC Generic Control RTD Technical Specification NFC Logical Link Control Protocol (LLCP) Technical Specification NFC Forum Connection Handover Technical Specification **Plus candidate specifications:** Digital Protocol Candidate Technical Specification NFC Activity Candidate Technical Specification NFC Signature Record Type Definition (RTD) Candidate Technical Specification **Drafts:** NFC RF Analogue Technical Specification NFC Simple NDEF Exchange Protocol (SNEP) Technical Specification	**1st wave of certification:** Test Cases for the Digital Protocol Test Cases for the Type 1 Tag Operation Test Cases for the Type 2 Tag Operation Test Cases for the Type 3 Tag Operation Test Cases for the Type 4 Tag Operation Device Test Application Specification **2nd wave of certification (under dev):** Test Specifications / Cases for the NFC RF Analog Specification

6.4 Technical architecture of an NFC device

Below is a functional schematic diagram that explains the different levels of communication on an NFC device in the 3 possible use cases: Card emulation, Reader/Writer and Peer-to-Peer:

The first 4 layers of this schematic from RF Analog to Tag Operation and LLCP are covered by the 4 test suites within the NFC Forum certification program.

Figure 13: Different levels of communication using NFC

6.5 Certification

Certification programs are established to ensure Interoperability in a complex ecosystem.

Concerning Contactless in Telecommunications the Global Certification Forum (GCF) and PTCRB have included the SWP/HCI in the scope of terminal conformance testing, and similarly, the NFC Forum has introduced end 2010 a new certification program for the testing of the Contactless RF interface in terminals.

6.5.1 The GCF certification program

The GCF (Global Certification Forum) was initially part of the GSM Association for many years and is now a not-for-profit private company registered as "GCF Ltd".

It includes registered members from Mobile Network Operators as well as Terminal Manufacturers and other 3rd parties like Test Manufacturers, Laboratories, Smartcard Manufacturers, etc.

The GCF defines the mandatory tests for a communicating equipment (handset, PDA, notebooks, datacard, M2M device) before it can be released commercially on the market. The tests cover terminal behaviors from safety measures to protect the end-user (level of RF emitted by the handset, temperature reached by the battery, etc.) up to functional testing (like browser's behavior, UICC interface & SIM Application toolkit implementation, etc.) and also security testing to ensure that the handset will not damage the network (authentication procedure, network attachment, etc.).

The GCF thus requires testing from Terminal Manufacturers on each model. At the end of this process, a detailed report describing which tests were successful or failed is provided for the given model. Terminal Manufacturers can perform these tests themselves (self-certification) using accredited tools or can go through accredited Test Houses who provide this service in their labs.

The GCF itself does not specify directly any tests but it selects tests coming from such standard committees or forums as 3GPP, ETSI, OMA, etc.

Access to the test plan and test results is reserved to the GCF members.

The GCF and PTCRB have selected Test cases from the ETSI TS 102 694-1 [22] and ETSI TS 102 695-1 [24] for NFC terminals that support the CLF connexion to the UICC interface.

Useful links:
http://www.globalcertificationforum.org
http://en.wikipedia.org/wiki/Global_Certification_Forum

See also the PTCRB (Equivalent to the GCF for the US market):
http://en.wikipedia.org/wiki/PTCRB
http://www.ptcrb.com/

And the CCF (in the CDMA world)
http://www.globalccf.org/certification.php

6.5.2 The NFC Forum Certification program

The Near Field Communication Forum was formed to advance the use of Near Field Communication technology by developing specifications, ensuring interoperability among devices and services, and educating the market about NFC technology. Formed in 2004, the Forum now has 140 members. Manufacturers, applications developers, financial services institutions, and more all work together to promote the use of NFC technology in consumer electronics, mobile devices, and PCs.

The NFC Forum has initiated a compliance program with a new certification label for terminals passing their tests based on standards and interoperability of NFC devices:

The below consumer-recognizable trademark was created in order to guarantee the promise of compliance and interoperability on NFC terminals. **Industry-facing mark** (not intended for use on the device, but can be used on product packaging, websites, sales materials, warranties, manuals, etc.). Only for NFC Forum members:

When a device passes the NFC Forum certification testing program, the below "N-mark" logo can be used by manufacturers to indicate the spot on the device where on this device to trigger NFC activity.

Consumer-facing mark. No cost of use, licence free, available to everyone:

Useful links:
http://www.nfc-forum.org/certification/

6.6 NFC/ Contactless Testing

The Test industry has developed a range of test equipments to allow debugging and conformance testing on the Contactless interface. Basically it concerns spy tools, simulation and conformance tools, sometimes combined in the same equipment: NFC Terminal & card simulator

A Contactless simulator can act as a contactless smart Card or a Contactless Device in order to test its counterpart. It shall support:

- Analogue/digital Simulation of the ISO/IEC 14443 & ISO/IEC 18092 and relevant low-level and application-related specifications detailed in section 5.3)
  - Analogue simulation for electrical measurements on the physical interface
  - Digital simulation for the logical test cases
- It shall also contains a special Test Case development API to allow self development of test cases and to run validation campaigns using self-developed or standard test suites
- Such a tool can also embed the spy or oscilloscope capability to trace & monitor the execution of the test cases and facilitate in-depth debugging

- Remote control of the test case triggering for inclusion into customer test proprietary environment
- Test automation to reduce the man/machine interaction and reduce test duration

Main use case: validation, non-regression testing

6.6.1 NFC Terminal & card conformance test bench

Same type of simulator tool as above and supporting additionally:
– standard validated test suites for conformance testing, with the tool + these test suites validated by an accredited laboratory for certification use
– authentic test reports generation for the official certification

For example:
– A contactless reader simulator + an ISO/IEC 10373-6 test suite allows to test a contactless smartcard (used in ePassport testing, for example)
– A smartcard simulator + the dedicated NFC Forum test suites allows to pass the NFC-Forum certification.

Main use case: pre-conformance or conformance testing for certification (EMVCo, NFC Forum, GCF...)

6.6.2 NFC related Application/Os test specifications

Several test specifications are available and may concern contactless applications development:

(a) The GlobalPlatform Card Compliance Programs:

The following Compliance Programs are available
» The Card Specification v2.1.1 Compliance Package v2.0
» The UICC Configuration Compliance Package

Each Compliance Package includes:
» a document outlining mandatory functionality and optional features
» a Test Plan and a detailed Test Specifications
» a Test Suites with Test Scripts
» Test Tools and Test Execution Environments

The Card Specification v2.1.1 Compliance Package v2.0 corresponds to the test of the GlobalPlatform Card Specification v2.1.1 [36].

The UICC Configuration Compliance Package corresponds to the tests of the GlobalPlatform UICC Configuration v1.0.0.0.

The GlobalPlatform UICC Configuration [36] is an implementation guide for deploying GlobalPlatform Card Specification v2.2 [30] and the Confidential Card Content Management Amendment A [31] within the mobile services sector.

Note: the test of the other GlobalPlatform Amendments (e.g. GlobalPlatform 2.2 Amendment C [35]) might be included in the next version of GlobalPlatform UICC Configuration and Compliance Package.

(b) Java Card Technology Compatibility Kit (TCK):

The Java Card Technology Compatibility Kit is provided under license terms by Sun (Oracle). These kits include the test plan and an execution environment to run the tests. The test coverage corresponds to the related Java Card Specification. The

TCK version 3.0.1 corresponds to the Java Card Specification v3.0.1 [30] and the TCK version 2.2.2 corresponds to the Java Card Specification v2.2.2.

6.7 SWP/HCI conformance testing

The ETSI has released two series of conformance test specifications for the validation of the features specified in the corresponding SWP and HCI core specifications both in terminals and UICCs:

- Part 1 for Terminal testing
- Part 2 for UICC testing

And a specific HCI part 3 for Host controller testing

The purpose of these test specification is mainly to verify that the physical connexion between the UICC and the CLF chipset and also the communication and application layers between these two entities are complying with the below quoted core specifications.

Compliance to these standards ensures that cards and terminal satisfy the required level of interoperability for UICC-based service deployment on such devices.

Below are the references of these specifications:

Core specification	Test specification
ETSI TS 102 613 **UI–C - Contactless Front-end (CLF) Interface; Part 1: Physical and data link layer characteristics Also called SWP (Single Wire Protocol)**	ETSI TS 102 694 **Test specification for the Single Wire Protocol (SWP) interface**
	Part 1: Terminal features
	Part 2: UICC features
ETSI TS 102 622 **UI–C - Contactless Front-end (CLF) Interface; Host Controller Interface (HCI)**	ETSI TS 102 695 **Test specification for the Host Controller Interface (HCI)**
	Part 1: Terminal features
	Part 2: UICC features
	Part 3: Host Controller features

The main target for these above quoted test specifications are card manufacturers, CLF chipset manufacturers and handset manufacturers who can use these tests for final validation or non-regression.

Mobile Operators and Service Providers can also use them on the equipments they have acquired before their deployment on the field.

These tests do not substitute to proprietary unitary testing that need to be carried out by the Industrials during the product development phase.

The tool industry has already implemented these SWP/HCI test specifications for both the UICC and the Terminals.

See section 6.1 regarding the development & conformance tools.

6.7.1 SWP/HCI ETSI Terminal & card conformance test bench

A conformance test tool as described in section 5.6.2 with additionally:
- Support of the validated ETSI SWP/HCI test benches and its regular updates, see specification references [11-15]. Possibility to run all or a selection of tests from these SWP/HCI test suites
- Generation of **authentic** test reports, after the execution of the **validated test cases**
- Debug functionality with the monitoring features on the ISO7816, SWP S1(Terminal) & S2(card) signals, see details below
- Test automation to reduce the man/machine interaction and reduce test duration

Main use case: pre-conformance or conformance testing for certification

6.7.2 Spy tools on ISO14443 / NFC/ ISO7816 / SWP / HCI

"Spy" tools or "Monitoring" tools are also called "sniffers" and they exist as standalone products.

Some are dedicated to a single interface monitoring (contact or contactless) , others offer combined contact/contactless monitoring.

It shall be noted that most simulators and conformance test tools quoted above also offer this spy functionality.

Main spy features:
- Presentation of Physical layer of all relevant signals with possibility to perform accurate timing measurements and analogue scope function on the following interfaces:
- ISO/IEC 14443, NFC, ISO/IEC 7816, SWP S1(Terminal) & S2(card) signals
- Protocol layer view for data exchanged on the ISO/IEC 14443,NFC, ISO/IEC 7816 and SWP/HCI with frame, packet and Byte views
- Application layer view, with the interpretation of the Contactless, GSM, USIM, SIM Toolkit, HCI commands & events
- Synchronized views of ISO14443, NFC, ISO7816, SWP S1(Terminal) & S2(card) signals in a single tool or a combination of spy tools
- Search pattern function on all data signals
- Export functions of sessions logs to text files
- Remote control of the logging session triggering for inclusion into customer proprietary environment

Main use cases: troubleshooting on the field, in R&D or validation laboratories

Annex A

Example1: Typical Full SWP/HCI Session Init

The sample-trace below shows a SWP-session initialisation between UICC and CLF.

It is just an example for a SWP session initialisation. Due to the different registry values of various HCI-gates, the HCI commands for the session initialisation may be different for different cards and UICC vendors.

The example includes following items:

- ACT Protocol (SYNC-ID, Power Mode, Speed Capabilities)
- Sliding Window Size, set and check Session ID, Clear All Pipes)
- Open Link + Admin Management Pipes
- Open RF Pipe for ISO-A and ISO-B CardEmu + Registry Settings
- Open Connectivity Pipe

	CLF→ ←UICC	Command	P_{ID}	Description
ACT	←	69 FF FF 00		SYNC with SyncID 'FF FF' (no ID assigned), Capability Byte: basic speed
	→	62 01		Power mode indication -> Full Power mode
	←	60		ACT initialised ready
SHDLC -U	→	F9 04 00		Reset SHDLC; Sliding window size: 4
	←	F9 02 00		Reset SHDLC; Sliding window size: 2
	→	E6		Acknowledgement
SHDLC-I	←	80 81 03	01*	ANY_OPEN_PIPE: Admin
	→	81 81 80	01	ANY_OK
	←	89 81 02 01	01	ANY_GET_PARAMETER → SESSION_IDENTITY
	→	8A 81 80 FF FF FF FF FF FF FF FF	01	ANY OK; ID=FF FF FF FF FF FF FF FF (no ID assigned)
	←	92 81 14 ED 0D	01	CLEAR ALL PIPE; randomly generated SYNC_ID: 'ED OD'
	→	93 81 80	01	ANY_OK
	←	9B 81 03	01	ANY_OPEN_PIPE: Admin
	→	9C 81 80	01	ANY_OK
	←	A4 80 03	00	ANY OPEN PIPE: Link-Mgt.
	→	A5 80 80	00	ANY OK
	←	AD 81 02 04	01	ANY_GET_PARAMETER → HOST_LIST
	→	AE 81 80 00 01	01	ANY_OK; Param-value: 00 01 → host IDs
	←	B6 81 01 03 01	01	ANY_SET_PARAMETER → WHITE_LIST
	→	B7 81 80	01	ANY_OK
	←	BF 81 10 41 02 41	01	ADM CREATE PIPE: Connectivity-Gate
	→	B8 81 80 02 41 01 41 60	01	ANY OK new dynamic Connectivity Pipe-ID: 60
	←	80 E0 03	60	ANY_OPEN_PIPE: Connectivity
	→	81 E0 80	60	ANY_OK
	←	89 81 10 23 00 23	01	ADM_CREATE_PIPE: ISO Type A
	→	8A 81 80 02 23 00 23 40	01	ANY_OK new dynamic ISO Type-A Pipe-ID: 40
	←	92 C0 03	40	ANY_OPEN_PIPE
	→	93 C0 80	40	ANY_OK
	←	9B C0 01 01 FF	40	ANY_SET_PARAMETER → MODE

	CLF→ ←UICC	Command	P_{ID}	Description
	→	9C C0 80	40	ANY_OK
	←	A4 C0 01 02 F1 F2 F3 F4	40	ANY_SET_PARAMETER → UID_REG
	→	A5 C0 80	40	ANY_OK
	←	AD C0 01 03 20	40	ANY SET PARAMETER → SAK
	→	AE C0 80	40	ANY_OK
	←	B6 C0 01 04 04 01	40	ANY_SET_PARAMETER → ATQA
	→	B7 C0 80	40	ANY_OK
	←	BF C0 01 05 A1 A2 A3	40	ANY_SET_PARAMETER → APPLICATION_DATA
	→	B8 C0 80	40	ANY_OK
	←	80 C0 01 06 40	40	ANY_SET_PARAMETER → FWI, SFGI
	→	81 C0 80	40	ANY_OK
	←	89 C0 01 07 01	40	ANY_SET_PARAMETER → CID_SUPPORT
	→	8A C0 80	40	ANY_OK
	←	92 C0 01 01 02	40	ANY_SET_PARAMETER → MODE
	→	93 C0 80	40	ANY OK
	←	9B 81 10 21 00 21	01	ADM CREATE PIPE: ISO Type B
	→	9C 81 80 02 21 00 21 41	01	ANY OK new dynamic ISO Type-B Pipe-ID: 41
	←	A4 C1 03	41	ANY OPEN PIPE
	→	A5 C1 80	41	ANY OK
	←	AD C1 01 01 FF	41	ANY_SET_PARAMETER → MODE
	→	AE C1 80	41	ANY_OK
	←	B6 C1 01 02 11 22 33 44	41	ANY_SET_PARAMETER → PUPI_REG
	→	B7 C1 80	41	ANY_OK
	←	BF C1 01 03 00	41	ANY_SET_PARAMETER → AFI
	→	B8 C1 80	41	ANY_OK
	←	80 C1 01 04 00 00 00 41	41	ANY_SET_PARAMETER → ATQB
	→	81 C1 80	41	ANY_OK
	←	89 C1 01 01 02	41	ANY_SET_PARAMETER → MODE
	→	8A C1 80	41	ANY OK
	←	92 81 01 01 01 D2 BE 74 C5 11 0C 09 9E	01	ANY_SET_PARAMETER → SESSION_IDENTITY → D2 BE 74 C5 11 0C 09 9E
	→	93 81 80	01	ANY_OK
	(SWP interface ready for application APDUs)			

**) note: summary of pipesIDs used in this example:*
0x00 (fixed) Link-Management
0x01 (fixed) Admin
0x60 (dynamic) Connectivity (dynamic)
0x40 (dynamic) ISO-A CardEmulation
0x41 (dynamic) ISO-B CardEmulation

Example2: typical Short Session Init (Re-activation in Full Power Mode)

The following sample-trace shows a typical short session init in Full-Power Mode after SWP reactivation, e.g. when the CLF detects an RF-field or the ACTIVATE Toolkit command was sent by the UICC. The SYNC-ID and Session-ID match due to a previous Full-Session init and therefore the UICC does not issue the CLEAR-All-Pipes command.

The short session-init in Full-Power Mode has following changes compared to the full-session init:

- ACT-SYNC message from the UICC does not include the Speed-Capabilities byte
- no ACT-response from CLF with Power-Mode info
- GetSession ID is optional (greyed below), from Release 10 of ETSI 102 622 the HCI-command GetSession-ID must be omitted in a short session-init

	CLF→ ←UICC	Command	P_{ID}	Description
ACT	←	61 ED 0D		SYNC with SyncID= ED 0D → from previous session
SHDLC -U	→	F9 04 00		Reset SHDLC; Sliding window size: 4
	←	F9 02 00		Reset SHDLC; Sliding window size: 2
	→	E6		Acknowledgement
SHDLC-I	←	80 81 03	01	ANY_OPEN_PIPE: Admin
	→	81 81 80	01	ANY OK
	←	89 81 02 01	01	ANY GET PARAMETER → SESSION IDENTITY
	→	8A 81 80 D2 BE 74 C5 11 0C 09 9E	01	ANY_OK; SessionID= D2 BE 74 C5 11 0C 09 9E → from previous session
	→	80 C0 53	40	EVT CARD ACTIVATED → RF-field detected for ISO-A pipe
	(SWP interface ready for application APDUs)			

Example3: typical Short Session Init (Low Power Mode)

The following sample-trace shows a typical short session init in Low-Power Mode. The SYNC-ID and Session-ID match due to a previous Full-Session init in Full-Power Mode.

In this example the CLF indicates the Low-Power mode by omitting the ACT-Response that contains the Power-Mode indication.

	CLF→ ←UICC	Command	P_{ID}	Description
ACT	←	69 ED 0D 00		SYNC with SyncID= ED 0D → from previous session, Capability Byte: basic speed
SHDLC -U	→	F9 04 00		Reset SHDLC; Sliding window size: 4
	←	F9 02 00		Reset SHDLC; Sliding window size: 2
	→	E6		Acknowledgement
SHDLC-I	→	80 C0 53	40	EVT_CARD_ACTIVATED → RF-field detected for ISO-A pipe
	(SWP interface ready for application APDUs)			

Revision History

Version description:

V **X.Y.Z**

X → Major version

Y → Minor version: rose after changes in content

Z → Raised after changes in typing, grammar or formatting.

Revision	Date	Description	Author
V1.0.0	06/15/2011	First Synopsis	SIMalliance Interoperability Working Group