Mobile Connect SIM Applet Authentication Specification

Version 2.2.1
06 December 2022

This is a Non-binding Permanent Reference Document of GSMA

Table of contents

1 Introduction

1.1 Overview

Mobile Connect is a portfolio of mobile-based secure identity services delivered by mobile operators, that can be integrated into third-party Service Provider's applications to provide authentication, authorisation, and permissioned access to a User's attributes.

One of the key aspects of the Mobile Connect architecture is its support for "Pluggable Authenticators" such that a range of authenticators can be easily employed to meet different Operator/SP/user needs whilst also ensuring that Mobile Connect is future-proof and can accommodate new authentication mechanisms as they come along (e.g., providing support for advanced biometric authenticators or the inclusion of passive behavioural authentication methods).

This document specifies an authenticator mechanism based around the use of a Card Authentication Application provisioned to the user's SIM/UICC. This document may also use the term 'Applet' and 'SIM Applet' `to designate the Card Authentication Application.

Note: this specification focus on the UICC platform. Nevertheless, older platform like SIM card (2G) or native cards may be eligible for Mobile Connect. In that case this platform shall support additional features related to application management, see section 5.2.

In the rest of this document the term 'card' is used to refer to the SIM or UICC platform.

Note: Operators may employ additional interfaces and/or wrap the Card Authentication Application capability into their own B2B services (e.g., to provide ETSI MSS services directly to a B2B partner). However, this document is focused purely on the requirements for implementing the Card Authentication Application as part of the Mobile Connect proposition.

1.2 Scope

In Scope	Out of Scope
• The Authentication Card Application requirements • The security requirements for the Card Authentication Application, messaging and assets storage • The messaging specification between the Card Authentication Application and the Authentication Server (MSSP). • UICC platform support for SIM Applet Specifications	• SIM applet UI related concepts

Table 1 : Scope of the specifications

1.3 Audience

The target audience for this document are the Mobile Operator and SIM vendors service/technical departments who are considering implementing / deploying / upgrading the SIM applet module to v2.2[1].

[1] It is operator's choice whether to maintain multiple SIM applet versions or replace the old version with the new version. At the time of writing this document Mobile Connect supports v2.2

Readers of this document are expected to have familiarity with and a good understanding of the SIM Applet low level concepts, how SIM Applet messaging works and a good understanding of Mobile Connect.

1.4 Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

1.5 Definitions

Term	Description
Card manufacturer	Supplier of the SIM/UICC and resident software (e.g., firmware and operating system).
Device	Equipment, into which a SIM/UICC is inserted, that provides communication functions.
HOTP	HOTP is an HMAC based One Time Password algorithm. It is a cornerstone of initiate of OATH. HOTP was published as an informational IETF RFC 4226 [1] in December 2005
Issuer Security Domain	A security domain on the UICC as defined by Global Platform Card specification [7]
Mobile Network Operator	An entity providing access capability and communication services to its Customers through a mobile network infrastructure.
OATH	OATH is an industry-wide collaboration to develop an open reference architecture by leveraging existing open standards for the universal adoption of strong authentication. OATH is comprised of industry leaders.
OCRA	Algorithm for challenge-response authentication developed by the OATH. The specified mechanisms leverage the HMAC-based One-Time Password(HOTP) algorithm and offer one-way and mutual authentication, and electronic signature capabilities.
OTA Platform	An MNO platform for remote management of SIM/UICCs.
SCP80	Secure channel defined in GlobalPlatform Card Specifications[7] similar to ETSI 102 225[5] (also commonly called ‘3.48’)
SCP81	Secure channel defined in GlobalPlatform Card Specification Amendment B[27] (also commonly called ‘RAMoHTTP’ or ‘Remote Application Management over HTTP’

1.6 Abbreviations

Abbreviation	Description
AES	Advanced Encryption Standard
3DES	Triple Data Encryption Standard
3DES CB	3DES in Cipher Block Chaining mode

Abbreviation	Description
AP	Application Provider. Equivalent to Service Provider in the context of this document.
APDU	Application Protocol Data Unit
BCD	Binary Coded Decimal
DCS	Data Coding Schema
HOTP	HMAC-based One-time Password
HTTP	Hypertext Transfer Protocol
ISD	Issuer Security Domain
MAC	Message Authentication Code
MNO	Mobile Network Operator
MSSP	Mobile Signature Service Provider
OATH	Open Authentication
OCRA	OATH Challenge-Response Algorithm
OTA	Over The Air
RFU	Reserved for Future Use
SCP	Secure Channel Protocol
SCP80	Secure Channel Protocol 80
SSD	Supplementary Security Domain
SIM	Subscriber Identity Module
SM	Short Message
SP	Service Provider
SM PP MT	Short Message Point to Point Mobile Terminated
SM PP MO	Short Message Point to Point Mobile Originated
SMS	Short Message Service
SMSC	Short Message Service Center
SOAP	Simple Object Access Protocol
TLV	Tag Length value
UICC	Universal Integrated Circuit Card
WSDL	Web Service Definition Language

1.7 References

Ref	Doc Number	Title
[1]	RFC 4226	HOTP: An HMAC-Based One-Time Password Algorithm, December 2005
[2]	RFC 6287	OCRA: OATH Challenge-Response Algorithm, June 2011
[3]	ISO/IEC 7816-4:2005 / ISO 7816-4:2013	Identification cards – Integrated circuit cards - Part 4: Organization, security and commands for interchange
[4]	ETSI TS 102 223	Smart Cards; Card Application Toolkit (CAT)

Ref	Doc Number	Title
[5]	ETSI TS 102 225	Smart Cards; Secured Packet Structure for UICC based Applications
[6]	ETSI TS 102 226	Smart Cards; Remote APDU structure for UICC based applications
[7]		GlobalPlatform Card Specification v.2.2.1
[8]		GlobalPlatform Card Specification v.2.2.1 UICC Configuration v1.0.1
[9]		GlobalPlatform Card Specification v.2.2 Amendment D: Secure Channel Protocol 03 v1.1
[10]	3GPP TS 31.115	Secured packet structure for (Universal) Subscriber Identity Module (U)SIM Toolkit applications
[11]	3GPP TS 31.116	Remote APDU Structure for (U)SIM Toolkit applications
[12]	3GPP TS 23.040	Technical Specification Group Core Network and Terminals; Technical realization of the Short Message Service (SMS)
[13]	3GPP TS 31.111	Universal Subscriber Identity Module (USIM) Application Toolkit (USAT)
[14]	GSM 03.40	Digital cellular telecommunications system (Phase 2+); Technical realization of the Short Message Service (SMS) Point-to-Point (PP) (GSM 03.40)
[15]	GSM 03.48	SIM Toolkit Secure Messaging; GSM 03.48
[16]	GSM 11.14	Digital cellular telecommunications system (Phase 2+); Specification of the SIM Application Toolkit for the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface
[17]	FIPS-197 (2001)	Advanced Encryption Standard (AES)
[18]		"Applied Cryptography: Protocols, Algorithms, and Source Code in C", 2nd Edition, Bruce Schneier, John Wiley & Sons. Triple DES in outer-CBC mode is described in clause 15.2.
[19]	TS 123 038	ETSI TS 123 038: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Alphabets and language-specific information (3GPPTS 23.038)"
[20]	SMPP	SMS Forum. Short Message Peer-to-Peer Protocol Specification
[21]	ETSI TS 102 204	Mobile Commerce (M-COMM); Mobile Signature Service; Web Service Interface
[22]	WSDL	W3C Note 15 March 2001: "Web Services Description Language (WSDL) 1.1", http://www.w3.org/TR/wsdl
[23]	SOAP	SOAP Version 1.2 Part 1: Messaging Framework", W3C Recommendation 24 June 2003, http://www.w3.org/TR/soap12-part1/ "SOAP Version 1.2 Part 2: Adjuncts", W3C Recommendation 24 June 2003, http://www.w3.org/TR/soap12-part2/
[24]	NIST SP 800-57 Part 1	NIST Special Publication 800-57: Recommendation for Key Management – Part 1: General (Revision 3), July 2012
[25]	RFC 4493	The AES-CMAC Algorithm
[26]	ETSI TS 143 019	Subscriber Identity Module Application Programming Interface (SIM API) for Java Card; Stage 2

Ref	Doc Number	Title
[27]		GlobalPlatform Card Specification v2.2: Amendment B - Remote Application Management over HTTP v1.1.3
[28]	IDY.04	Mobile Connect Technical Architecture and Core Requirements

2 Authentication modes

The aim of the Mobile Connect Authentication service is to provide Service Providers with a range of authentication options, each authentication mechanism being mapped and certified to a defined set of Levels of Assurance based on ISO/IEC 29115 Clause 6.

This document defines and specifies an authenticator mechanism using an applet provisioned to the user's card. Three authentication modes have been identified, mapped to the LoA levels:

Level of Assurance	Description	Authentication mode
1	N/A	N/A
2	Single-factor authentication ("something I have")	B.1 Click OK
3	Two-Factor authentication ("something I have" plus "something I know")	B.2 Enter Personal Code
4	Multi-factor plus PKI	B.3 Multi-factor + Mobile Signature (certificates)

Table 2 Authentication modes

As shown in the table, three authentication modes are supportable via the Card Authentication Application:

- B.1 uses the possession of Device/Card as the factor used for authentication by requesting user consent through a simple button press on the mobile device: e.g., 'click OK', 'Press 1' etc.; the challenge to the Device/Card is returned back with a signed response authenticating the possession of the Device/Card.
- B.2 is similar to B.1 but with the addition of a Personal Code challenge to the user (user entered Personal Code is verified on the applet)
- B.3 Mobile Signature would be used specifically for those use cases where secure, non-repudiated identity assertion is required for verticals such as banking and Government; this level of service would require robust identity proofing and the deployment of a certificate chain. Detailed implementation is not considered within this specification.
- Note that the adoption of B.3 does not change the User Experience (UX) which will depend on whether B.3 is used in conjunction with B.1 or B.2. However, if B.1 "Click OK" is used in conjunction with B.3, the overall level of assurance will be classed as LoA3 and not as LoA4.

3 User Journey on Device

This section provides an overview of the different user flows for the Card Authentication Application mechanism and complements the service flows for the wider Mobile Connect proposition [28].

3.1 The "Click OK" journey

If the Service Provider only requires a low level of assurance it may be enough to simply ask the user to 'Click OK' to authenticate, authorise a transaction or provide consent (see section 2). In this scenario, the Card Authentication Application will pop-up a display text requiring user acknowledgement in the form of a click upon a “Yes/No” or “Cancel/OK” button. This pop-up display is triggered by the reception of a message that can only be sent by the Authentication Server (MSSP).

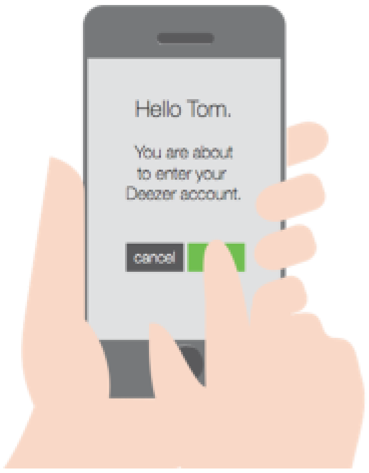


Figure 1 : Click OK Journey

It is recommended to display the message on the mobile device for a limited amount of time for the user to respond. After this limited amount of time, the Applet sends a time-out status to the Authentication Server (MSSP).

3.2 The “Personal Code” journey

If the Service Provider requires a higher level of assurance, it may require their users to authenticate, authorise a transaction or provide consent by entering their Personal Code. In this scenario, the Card Authentication Application will pop-up a display text and will require a Personal Code entry (4 to 8 digits). This pop-up display is triggered by the reception of a message that can only be sent by the Authentication Server (MSSP). The entry value is checked locally against the stored Personal Code value in the Card Authentication Application (see section 3.3).

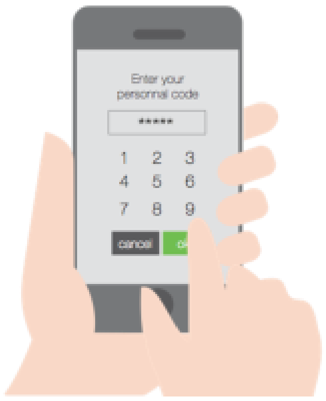


Figure 2 : Personal Code Journey

The Authentication platform should have the option to select a ‘one-step’ or ‘two-step’ user journey to meet local requirements or operators’ requirements:

one-step user-journey: display message pop-up and Personal Code input field are displayed in the same pop-up message

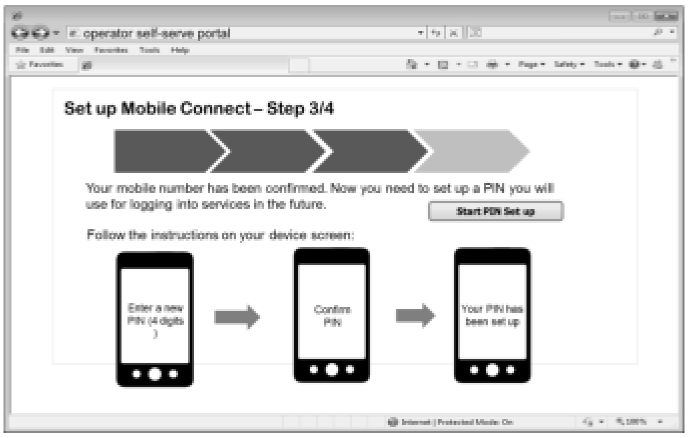
two-step user-journey: the display message pop-up must first be acknowledged by the user. It is then followed by the Personal Code input field pop-up.

The Card Authentication Application shall implement a retry mechanism and optionally a time out management (recommended) ; and in case of erroneous personal code entry, several retry pop-up (configurable) may be displayed before the Applet returns a failed result and the MNO shall block the Mobile Connect service until MNO perform a reactivation of the service*.*

In case of a new authentication request arrives and the previous one is still being processed; a FIFO model should be used for the requests. This will be handled by the MSSP and the ID Gateway.

3.3 Personal Code Creation Journey

When the Card Authentication Application is first enabled in the user's Card (as a result of user enrolment), the user will be required to select and register the Personal Code they wish to use to authenticate. This is conducted solely via the user's mobile phone so that their Personal Code remains private and only known to the User.


operator self-serve portal

Set up Mobile Connect – Step 3/4

Your mobile number has been confirmed. Now you need to set up a PIN you will use for logging into services in the future.

Follow the instructions on your device screen:

Figure 3 : Personal Code Create Journey

- The applet displays a text (Message Id #1 in Table 5) to request the user to enter their Personal Code. This text display is triggered by the reception of a message sent only by the Authentication Server.
- The user enters their Personal Code (or clicks on the "Cancel" button)
- The applet displays a text (Message Id #2 in Table 5) to request the user to enter their Personal Code a second time
- The user enters their Personal Code (or clicks on the "Cancel" button)
- If both Personal Codes are matching, the applet displays a confirmation message (Message Id #3 in Table 5) on the screen of the mobile phone and stores locally the new Personal Code value
- If both Personal Codes are NOT matching, the applet displays an error message (Message Id #4 in Table 5) on the screen of the mobile phone. If maximum number of attempts is not reached, the Applet shall restart the sequence from step 1; else the applet shall display a final error message (Message Id #8 in Table 5) indicating that the Mobile Connect service has not been properly activated.

3.4 Wrong Personal Code Journey /unblock

After a configurable amount of consecutive wrong Personal Codes, the Personal Code is blocked and cannot be validated anymore.

The Mobile Connect system is notified of the situation in the response message sent from the Applet. The Mobile Connect system sends an SMS informing of the situation and indicating the possible means that the end-user can use to unblock its Personal Code (this doesn't work with an iPad tablet).

One of the possible solutions is the following sequence:

The end-user navigates on the dedicated Mobile Connect portal and clicks on a 'Personal code unblock' link or button (or any equivalent user experience). This service is available without requiring any end-user authentication.

The Mobile Connect system sends a mail, on a private address mail (that is provided during the registration process), containing an URL link allowing to trigger the unblock sequence.

The end-user logs on its private mail account and click on the link that instructs the Mobile Connect system to send a specific command to unblock the Personal Code on the end-user mobile equipment (this command is generated and sent by the MSSP).

Another solution could go through a Customer Care service of the home MNO. The Customer Care agent, after having authenticated its end-user, may trigger the sending by the MSSP of the same specific command to unblock the Personal Code on the end-user mobile equipment (what happens in details before the call of the MSSP is out of scope this document). This is a more preferred solution and provides a more homogeneous solution with the PKI services, where this flow shall verify the identity of the user.

3.5 Reset Personal Code Journey

The end-user navigates on the dedicated Mobile Connect portal and authenticates using the Mobile Connect service. Then the end-user clicks on a 'Personal code reset' link or button (or any equivalent user experience). The Mobile Connect system sends a specific command triggering the Personal Code change on the end-user mobile equipment (this command is generated and sent by the MSSP).

Another possibility could be provided by a USSD code (or equivalent) sent from the end-user mobile equipment.

As for the previous case 'Wrong Personal Code Journey /unblock', another possibility would be that the end-user goes through a Customer Care service.

3.6 User cancels the Authentication request

In case the user cancels the Authentication request on the popup at the applet, the applet closes down. The MSSP sends error to the ID Gateway and the SP gets the Authorisation error and may display an appropriate message at the service consumption page.

3.7 Change smart card within the same Operator

At any time, the user (subscriber) may subscribe to additional Operator services (NFC...) requiring the issuance of a new card. For this version of Mobile Connect, the end-user will be requested to re-register for Mobile Connect.

3.8 Operator change

At any time, the user (subscriber) may change operator. In that case a new card will be issued to the end-user. The end-user will be requested to re-enrol to the Mobile Connect service through their new operator. Note that the end-user may retain their existing MSISDN.

3.9 Unsubscribe Mobile Connect

At any time, the user may unsubscribe Mobile Connect service. The end-user navigates on the dedicated Mobile Connect portal and authenticates using the Mobile Connect service. Then the end-user clicks on a ‘Unsubscribe’ link or button (or any equivalent user experience). The Mobile Connect system sends a command requesting a signed confirmation using the Mobile Connect service. If agreed by the end-user, the Mobile Connect system could either:

- Simply unsubscribe the end-user at MSSP level
- Or unsubscribe the end-user at MSSP level and also send a command to deactivate or delete the Mobile Connect service on the end-user mobile equipment (this command is generated and sent by the MSSP).

As for the previous case ‘Wrong Personal Code Journey /unblock’, another possibility would be that the end-user goes through a Customer Care service.

4 Mobile Connect authentication architecture with user’s SIM

Figure 4 illustrates the Mobile Connect architecture using a SIM Applet based authenticator. Note that INT1 and INT2 represent the Mobile Connect and Discovery APIs and are out of scope of this specification. Further details can be found in [28].

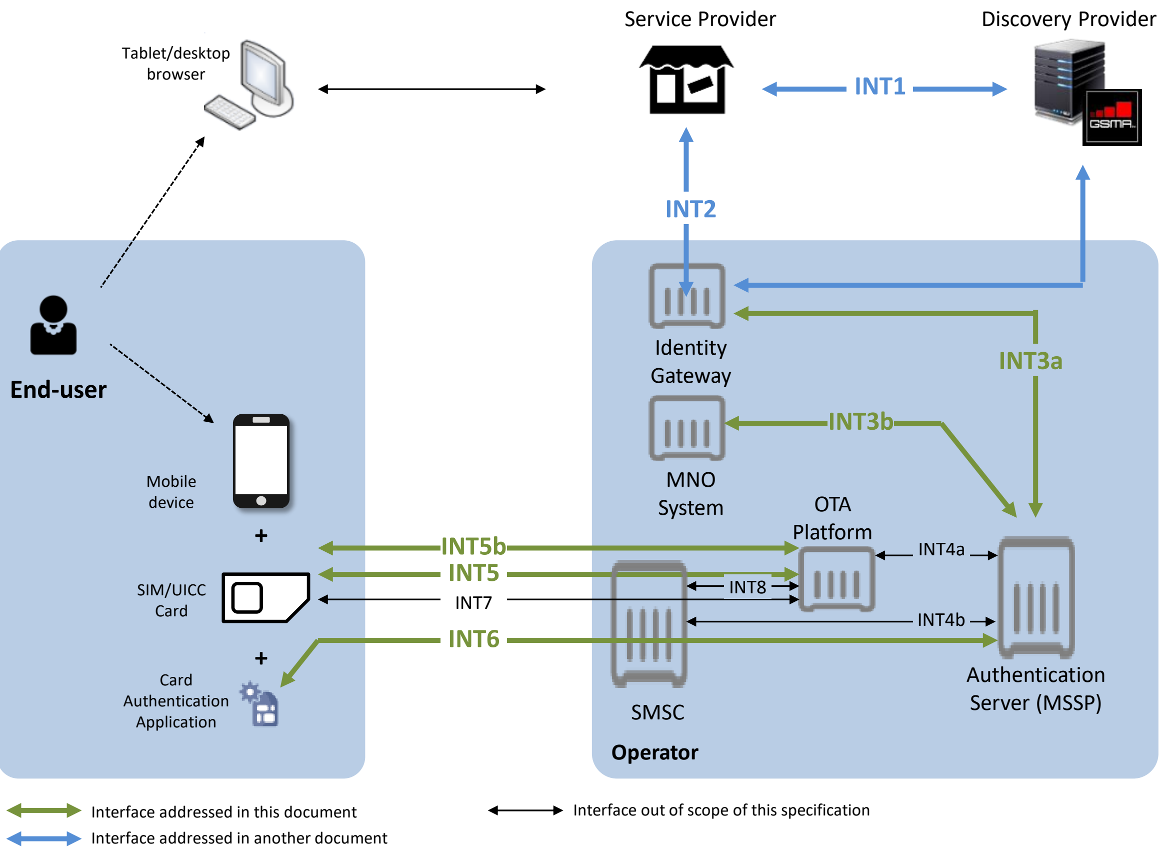


Figure 4 Mobile Connect architecture overview

This specification introduces the following new entities:

- OTA Platform: the component that is in charge to provide the card remote administration capabilities that is used in Mobile Connect solution to load and install the Card Authentication Application. The OTA Platform may also be used in certain deployment to provide the secure messaging for the MSSP to access the Card Authentication Application.
- SMSC: the component that is in charge to deliver/receive SMS to/from Mobile Device
- MNO System: this component refers to non-specified entities provided by the MNO to perform the procedures described in section 7, different than the Authentication/Signature request.

In some deployments the MSSP may incorporate the OTA Platform, and be seen as a single entity.

The INT3a (Identity GW to MSSP) is addressed in this document in section 9.1.

The INT3b (MNO to MSSP) is addressed in this document in section 9.2

The interface **INT5** (Card to SMSC/OTA Platform with SCP80) is addressed in this document in section 8.1.

The interface INT5b (Card to OTA Platform with SCP81) is addressed in this document in section 8.1.

The interface **INT6** (Card Authentication Application to MSSP) is addressed in this document in section 8.2. This interface defines the functions necessary to handle end-user authentication by the Authentication Server (MSSP). The functions are realized through APDU commands. The INT6 is used over the INT5 and INT4a or INT4b.

The interface **INT7** (Card to OTA Platform) for the purpose of the remote application management is out of scope of this document; nevertheless, the section 8.3 provides a description of the expected functions.

The interface **INT4a** (MSSP to OTA Platform) is out of scope of this specification. This interface provides the Authentication Server (MSSP) with the ability to send/receive secure messages to/from the Card Authentication Application. This interface shall be used in the case where the MSSP relies on secure messaging provided by the OTA Platform; handling the transport credential (e.g. 03.48 keys). This is discussed in section 6.3.1.1.

The interface **INT4b** (MSSP to SMSC) and **INT8** (OTA Platform to SMSC) **are out of scope of this specification**. These interfaces provide the ability to send/receive messages to/from the Card Authentication Application. An example of an interface implementation is the Short Message Peer-to-Peer Protocol Specification (SMPP) [20].

5 Card platform

The Mobile Connect solution targets mainly the UICC platform which provides a complete framework for management of applications (OTA loading, personalization...) and supports an interoperable Card Authentication Application (reducing the cost of deployment). Nevertheless, this specification doesn't the use of a SIM platform, in which case the SIM will have to provide additional features that are outside of the standards applicable to the SIM.

5.1 UICC platform

The UICC shall be compliant with the set of ISO, ETSI and 3GPP specifications applicable to UICC. The UICC shall especially be compliant with the set of specifications referenced in this document.

Main features required in the context of this specification:

- APDU management ISO/IEC 7816-4 [3] / ETSI TS 102 221
- Secure messaging over SMS as defined in 3GPP TS 31.115 [10] (and more generally in ETSI TS 102 225 [5]). See section 8.1
- Application toolkit as defined in 3GPP TS 31.111 [13] (and more generally in ETSI TS 102 223 [4])
- Application management, as defined 3GPP TS 31.116 [11] (and more generally in ETSI TS 102 226 [6]) and the GlobalPlatform Card Specifications [7].
- Optionally: RAMoHTTP as defined in the GlobalPlatform Card Amendment B [27]. See section 8.1

5.2 SIM platform

The SIM platform may also be eligible. The same main features as for UICC are requested:

- APDU management ISO/IEC 7816-4 [3] /GSM 11.11
- Secure messaging over SMS as defined in GSM 03.48 [13]
- SIM Application toolkit as defined in GSM 11.14 [16]
- Optionally: RAMoHTTP as defined in the GlobalPlatform Card Amendment B [27]. See section 8.1

The capability of loading, managing and personalizing an application has to be addressed by features out of standards applicable to SIM platform and can't be addressed in this specification. The MNO shall be able to handle such specific features within its OTA platform.

6 Card Authentication Application

6.1 Guiding principles and general requirements

1) The Card Authentication Application shall work across all mobile devices featuring a display and input capabilities (basic, feature and smart phones, tablets with SIM/UICC).

The device shall support:

- SMS-PP MT and SMS-PP MO (depending of the implementation choice in Section 6.3) as defined in ETSI TS 102 223 [4], and in 3GPP TS 23.040 [12] (or either GSM 03.40 [14] for device with SIM cards)
- As a minimum, the following set of commands as defined in ETSI TS 102 223 [4] and 3GPP TS 31.111 [13].
    - PROVIDE LOCAL INFORMATION (location information, IMEI, NMR, date and time, access technology) (optional)
    - Access to TERMINAL PROFILE
    - SEND SHORT MESSAGE (depending of the implementation choice in Section 6.3)
    - ENVELOPE (SMS-PP DOWNLOAD)
    - DISPLAY TEXT
    - GET INKEY
    - GET INPUT
    - RING TONE (optional)
    - Optional class 'e' commands for RoHTTP purpose
- Note: The behavior of some pro-active commands used in this specification may be different depending on the device. This means that end user experience may vary depending on device (e.g. time management, wake-up from standby mode…)

2) The Card Authentication Application shall work on UICC card with SIM application (2G) or/and (U)SIM application (3G/4G), or SIM card (2G) see section 5.2.

3) It shall be possible to load, install and configure the Card Authentication Application over the air (OTA)

4) The Card Authentication Application shall support the customer journey as it is defined and described within the "user Journey" chapter of this document, see section 3.

5) In order to ensure high completion rates using an OTA download deployment, the code (OTA package) of the Card Authentication Application size shall be as small as possible.

Depending on the targeted market segment, the Operator is free to choose the most appropriate deployment mode (loading at card issuance or OTA loading) for a given user.

6) The End-user owns and chooses the Personal Code.

The Personal Code value is defined during Personal Code creation.

7) The Personal Code is locally stored within the Card Authentication Application and not shared with the Server, the MNO or any other party. In particular it shall **not** be readable OTA by the MNO with RAM or RFM functions, and shall **not** be accessible from any other application in the SIM Card.

6.2 Authentication Handler

An Authentication Handler is an authentication method that is installed and configured on-board of the Card Authentication Applet and that the Authentication Server can invoke to authenticate the end-user.

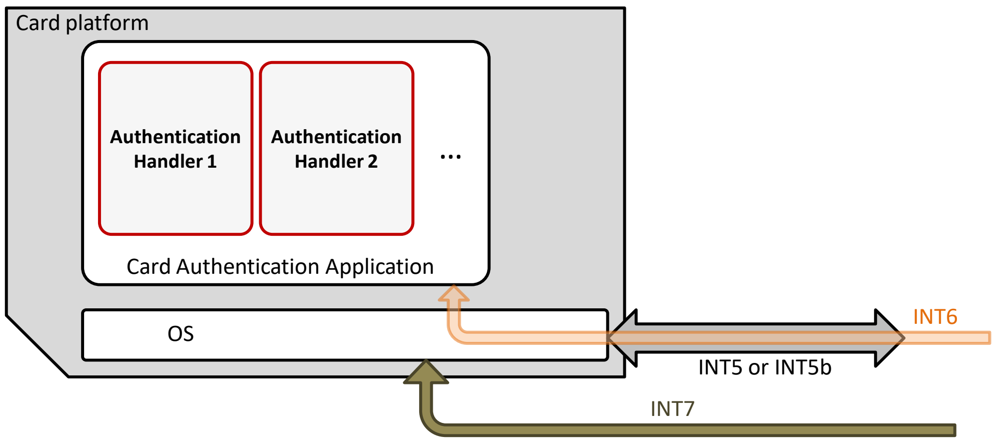


Figure 5 Authentication Handler

The various Authentication Handlers that are defined in this document are:

- “Click OK” with authentication based on secure messaging layer
- “Click OK” with authentication based on 3DES-CBC
- “Click OK” with authentication based on AES-CMAC [25]
- “Click OK” with authentication based on OATH OCRA
- “Personal Code input” with authentication based on secure messaging layer
- “Personal Code input” with authentication based on 3DES-CBC
- “Personal Code input” with authentication based on AES-CMAC [25]
- “Personal Code input” with authentication based on OATH OCRA

The first four Authentication Handlers are all implementing the LoA 2 (see section 2). The difference sits on the cryptographic algorithm that are used to sign the response sent to the Authentication Server

“Click Ok’ refers to the end-user journey described in section 3.1.

The four subsequent Authentication Handlers are all implementing the LoA 3 (see section 2). As for the previous, the difference sits on the cryptographic algorithm.

“Personal Code input” refers to the end-user journey described in section 3.2.

Authentication Handlers list will be enriched in further versions of this specification, including for instance:

- “Click OK” with authentication based on PKI(e.g. RSA, ECC)
- “Personal Code input” with authentication based on PKI (e.g. RSA, ECC)

A Card Authentication Applet may implement one or several Authentication Handlers. The number of supported Authentication Handler directly influences the final size of the Applet - the lower the number of Authentication Handlers that are supported, the smaller the Applet size. The choice of supported Authentication Handler may depend on applet provider choice, the deployment context driven by the MNO or various others criteria.

The Authentication Service Provider is free to instantiate one or several Authentication Handlers based on its own capability to support those Authentication Handlers, the Applet capabilities, etc.

The Card Authentication Application shall provide an OTA discovery mechanism allowing the Authentication Server to know which Authentication Handlers are supported.

The following table provides a comparison of the different Authentication Handlers:

	No Application Security	AES-CMAC	3DES-CBC	OATH-HOTP/OCRA	PKI
Requires a PKI Engine *(Crypto-processor)* on Card	NO	NO	NO	NO	YES
Deployable by SMS OTA	YES (applet size ~4Ko)	YES (applet size ~6K)	YES (applet size ~6K)	Complex (applet size ~10Ko)	Very complex (applet size >20Ko)
Off-GSM coverage mode	No	No	No	Yes (if the message needs to be displayed, then coverage needed)	No
Recognized, and valued by B2B markets *(bank, enterprise)*	Well known in Telecom market	Well known in Telecom market	Well known in Telecom market	Well known	Well known (Legally binding)
Robustness of cryptography	Depending on selected security level of the transport (may be 3DES or AES)	State of the art as symmetric cryptographic	Strong (only few years of confidence left according to several organizations)	Strong	Depending on algorithm
Card eligibility rate	Very good	Only very recent cards	Very good	Very good	Only cards with PKI Crypto-processor

Table 3 Comparison of Authentication Handlers

6.2.1 Life-cycle of the Authentication Handler

Each Authentication Handler has its own life-cycle and can be managed independently.

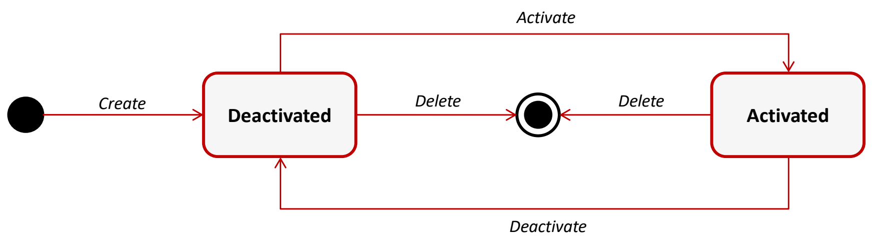


Figure 6 Authentication Handler life cycle

When created, the Authentication Handler gets in Deactivated state. The creation step requires that all necessary data required by the Authentication Handler is provided.

The Authentication Handler can be activated, deactivated or deleted at any time.

The Card Authentication Applet shall reject any authentication request using a deactivated Authentication Handler.

6.2.2 "Click OK" with authentication based on secure messaging layer

This Authentication Handler relies only on the security applied on the response message, at secure messaging level, to provide the authentication challenge response.

This Authentication handler doesn't require any specific data to be provided at creation time to be operational.

6.2.3 "Click OK" with authentication based on 3DES-CBC

This Authentication Handler shall compute the authentication challenge response using a 3DES-CBC algorithm.

This Authentication handler requires the following specific additional data to be provided by the Authentication Server at initialization time to be operational:

- Authentication key on 112 bits (for two keys mode) or 168 bits (for three keys mode).

The process of 3DES-CBC signature generation is provided in section Annex B.

6.2.4 "Click OK" with authentication based on AES-CMAC

This Authentication Handler shall compute the authentication challenge response using a AES-CBC algorithm.

This Authentication handler requires the following specific additional data to be provided by the Authentication Server at initialization time to be operational:

- Authentication key on 128 bits (as defined in RFC 4493 [25])

The process of AES-CMAC signature generation is provided in section Annex C.

6.2.5 "Click OK" with authentication based on OATH OCRA

This Authentication Handler shall compute the authentication challenge response accordingly to OATH OCRA defined in RFC 6287 [2].

This Authentication handler requires the following specific additional data to be provided by the Authentication Server at initialization time to be operational:

- Authentication key on 160 bits
- Counter

The Authentication key corresponds to the Key (K) designated in RFC 4226 [1] and in RFC 6287 [2]. This value can be changed by the Authentication Server at any time and whatever the applet or Authentication Handler status (activated or deactivated).

The counter corresponds to the Counter (C) designated in RFC 4226 [1] and in RFC 6287 [2], and contains an unsigned 8-byte value. If the counter is not set, the applet shall supply a default value of '0x0000000000000000'.

6.2.6 "Personal Code input" with authentication based on secure messaging layer

This Authentication Handler relies only on the security applied on the response message, at secure messaging level, to provide the authentication challenge response.

In addition, the Applet requires the verification of the Personal Code entered by the end-user.

This Authentication handler doesn't require any specific data to be provided at initialization time to be operational.

6.2.7 "Personal Code input" with authentication based on 3DES-CBC

This Authentication handler is similar to the "Click OK" with authentication based on 3DES except that it requires the Personal Code to be verified by the end-user before sending the authentication challenge response.

6.2.8 "Personal Code input" with authentication based on AES-CMAC

This Authentication handler is similar to the "Click OK" with authentication based on AES except that it requires the Personal Code to be verified by the end-user before sending the authentication challenge response.

6.2.9 "Personal Code input" with authentication based on OATH OCRA

This Authentication handler is similar to the "Click OK" with authentication based on OATH OCRA except that it requires the Personal Code to be verified by the end-user before sending the authentication challenge response.

6.3 Applet deployment and card architecture options

There are a number of ways in which the card could be used for providing a simple authentication capability:

- An applet as a standalone application and providing a simple text-based user interface
- An applet as a SIM browser plug-in; the applet delegates the GUI management to the SIM browser (e.g. a S@T browser or a WIB browser)
- An applet as a standalone application but delegating the GUI management to a native application on the device.

This document focuses on the an applet as a standalone application and providing a simple text-based user interface

Two options will be explored: deployed directly under the ISD or in a dedicated SSD. This is applicable to UICC platform and influences the interface that the MSSP has to use to target the applet.

Note: This section focuses on transport level of the messages and the consequences on card side and backend side. Having a secured transport mechanism is mandatory to carry the authentication messages between Card Authentication Application and MSSP. Then on top of this secured transport, Authentication Handler may add additional security like having a signature computed with specific credentials (e.g. Authentication Handler based on 3DES-CBC or Authentication Handler based on OATH OCRA).

Option 3 is a future option and will need 2 additional APDUs to be defined for the interaction between the device app and the SIM applet.

6.3.1 Secure messaging relying on OTA platform

This section lists the deployment models where the secure messaging is handled by the OTA Platform; the Authentication Server (MSSP) relying on this security layer to communicate with the Applet.

6.3.1.1 Applet deployed in ISD using OTA keys

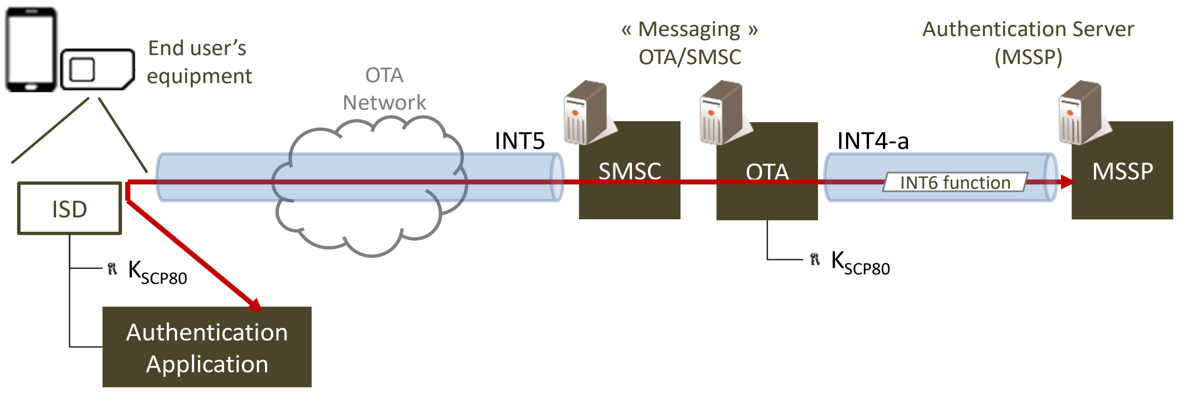


Figure 7: Applet Deployment in ISD / using OTA keys

In this UICC architecture, secure messaging with the Card Authentication Application relies on the OTA keys owned by the MNO and secure messaging over SMS as discussed in the section 4. The applet doesn't need to provide any additional transport security. This approach can be emulated by SIM card even if notion of Security Domain doesn't exist.

The Authentication Server (MSSP) needs to be connected to the MNO OTA to access a simple messaging service to send one of the functions of the INT6 interface and receive corresponding response.

The main limitation of this architecture is that the Card Authentication Application cannot have access in an interoperable way to the ISD SCP80 keys to send a secure message (note that it doesn't concern a response message that can still be secured), so that it can limit the authentication scenarios.

Advantages of this solution:
+ Rely on standard secure messaging, limiting the size of the messages
+ Simple Card architecture
+ Avoid additional code in applet to manage applicative security

Drawbacks of this solution:
- Need to have a trusted relationship between OTA owner and Authentication Server MSSP (No end-to-end communication between Applet and MSSP)
- No possibility for the applet to send a secure message (but secure response is still possible)

6.3.1.2 Applet deployed in ISD using OTA keys and https

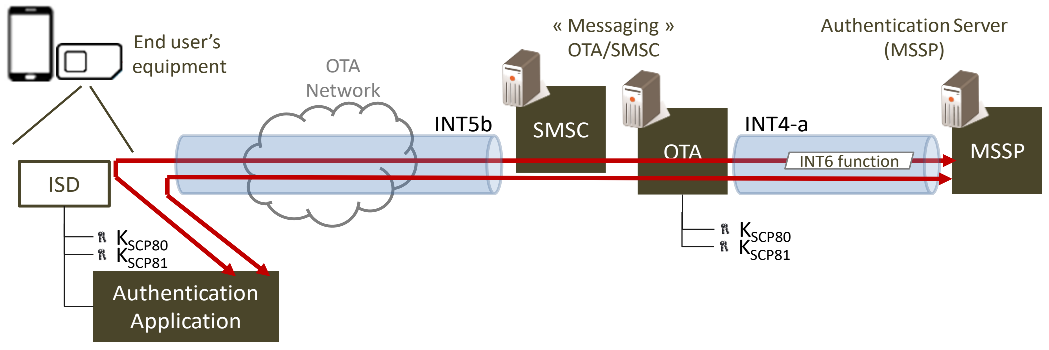


Figure 8 : Applet deployment in USD / using OTA keys and https

In this UICC architecture, secure messaging with the Card Authentication Application relies on the OTA keys owned by the MNO and secure messaging over SMS as discussed in the section 4 to establish a https connection (INT5b) or re-use the secure messaging over SMS (INT5) as described in previous section 6.3.1.1. The applet doesn't need to provide any additional transport security.

The Authentication Server (MSSP) needs to be connected to the MNO OTA to access a simple messaging service and http to send one of the functions of the INT6 interface and receive corresponding response by one or the other way.

Advantages of this solution:

+ Relies on standard secure messaging - SMS and https.

+ Simple Card architecture

+ Solves some issues linked to SMS MO data (e.g. when roaming this would be forbidden by the network, SMSC management)

Drawbacks of this solution:

- Need to have a trusted relationship between the OTA platform owner and the Authentication Server (MSSP) - (No end-to-end communication between Applet and MSSP)

- Additional code in the applet is necessary for https management

- An https connection is not efficient for this amount of data (SMS MT with connection parameters + time to establish the https session)

6.3.2 Secure end-to-end messaging (MSSP to Applet)

This section lists the deployment models that allow a secure end-to-end messaging between the Authentication Server (MSSP) and the Applet. Nevertheless, note that in case the Applet is deployed OTA, the credentials (allowing a secure end-to-end communication between the MSSP and the Applet) have to be loaded on the Card in a confidential way regarding the OTA Platform.

Such a mechanism is out of scope of this specification that assumes that there is a trust relationship between the OTA Platform and the MSSP (both entities being owned by the same MNO).

For information, such a mechanism is defined (for UICC) by GlobalPlatform Confidential Card Content Management Card Specification v2.2 - Amendment A.

6.3.2.1 Applet deployed in a dedicated SSD using OTA keys

This deployment has the Card Authentication Application deployed under a dedicated Supplementary Security Domain (SISD) with its own OTA keys.

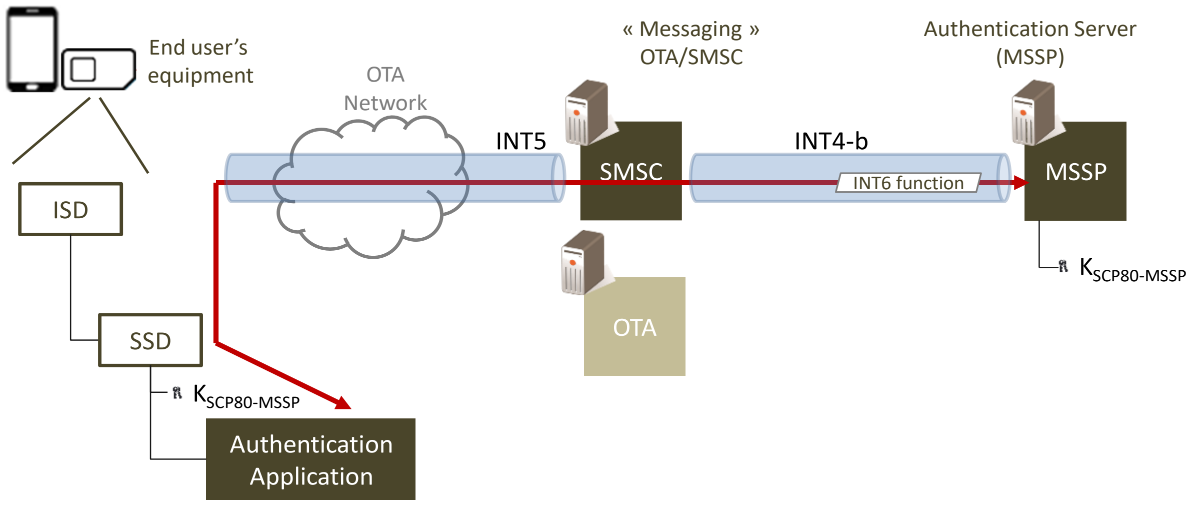


Figure 9 : Applet deployment in dedicated SSD / using OTA keys

Compared to the previous architecture the SCP80 keys are under a dedicated SSD owned by the Authentication Server (MSSP). The Card Authentication Application doesn't need to provide any additional transport security. This approach can be emulated by SIM card even if notion of Security Domain doesn't exist.

The Authentication Server (MSSP) shall have its own OTA capabilities (mainly the capability to generate secure SCP80 message on its own), and simply connected to a simple SMSC for SMS sending.

In that case, the applet issuance phase shall also include the creation of the SSD and the personalization of the SSD with OTA Keys $K_{SCP80-MSSP}$. For this last step, GlobalPlatfom defines some key establishment procedures that would allow the Authentication Server to confidentially set-up the $K_{SCP80-MSSP}$ into the SSD. But these procedures are out of scope of this document as it assumes that both Authentication Server and OTA are handled by the same MNO entity; and it would also complicate the interfaces between these roles.

As for the previous architecture the Card Authentication Application cannot have access in an interoperable way to send a secure message with Keys $K_{SCP80-MSSP}$.

Advantages of this solution:
+ Rely on standard secure messaging, limiting the size of the messages
+ Avoid additional code in applet to manage applicative security
+ No dependency on MNO OTA, simple SMSC connection required
+ End-to-end secure channel

Drawbacks of this solution:
- More complex Card architecture, Card has to support several SD.
- No possibility for the applet to send a secure message (but secure response are still possible)
- Complicates MSSP that needs to handle secure messaging on its own

6.3.2.2 Applet deployed in a dedicated SSD using OTA keys and https

This deployment has the Card Authentication Application deployed under a dedicated Supplementary Security Domain (SSD) with its own OTA keys.

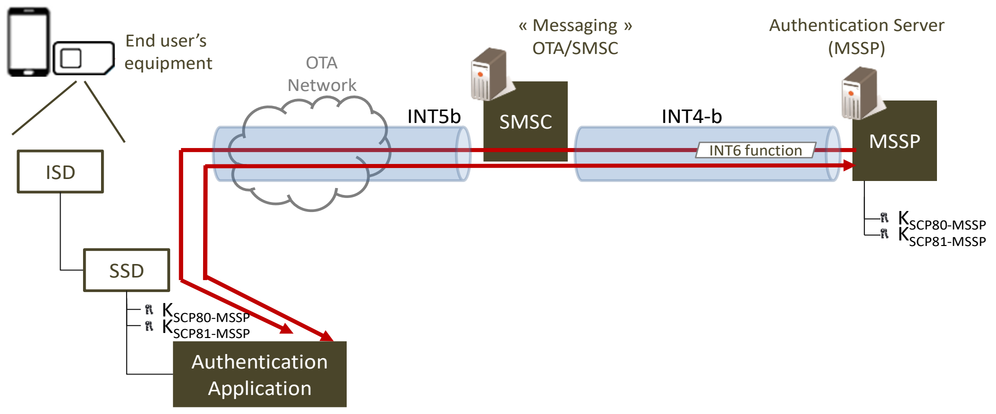


Figure 10: Applet deployment in SSD / using OTA keys and https

Compared to the previous architecture the SCP80 & SCP81 keys are under a dedicated SSD owned by the Authentication Server (MSSP). The Card Authentication Application doesn't need to provide any additional transport security. This approach can be emulated by SIM card even if notion of Security Domain doesn't exist.

The Authentication Server (MSSP) shall have its own OTA capabilities (mainly the capability to generate secure SCP80 and SCP81 messages on its own), and simply connected to a simple SMSC for SMS sending and a https web server.

In that case, the applet issuance phase shall also include the creation of the SSD and the personalization of the SSD with OTA Keys $K_{SCP80-MSSP}$. For this last step, GlobalPlatfom defines key establishment procedures that would allow the Authentication Server to confidentially set-up the $K_{SCP80-MSSP}$ and $K_{SCP81-MSSP}$ into the SSD. But these procedures are out of scope of this document as it assumes that both Authentication Server and OTA are handled by the same MNO entity; and it would also complicate the interfaces between these roles.

As for the previous architecture the Card Authentication Application can have access in an interoperable way to send a secure message with Keys $K_{SCP80-MSSP}$ and $K_{SCP81-MSSP}$.

Advantages of this solution:
+ Rely on standard secure messaging SMS and http
+ End-to-end secure channel

Drawbacks of this solution:
- More complex Card architecture, Card has to support several SD.
- No possibility for the applet to send a secure message (but secure response are still possible)
- Additional code in applet is necessary for https management
- Complicates MSSP that needs to handle secure messaging on its own
- https connexion is not efficient for this amount of data (SMS MT with connection parameters + time to establish the https session)
- Dependency on MNO OTA with a simple SMSC connection required but a new Https connection required

6.3.2.3 Applet deployed in a dedicated SSD with applicative SCP02/SCP03

This deployment has the Card Authentication Application deployed under a dedicated Supplementary Security Domain (SSD) with applicative keys SCP02 as defined in GlobalPlatform Card Specifications [7] or SCP03 as defined in GlobalPlatform Amend Secure Channel Protocol 03 [9].

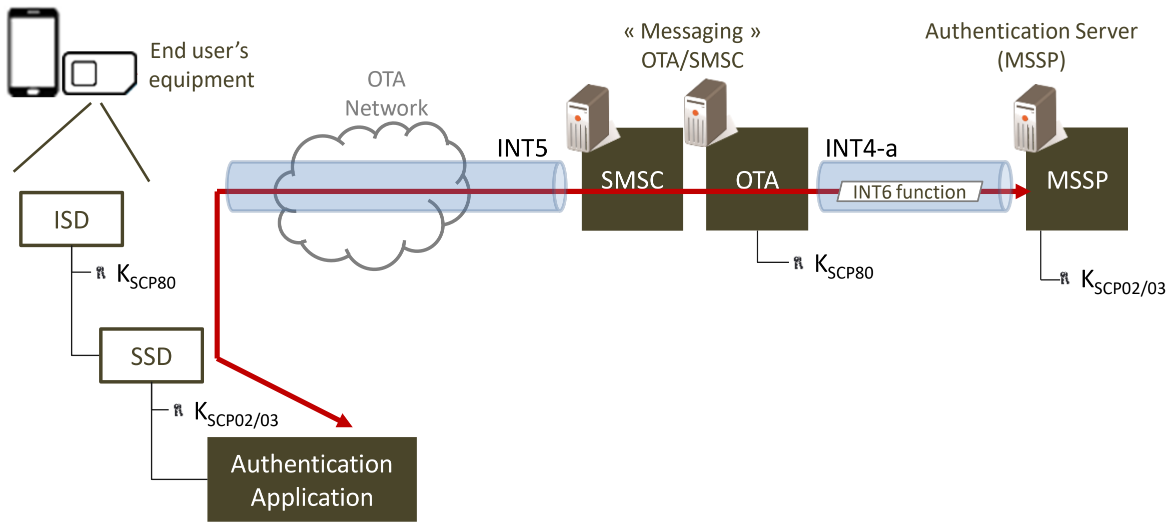


Figure 11 : Applet deployment in dedicated SSD / with SCP02/SCP03

In this UICC architecture, secure messaging with the Card Authentication Application relies both on the OTA keys owned by the MNO and secure messaging over SMS as discussed in the section 4, and the additional security layer SCP02 or SCP03. The applet doesn't need to provide any additional transport security. This approach is clearly not applicable to SIM Card. (TBC this statement)

The Authentication Server (MSSP) needs to be connected to the MNO OTA to access a simple messaging service to send one of the functions of the INT6 interface and receive corresponding response.

The Card Authentication Application issuance phase shall also include the creation of the SSD and the personalization of the SSD with Keys $K_{SCP02/03}$. As for the previous case, the $K_{SCP02/03}$ can be confidentially set-up by the Authentication Server using the same GlobalPlatform procedures.

Advantages of this solution:
+ Rely on standard secure messaging 3GPP TS 31.115 [10], and applicative SCP02/03
+ Avoid additional code in applet to manage applicative security
+ End-to-end secure channel

Drawbacks of this solution:
- Less available space for application payload in message due to security overhead
- More complex Card architecture, Card has to support several SD.
- Not applicable to SIM. (TBC)
- No possibility for the applet to send a secure message (but secure response are still possible)
- Complicates MSSP that needs to handle SCP02/03

6.3.2.4 Applet deployed in a dedicated SSD with applicative SCP02/SCP03 and SMS/https

This deployment has the Card Authentication Application deployed under a dedicated Supplementary Security Domain (SSD) with applicative keys SCP02 as defined in GlobalPlatform Card Specifications [7] or SCP03 as defined in GlobalPlatform Amend Secure Channel Protocol 03 [9].

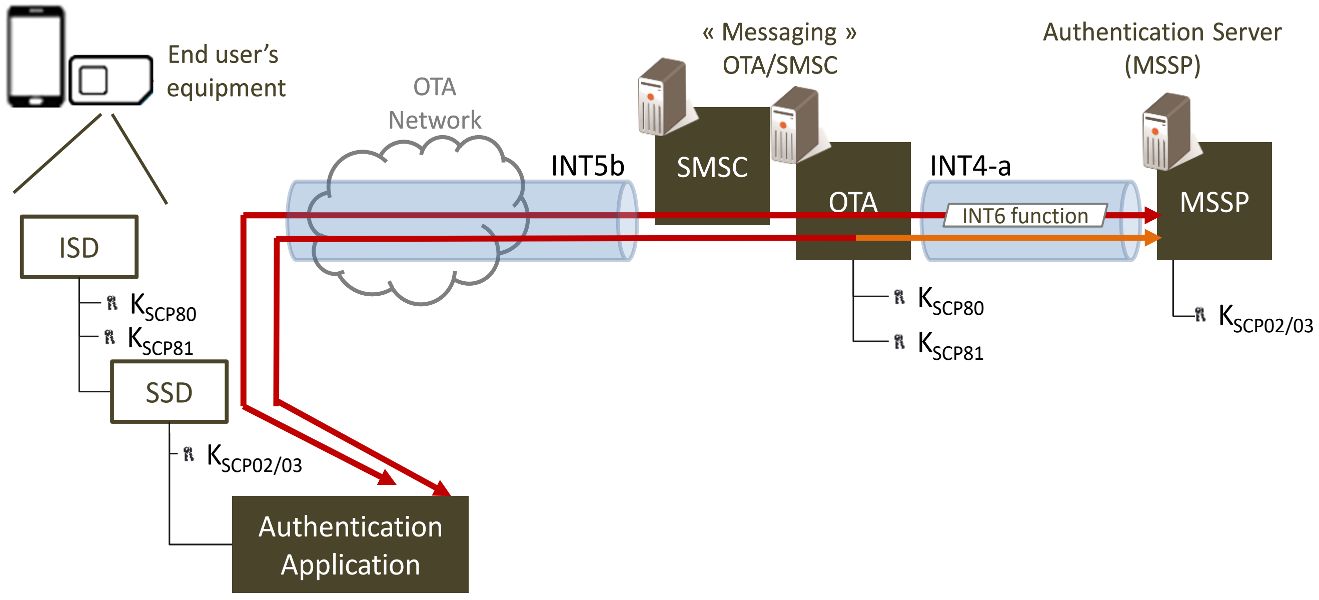


Figure 12: Applet deployment in SSD / with SCP02/SCP03 and SMS/https

In this UICC architecture, secure messaging with the Card Authentication Application relies both on the OTA SMS or HTTPS keys owned by the MNO and secure messaging over HTTPS or SMS as discussed in the section 4, and the additional security layer SCP02 or SCP03. The applet doesn't need to provide any additional transport security.

The Authentication Server (MSSP) needs to be connected to the MNO OTA to access a simple messaging service (SMS) to send one of the functions of the INT6 interface and receive corresponding response via https secured with HTTPS.

The Card Authentication Application issuance phase shall also include the creation of the SSD and the personalization of the SSD with Keys $K_{SCP02/03}$. As for the previous case, the $K_{SCP02/03}$ can be confidentially set-up by the Authentication Server using the same GlobalPlatform procedures.

Advantages of this solution:
+ Rely on standard secure messaging 3GPP TS 31.115 [10], Global Platform Card Amendment D [9] and applicative SCP02/03
+ Solve some issues link to data SMS MO (in roaming, forbidden by the network, SMSC management) + End-to-end secure channel
+ Compatible with SE and SIM environment

Drawbacks of this solution:
- Less available space for application payload in message due to security overhead
- More complex Card architecture, Card has to support several SD.
- Additional code in applet is necessary for https management
- No possibility for the applet to send a secure message (but secure responses are still possible)
- Complicates MSSP that needs to handle SCP02/03

- https connection is not efficient for this amount of data (SMS MT with connection parameters + time to establish the https session)
- Dependency on MNO OTA with a simple SMSC connection required but a new Https connection required

6.3.2.5 Applet deployed in ISD using applicative security

This deployment has the Card Authentication Application deployed under the ISD, applet having its own key to assume secure messaging.

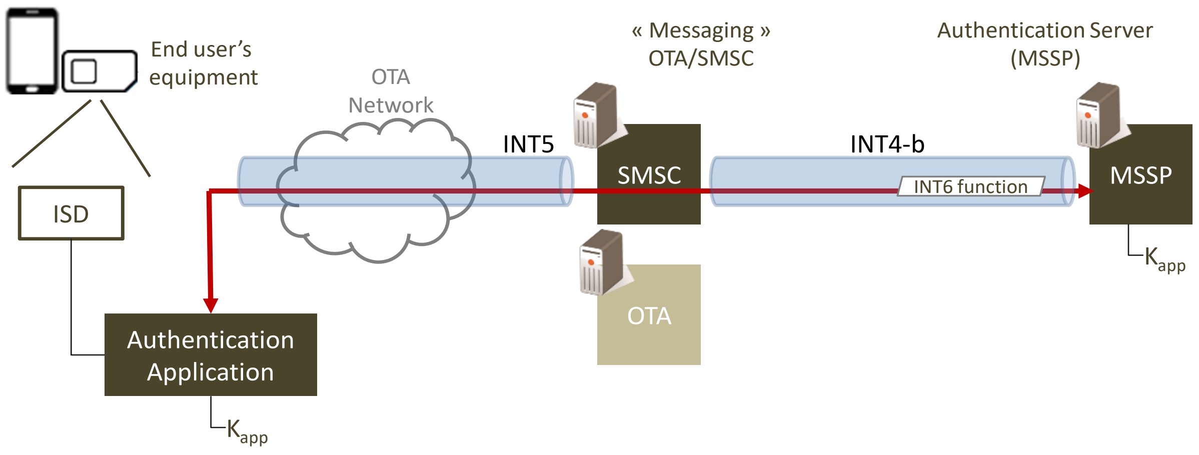


Figure 11 : Applet deployment in ISD using applicative security

In this UICC architecture, Authentication Server (MSSP) needs to be simply connected to the SMSC for sending of the secured message. This approach can be emulated by SIM card even if notion of Security Domain doesn't exist.

The Secure message is still a SMS containing a Command Packet structure as defined in 3GPP TS 31.115 [10], but indicating no security (SPI=00 00) (so not involving OTA keys of the ISD) to take benefit of the addressing capabilities of the protocol. Then the Command Packet contains the applicative payload being secured by an applicative protocol on top.

During the Card Authentication Application issuance, the Authentication Server (MSSP) is fully in charge of personalizing the applet with the necessary applicative keys.

Advantages of this solution:
+ Simple Card architecture
+ Possibility for the applet to send a secure message
+ No dependency on MNO OTA, Simple SMSC connection required
+ End-to-end communication

Drawbacks of this solution:
- Increase size of the applet due to additional secure messaging management
- Increase size of the message (additional layer required in the message structure)
- Complicates MSSP that needs to handle additional security transport

6.3.3 Recommendation

The table below recaps the main differentiation points of the various deployment modes (or equivalent one in the case of the SIM):

Data name	‘Applet deployed in ISD using OTA keys’	‘Applet deployed in a dedicated SSD using OTA keys‘	‘Applet deployed in a dedicated SSD with applicative SCP02’	‘Applet deployed in ISD using applicative security’
Impact of the size of Applet	1	1	1	4
Available room for customer text display	1	1	2	4
Card eligibility rate	1	2	3	1
Allows End-to-end transport security	2	1	1	1
Potential impact on Card cost	1	2	3	1

Table 4 : Differences between various deployment modes

Each solution is ranked with value ‘1’, ’2’, ’3’, ’4’ compared to the others solution.

‘1’ is the best solution, where ‘4’ is the worst solution in term of potential impacts.

This specification recommends using one of the two models, in the preferred order:

1) ‘Applet deployed in ISD using OTA keys’. The main reasons for this choice is that it relies on a simple Card architecture that suits most of the existing Cards, minimize the size of the Applet and maximize the size available in the message for applicative payload (e.g. size of the text to be displayed for authentication request).

2) ‘Applet deployed in a dedicated SSD using OTA keys‘. The main reasons for this choice are that it allows a secure end-to-end messaging (allowing the MSSP to verify all the parts of the message) and gets MSSP independent from the OTA Platform.

The model ‘Applet deployed in a dedicated SSD with applicative SCP02’ complexifies the deployment solution (provisioning of the different parties, MSSP to handle SCP02/03...) and requires Card features that will drastically limit the possibility to leverage on the already deployed Cards (and also increase cost of the required Card). This model will not allow to be deployed on all cards.

The model ‘Applet deployed in ISD using applicative security’ has the major drawback to increase the size of the Applet (byte code to handle secure messaging) and reducing the size available in the message for applicative payload (e.g. size of the text to be displayed for authentication request). But, for those scenarios in which the implementation of this E2E transport key at application level is necessary, an optional functionality is specified in order to allow the management of these keys.

6.4 Applet lifecycle management

As part of the Mobile Connect registration process, an Operator employing the Card Authentication Application mechanism will also need to check whether or not the user’s card either has the Card Authentication Application already installed, or whether the card is suitable for the applet to be pushed OTA to the card.

If the Card Authentication Application is already present, it can be invoked OTA (pop-up on the phone) and the user requested to setup a Personal Code (by entering it twice) – see Personal Code Creation Journey. If the applet is not present but an OTA upgrade is possible, the user receives a message that their Operator is able to upgrade him soon, in the following hours/days/weeks depending on the Operator OTA capabilities and chosen approach (e.g., on-demand or batch). Alternatively, if the card needs to be changed, depending on the Operator policy the user will be proposed a card swap (either in-store or via post).

The easiest approach is for the Operator to request their card vendor(s) to integrate the Card Authentication Application in the Operator's card. Personalization of the applet may then be done in pre-issuance or OTA using an MSSP API.

However, given the time it will take for all users to replace their cards to those pre-embedded with the Card Authentication Application, Operators should also consider provisioning the Card Authentication Application OTA to cards already in-use.

If deploying to existing cards, there will need to be an eligibility check to ensure the card is capable of supporting the Card Authentication Application (e.g., card type and space memory capacity). Either the MSSP or the Identity GW will therefore need to be able to check Card eligibility and order a Card Authentication Application OTA download when needed.

"Eligibility check": the service must handle administrative commands in order to check the status of the Applet local installation. This includes:

- Applet loaded on the card or not
- Applet installed on the card or not
- Applet initialised on the card or not (Specific applet command(s))
- If Applet is not present, available memory check

"Applet OTA download/install": the service must handle administrative commands in order to order the Applet OTA download and install.

- The command should return a status: OTA planned, Applet downloaded, Applet installed, Applet download (e.g. GlobalPlatform command INSTALL_FOR_LOAD)
- Installation (e.g. GlobalPlatform command INSTALL_FOR_INSTALL)

Suspension of the applet must also be supported (either managed by the Authentication Server or the Identity GW)

Note: the method of performing the background check to determine whether the user's card already has the Applet or to check whether the card is capable of receiving the applet OTA will be added to the document in a later version based on feedback/guidelines from Operators and SIM vendors.

6.5 Multi Language support

To provide the best user experience, the Card Authentication Applet shall allow the MSSP to configure the language of the different textual messages that are displayed during the various end-user journeys. This only concerns static messages handled by the Applet alone and not the messages sent by the MSSP within the authentication request as this message can already be sent in the appropriate language.

The Applet configuration with a specific language shall be done at loading and installation step by the MSSP based on the received information (see "Mobile Registration" function at section 9.1.6). This specification doesn't mandate a specific mean to fulfil this feature. It may be for instance a dedicated Applet for each language (not recommended), or a unique Applet package configured with a separate resource bundle (recommended) loaded and installed separately (but still under the control of the MSSP). Another future option is to have separate command to switch the language. Other means could be envisaged.

In order to implement the end-user journey described in section 3 and provide a similar end-user experience across all the Card Authentication Application implementations, it is recommended that the Card Authentication Application handles the following list of local messages:

Message Id	Message	Description

1.PC_n	Please, enter your Personal code ('n' digits)	Used during 'Personal code journey', § 3.2 Used during 'Personal code creation journey', §3.3 Used during 'Reset/Change Personal Code Journey', § 3.5
2.PC_CONFIRM	Please, confirm Personal Code	Used during 'Personal code creation journey', §3.3 Used during 'Reset/Change Personal Code Journey', § 3.5
3.PC_INIT_OK	New Personal Code validated	Used during 'Personal code creation journey', §3.3 Used during 'Reset/Change Personal Code Journey', § 3.5
4.PC_INIT_KO	Not the same Personal Code value, try again	Used during 'Personal code creation journey', §3.3 Used during 'Reset/Change Personal Code Journey', § 3.5
5.PC_OK	Personal code is valid	Used during 'Personal code journey', § 3.2
6.PC_KO_m	Personal code is not valid, 'm' remaining attempt(s)	Used during 'Personal code journey', § 3.2
7.PC_BLK	Personal code is blocked	Used during 'Personal code journey', § 3.2
8. PC_NOT_CREATED	Personal code has not been created	Used during 'Personal code creation journey', §3.3

Table 5 : Multi language support

Messages are provide in English for example, and have to be changed according to the end-user language preferences.

6.6 Applet data

The Card Authentication Application, in addition to any data specific to implementation, or data specific to a particular Authentication Handler shall handle the data described in this section. Table 6 provides the list of data. Refer to next sections for the description of each data.

Data name	Known by user	Known by MSSP	Default value
Supported Authentication Handlers types	-	X	-
GSMA Version	-	X	-
Activation flag	-	X	Deactivated
Installation date	-	X	-
Max PC attempt	X	X	3
Personal Code (PC)	X	-	-
PC Length	X	X	4
Time out	X	X	30
E2E transport key flag		X	Deactivated
E2E transport key type			(Opt)
MSSP TP-DA		X	(Opt)

Table 6 Applet data

6.6.1 Supported Authentication Handler types

The “Supported Authentication Handlers type” indicates the list of Authentication Handler types that can be created on-board on the Applet (note that it doesn’t mean they are ready to be used for an authentication request, but ready to be created according to Section 6.2)

This value is intrinsic to the applet and cannot be changed.

Data structure: 2 bytes

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
-	-	-	-	-	-	-	0	No support of “Click OK” with authentication based on secure messaging layer
-	-	-	-	-	-	-	1	Support of “Click OK” with authentication based on secure messaging layer
-	-	-	-	-	-	0	-	No support of “Personal Code input” with authentication based on secure messaging layer
-	-	-	-	-	-	1	-	Support of “Personal Code input” with authentication based on secure messaging layer
-	-	-	-	-	0	-	-	No support of “Click OK” with authentication based on 3DES-CBC
-	-	-	-	-	1	-	-	Support of “Click OK” with authentication based on 3DES-CBC
-	-	-	-	0	-	-	-	No support of “Personal Code input” with authentication based on 3DES-CBC
-	-	-	-	1	-	-	-	Support of “Personal Code input” with authentication based on 3DES-CBC
-	-	-	0	-	-	-	-	No support of “Click OK” with authentication based on OATH OCRA
-	-	-	1	-	-	-	-	Support of “Click OK” with authentication based on OATH OCRA
-	-	0	-	-	-	-	-	No support of “Personal Code input” with authentication based on OATH OCRA
-	-	1	-	-	-	-	-	Support of “Personal Code input” with authentication based on OATH OCRA
-	0	-	-	-	-	-	-	No support of “Click OK” with authentication based on AES-CMAC
-	1	-	-	-	-	-	-	Support of “Click OK” with authentication based on AES-CMAC
0	-	-	-	-	-	-	-	No support of “Personal Code input” with authentication based on AES-CMAC
1	-	-	-	-	-	-	-	Support of “Personal Code input” with authentication based on AES-CMAC

Table 7 Authentication Handler types (Byte 1)

The Byte2 is RFU.

6.6.2 GSMA Version

This data indicates the latest version of the GSMA Card Authentication Application specification (this document) supported by the applet.

This value is intrinsic to the applet and cannot be changed.

Note: The present document is versioned on 3 digits. Only major and minor digit can bring or change the functional features. The last digit shall only include minor corrections and editorial fixes that doesn't change the functional. That's why, the only two first digits are managed at applet level.

Data structure: 2 bytes

- First byte: major version in packet BCD (e.g. '01h' to indicate major version '1')
- Second byte: minor release in packet BCD (e.g. '15h' to indicate minor release '15')

6.6.3 Activation flag

This field indicates the activation status of the applet:

- Activated: the applet can be configured and is able to provide authentication services (using an activated Authentication Handler).
- Deactivated: the applet can only be configured. Authentication services shall not be usable whatever the state of an Authentication Handler.

The applet can only be activated by the Authentication Server. The applet can be activated or deactivated at any time.

Data structure: 1 byte

Defined values:

- '00h': Deactivated
- '01h': Activated

6.6.4 Installation date

This field provides the date and time when the applet has been installed on the card.

This value can be set only once.

Data structure: 4 bytes

Defined values: Time elapsed in seconds since midnight UTC of January 1, 1970 [UT]

6.6.5 Max Personal Code attempts

This field indicates the number of times the Personal Code can be entered by the user when required to authenticate himself, with a wrong value, before being blocked. It also indicates the number of times the user may fail trying to set a new Personal Code in the operation Reset Personal Code before returning an error. Note: the consumed attempts are independent for both operations (the user may block the Personal Code when trying to authenticate himself, but this wouldn't block the Reset Personal Code operation).

Setting this parameter is optional. If the parameter is not set, the applet shall supply a default value of 3 attempts. It means in that case that the Personal Code will be blocked after 3 consecutive wrong Personal Code entered by the user.

This value shall be set prior to the setting of personal code in the applet.

Data structure: 1 byte

Defined values: '01h' to '0Fh'

6.6.6 Personal Code (PC)

This field contains the Personal Code of the user. The Personal Code is only known by the user and is stored locally in the applet. The Personal Code length shall be set accordingly to the ‘PC Length’ field.

This value shall be entered by the user each time the Personal Code is reset by the Authentication Server.

This value shall be set before the activation of the first Authentication Handler requiring a Personal Code.

6.6.7 PC Length

This field indicates the length (in digits) of the Personal Code that has to be provided by the user. If the parameter is not set, the applet shall supply a default value of 4 digits. The maximum length shall be 8 digits.

This value can be changed by the Authentication Server at any time and whatever the applet status (activated or deactivated). It shall be taken into account at the next PC reset.

Data structure: 1 byte

Defined values: ‘04h’ to ‘08h’

6.6.8 Time Out

This field must not be supported anymore. In case of receive operations to set this field as it was specified in previous versions of this document, it must be simply ignored.

Note: This field has been removed from the specification due to inconsistent and dangerous behavior of Sim Toolkit [10] implementations when a duration TLV is included in proactive commands. For example, it has been detected that an expiration event will return a RES_CMD_PERF (ok) instead of RES_CMD_PERF_NO_RESP_FROM_USER (expiration without user response). Taking into account this issue, the Click OK authenticators won’t have the expected behavior and would actually be dangerous, authenticating users without their explicit consent.

6.6.9 MSSP address

This field contains the MSSP address to be used as TP-DA value that the Card Authentication Application may use to send a binary SM MO containing the response of command(s) execution. This case is described in 8.1.1.2.2.

This data is optional and applet may not support it.

MSSP Address should be encoded as TP-DA address defined in TS 123 040.

Data structure:

- 1 Byte for Address Length - The Address-Length field is an integer representation of the number of useful semi-octets within the Address-Value field, i.e. excludes any semi-octet containing only filling bits.
- 1 Byte for TON and NPI
- 0 to 10 Bytes for Address-Value field (dialing number string) using semi-octet representation.

Defined values: Refer to ETSI TS 102 223 [4].

6.6.10 E2E transport key activation flag

This field indicates the activation status of the E2E transport key:

- Activated: the applet must use the configured key to decrypt all received commands and encrypt responses.
- Deactivated: the applet doesn't do anything regarding channel encryption.

The E2E transport key can only be activated by the Authentication Server; it can be at any time as soon as a transport key has been configured.

The E2E transport key can be deactivated by the Authentication Server at any time.

Data structure: 1 byte

Defined values:

- '00h': Deactivated
- '01h': Activated

6.6.11 E2E transport key type

In case of being configured, this field indicates which algorithm is used to encrypt the channel. The values are the ones defined in GlobalPlatform Card Specifications [7], section "11.1.8 Key Type Coding". Only Triple DES in CBC mode and AES are supported.

Data structure: 1 byte

Defined values:

- '82h': 3DES-CBC key type
- '88h': AES key type

6.6.12 Applet Release

The value reference the Applet Implementation version and is used by the MSSP for applet versioning management.

This fields is optional and the applet may not support it.

The value is intrinsic to the applet and should not be changed.

This data structure provides the MSSP application access to the following applet implementation information:

- Applet version :
  - Mobile Connect Applet ID :
    - MC_RID (Mobile Connect Registered application provider Identifier): 5 bytes hex
    - MC_PIX (Mobile Connect Provider Application Identifier): 4 bytes hex
  - Version Data structure:
    - First byte, major release in packed BCD (e.g. '01h' to indicate major version '1')

        ▪ Second byte, minor release in packed BCD (e.g. ‘15h’ to indicate minor release ‘15’)

6.6.13 Applet Locale

The value references the Applet Implementation version.

This value is intrinsic to the applet and cannot be changed.

This fields is optional and the applet may not support it.

Used by MSSP to determine ‘locale’ settings.

- Locale (for displayed message):
    - Language: ISO 639, 2bytes UTF-8
    - Script: ISO 15924, 2 bytes BCD
    - Country of use: ISO 3166 2bytes UTF-8

7 Detailed procedures

The flows described in this section focus on the interactions between ID GW, MNO, MSSP and Card Authentication Applet, and hides the other Mobile Connect aspects and components, like interactions between Service Provider and ID GW (over Open ID Connect). See [28] for further details on the Mobile Connect architecture and APIs. “MNO” in these diagrams refers to a non-specific entity provided by the MNO in order to execute management operations, depending upon implementation.

Any failure returned to the ID GW or MNO by the MSSP in these sequences is an applicative error and not a SOAP error or fault. SOAP errors are not illustrated.

Commands send from MSSP to Card Authentication Applet can be regrouped in a same SMS (provided that they fit a single SMS). Nevertheless, for purpose of clarity, the following sequences illustrate one command per SMS.

7.1 Authentication/Signature request

The following figure describes the call flow for the case where the user, navigating a Service Provider’s service, clicks on a Mobile Connect button (or something equivalent depending on the Service Provider’s service implementation) and triggers the authentication using the mobile phone of the user.

7.1.1 SMS MT & MO flowchart

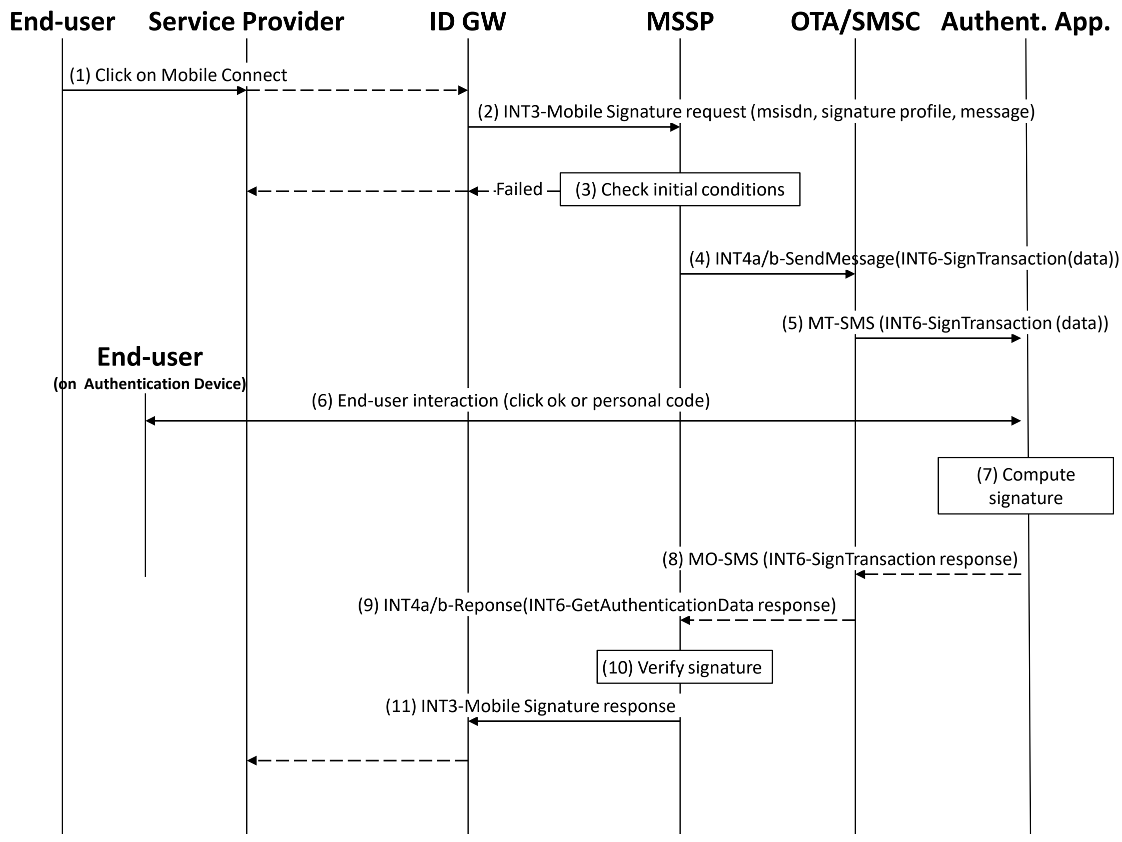

End-user
Service Provider
ID GW
MSSP
OTA/SMSC
Authent. App.
(1) Click on Mobile Connect
(2) INT3-Mobile Signature request (msisdn, signature profile, message)
Failed
(3) Check initial conditions
(4) INT4a/b-SendMessage(INT6-SignTransaction(data))
(5) MT-SMS (INT6-SignTransaction (data))
End-user
(on Authentication Device)
(6) End-user interaction (click ok or personal code)
(7) Compute signature
(8) MO-SMS (INT6-SignTransaction response)
(9) INT4a/b-Reponse(INT6-GetAuthenticationData response)
(10) Verify signature
(11) INT3-Mobile Signature response

Figure 13 Sequence flow describing authentication/signature

[Note: This flow is an indicative flow in the Applet / MSSP authentication context, and hides the other Mobile Connect aspects and components, e.g. Identity GW etc.]

(1) The user navigating a Service Provider’s service clicks on a Mobile Connect button. By a process that is out of scope of this document, the Identity GW receives the authentication request.

(2) The Identity GW calls the “**INT3a-MobileSignature**” function with its relevant input data (at least end-user identifier (e.g. MSISDN), signature profile, message…)

(3) The MSSP verifies that the Identity GW request is acceptable. The check shall include at least the following steps:

- the request well formed (all data present...)
- the end-user already registered for the requested signature profile
- the end-user activated

If any of the conditions to be verified are not satisfied, the MSSP shall return a response indicating the failure, and the procedure shall end.

(4) (5) The MSSP shall send a secure message targeting the SIM applet and containing the “**INT6-Sign transaction**” command with its relevant input data. Depending on the SIM / UICC architecture, the MSSP may either manage the secure message part or delegate it to the MNO OTA.

(6) The Card Authentication Application handles the user experience as described in user Journey 3.1 or 3.2 depending on the Level of Assurance requested by the Service Provider

(7) The Card Authentication Application generates authentication data.

(8) (9) The SIM applet returns the MO-SMS containing the execution status of the "**INT6-Sign Transaction**" command to the MSSP.

(10) The MSSP verifies the authentication data returned by the Card Authentication Application. Depending on the SIM / UICC architecture, the MSSP may have to additionally handle the secure transport layer.

(11) The MSSP returns the response of the "**INT3a-MobileSignature**" function indicating if the user has been authenticated or not. This procedure illustrates the 'Asynchronous server-server mode' defined in section 9.1.2.1.

7.1.2 SMS MT & HTTPS flowchart

The previous flow is modified below to integrate a SCP81 session between the OTA and the Authentication application.

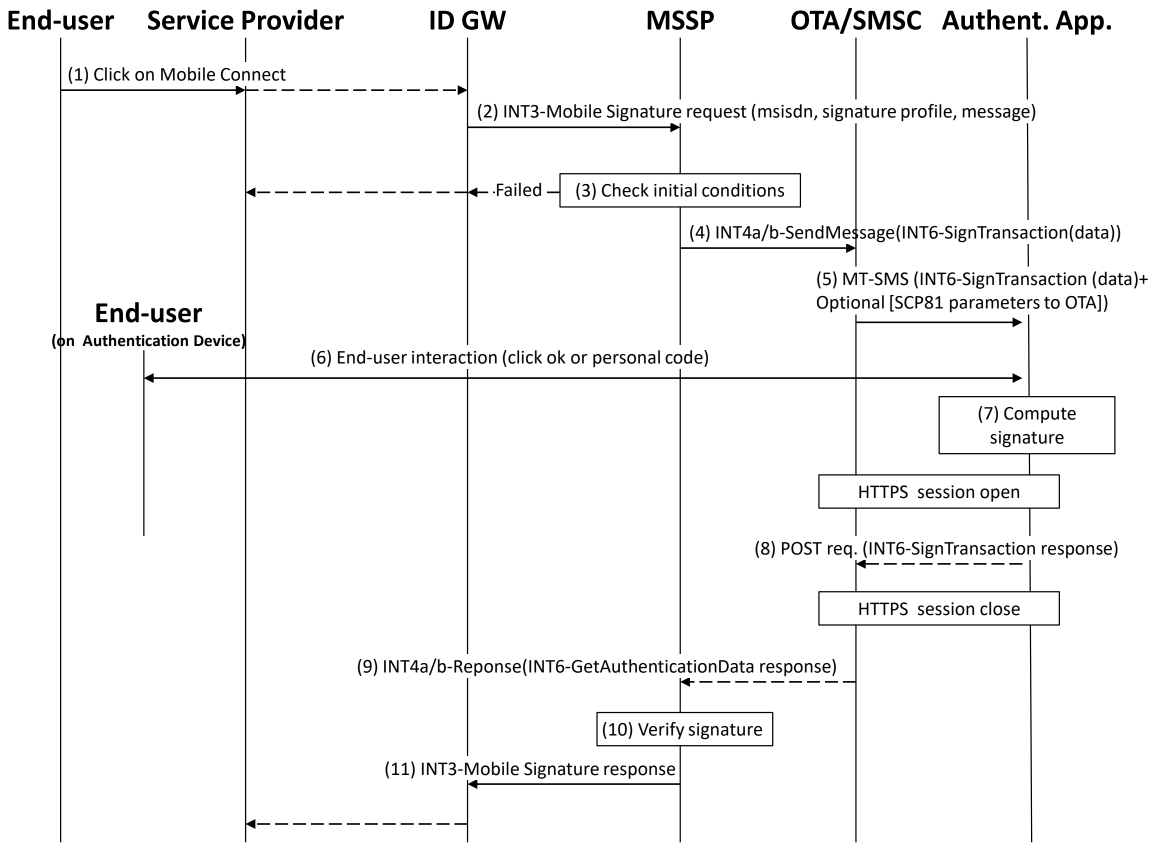

End-user
Service Provider
ID GW
MSSP
OTA/SMSC
Authent. App.
(1) Click on Mobile Connect
(2) INT3-Mobile Signature request (msisdn, signature profile, message)
Failed
(3) Check initial conditions
(4) INT4a/b-SendMessage(INT6-SignTransaction(data))
(5) MT-SMS (INT6-SignTransaction (data)+ Optional [SCP81 parameters to OTA])
End-user (on Authentication Device)
(6) End-user interaction (click ok or personal code)
(7) Compute signature
HTTPS session open
(8) POST req. (INT6-SignTransaction response)
HTTPS session close
(9) INT4a/b-Reponse(INT6-GetAuthenticationData response)
(10) Verify signature
(11) INT3-Mobile Signature response

Figure 14 Sequence flow describing authentication/signature

[Note: This flow is an indicative flow in the Applet / MSSP authentication context, and hides the other Mobile Connect aspects and components, e.g. Identity GW etc.]

(1) The user navigating a Service Provider's service clicks on a Mobile Connect button. By a process that is out of scope of this document, the Identity GW receives the authentication request.

(2) The Identity GW calls the "**INT3a-MobileSignature**" function with its relevant input data (at least end-user identifier (e.g. MSISDN), signature profile, message…)

(3) The MSSP verifies that the Identity GW request is acceptable. The check shall include at least the following steps:

- the request well formed (all data present...)
- the end-user already registered for the requested signature profile
- the end-user activated

If any of the conditions to be verified are not satisfied, the MSSP shall return a response indicating the failure, and the procedure shall end.

(4) (5) The MSSP shall send a secure message targeting the Authentication applet or the parent SD (ISD or dedicated SSD) with the "**INT6-Sign transaction**" command with its relevant input data. An optional field is available to add the HTTPs parameters as defined in Global Platform Amendment B [27] for the applet response:

- If the field is null (not present): the application is configured with a default bearer option (SMS or HTTPs) and will select this one for the response

- If the field is empty (length = 0): the connection parameters of the parent SD or the authentication application itself (TBD) will be used

- If the field is not empty: As described in GlobalPlatform Amendment B [27], all pushed connection parameters will overload the local parameters

Depending on the SIM / UICC architecture, the MSSP may either manage the secure message part or delegate it to the MNO OTA.

(6) The Card Authentication Application handles the user experience as described in user Journey 3.1 or 3.2 depending on the Level of Assurance requested by the Service Provider

(7) The Card Authentication Application generates authentication data.

(8) (9) The authentication application will initiate a HTTPs session with OTA or MSSP depending of the parameters provided in (5). The applet returns through a SCP81 POST Request containing the execution status of the "**INT6-Sign Transaction**" command to the MSSP. When the POST is received, the server will send a POST Response 204 without content to close the session.

(10) The MSSP verifies the authentication data returned by the Card Authentication Application. Depending on the SIM / UICC architecture, the MSSP may have to additionally handle the secure transport layer.

(11) The MSSP returns the response of the "**INT3a-MobileSignature**" function indicating if the user has been authenticated or not. This procedure illustrates the 'Asynchronous server-server mode' defined in section 9.1.2.1.

7.2 First registration of an end-user

The following figure describes the call flow corresponding to the registration of the end-user in the MSSP. This sequence is triggered by the MNO.

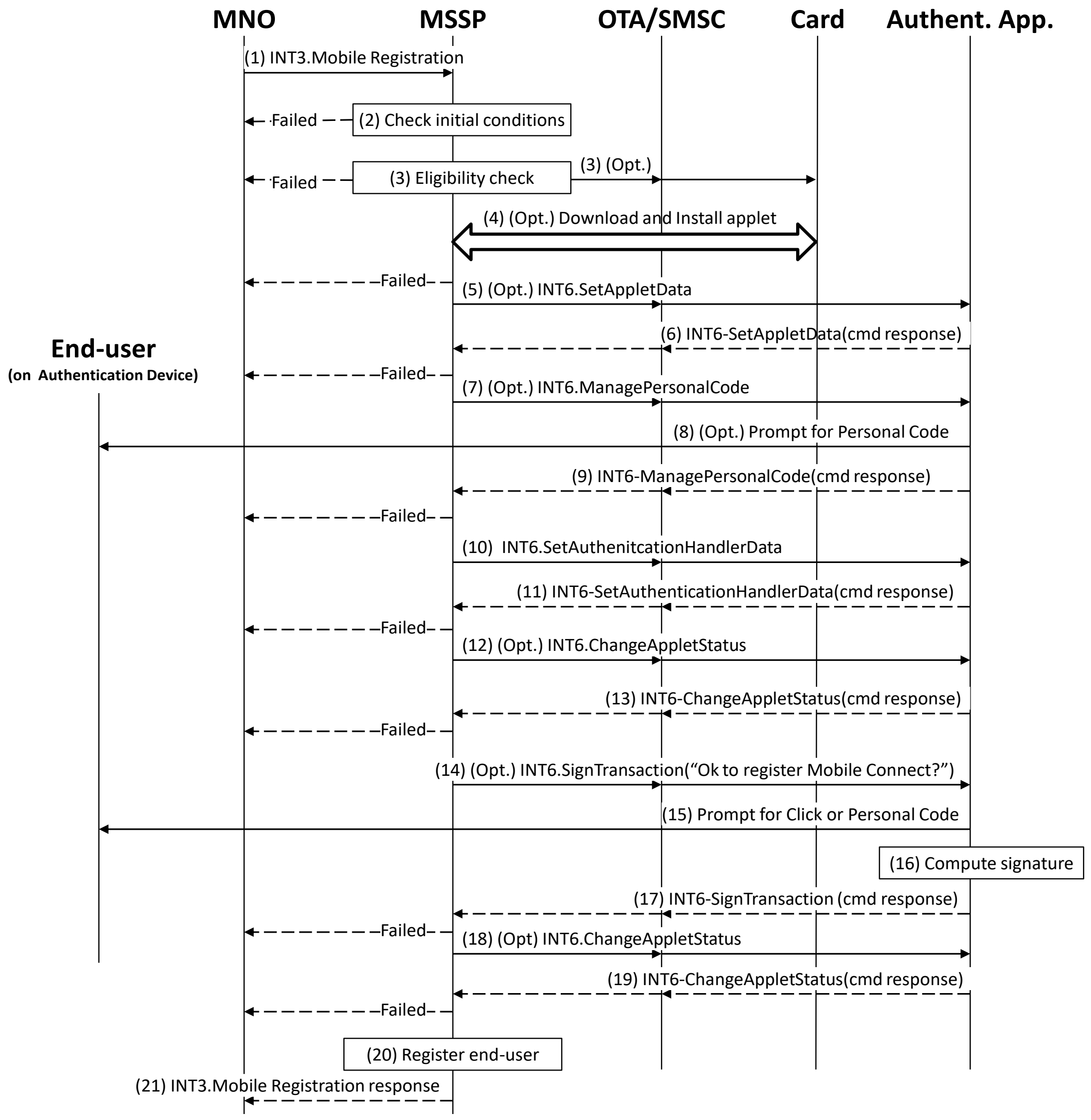

MNO
MSSP
OTA/SMSC
Card
Authent. App.
End-user
(on Authentication Device)
(1) INT3.Mobile Registration
Failed
(2) Check initial conditions
Failed
(3) Eligibility check
(3) (Opt.)
(4) (Opt.) Download and Install applet
Failed
(5) (Opt.) INT6.SetAppletData
(6) INT6-SetAppletData(cmd response)
Failed
(7) (Opt.) INT6.ManagePersonalCode
(8) (Opt.) Prompt for Personal Code
(9) INT6-ManagePersonalCode(cmd response)
Failed
(10) INT6.SetAuthenitcationHandlerData
(11) INT6-SetAuthenticationHandlerData(cmd response)
Failed
(12) (Opt.) INT6.ChangeAppletStatus
(13) INT6-ChangeAppletStatus(cmd response)
Failed
(14) (Opt.) INT6.SignTransaction("Ok to register Mobile Connect?")
(15) Prompt for Click or Personal Code
(16) Compute signature
(17) INT6-SignTransaction (cmd response)
Failed
(18) (Opt) INT6.ChangeAppletStatus
(19) INT6-ChangeAppletStatus(cmd response)
Failed
(20) Register end-user
(21) INT3.Mobile Registration response

Figure 15 Sequence flow describing registration

Note: to simplify the figure, the INT4a/b and INT5/INT5b allowing communication between MSSP and Card Authentication Application are not illustrated. But this would be identical to what is presented in section 7.1.

Pre-conditions:

- End-user is not already registered in the MSSP (first registration)
- End-user belongs to the MNO
- End-user has already been authenticated by the MNO (by another mechanism)

(1) The MNO calls the “**INT3b-MobileRegistration**” function with its relevant input data (at least end-user identifiers (can be many, but at least MSISDN is mandatory), the requested signature profile…)

(2) The MSSP shall check that the end-user can be registered. The check shall include at least the following steps:

- The request well formed (all data present...)
- the end-user already registered for the requested signature profile

If any of the conditions to be verified are not satisfied, the MSSP shall return a response indicating the failure, and the procedure shall end.

(3) If the registration request is acceptable, the MSSP shall check the eligibility of the end-user. The MSSP shall verify the end-user’s mobile equipment status:

- Is the Card Authentication Applet already loaded and installed on the Card? If yes, the MSSP shall go to the step (5).
- Is the Card Authentication Applet can be loaded and installed on the card? This may be done using simply the card content representation of the OTA Platform (if available), or by interrogating OTA the Card.

If Card Authentication Applet cannot be loaded and installed, MSSP shall return a response indicating the failure, and the procedure shall end.

(4) Optionally the MSSP may trigger the load and installation of the Applet. This step is performed using OTA Platform functions and is out of scope of this specification. In case of error during this step, the MSSP shall return a response indicating the failure, and the procedure shall end.

(5) Optionally the MSSP may send the “**INT6-SetAppletData**” function to modify the Applet configuration.

(6) The Applet shall return the response of the “**INT6-SetAppletData**” function to the MSSP. In case of error during this step, the MSSP may return a response indicating the failure, and the procedure may end.

(7) Optionally the MSSP may send the “**INT6-ManagePersonalCode**” function to the Applet to initialize the Personal Code on the Applet (in that case the steps (8) and (9) are executed, else the procedure goes to step (10)).

(8) The Applet shall request the end-user to enter its Personal Code according to the end-user journey described in section 3.3.

(9) The Applet shall return the response of the “**INT6-ManagePersonalCode**” function to the MSSP. In case of error during this step, the MSSP shall return a response indicating the failure, and the procedure shall end.

(10) The MSSP shall send the “**INT6-SetAuthenticationHandlerData**” function to the Applet to install the new Authentication Handler corresponding to the requested signature profile in step (1).

(11) The Applet shall return the response of the “**INT6-SetAuthenticationHandlerData**” function to the MSSP. In case of error during this step, the MSSP shall return a response indicating the failure, and the procedure shall end.

(12) Optionally the MSSP shall send the “**INT6-ChangeAppletStatus**” function to the Applet to activate the newly installed Authentication Handler and optionally the applet.

(13) The Applet shall return the response of the “**INT6-ChangeAppletStatus**” function to the MSSP. In case of error during this step, the MSSP shall return a response indicating the failure, and the procedure shall end.

(14) Optionally the MSSP shall send the “**INT6-SignTransaction**” function to the Applet, requesting the end-user to confirm its registration to Mobile Connect

(15) The Applet shall request the end-user to either accept or cancel according to “Click-OK journey” section 3.1, or to confirm by entering its Personal Code according to “Personal Code journey” described in section 3.2.

(16) The Applet shall compute the signature response with the newly installed Authentication Handler.

(17) The Applet shall return the response of the “**INT6-SignTransaction**” function to the MSSP. In case of error during this step, the MSSP shall return a response indicating the failure, and the procedure shall end.

(18) If service validation fails, the MSSP shall send the “**INT6-ChangeAppletStatus**” function to the Applet to deactivate the Authentication Handler and the Applet (depending on the status of activated Authentication Handlers).

(19) The Applet shall return the response of the “**INT6-ChangeAppletStatus**” function to the MSSP. In case of error during this step, the MSSP shall return a response indicating the failure, and the procedure shall end.

(20) The MSSP shall update its local database to reflect the newly registered end-user.

(21) The MSSP shall return the response of the “**INT3b-MobileRegistration**” function indicating that the end-user has been registered.

Note: If at one point there is an error or if the user abort the process, then the applet should be reset to initial status. An Authentication Handler aborted setting should be deleted, with associated applicative keys and if no more Authentication Handler are present, then the Applet data parameters should be also be deleted including the Personal Code.

7.3 Un-registering an end-user (unsubscribe)

The following figure describes the call flow corresponding to the un-registration of the end-user in the MSSP. This sequence is triggered by the MNO.

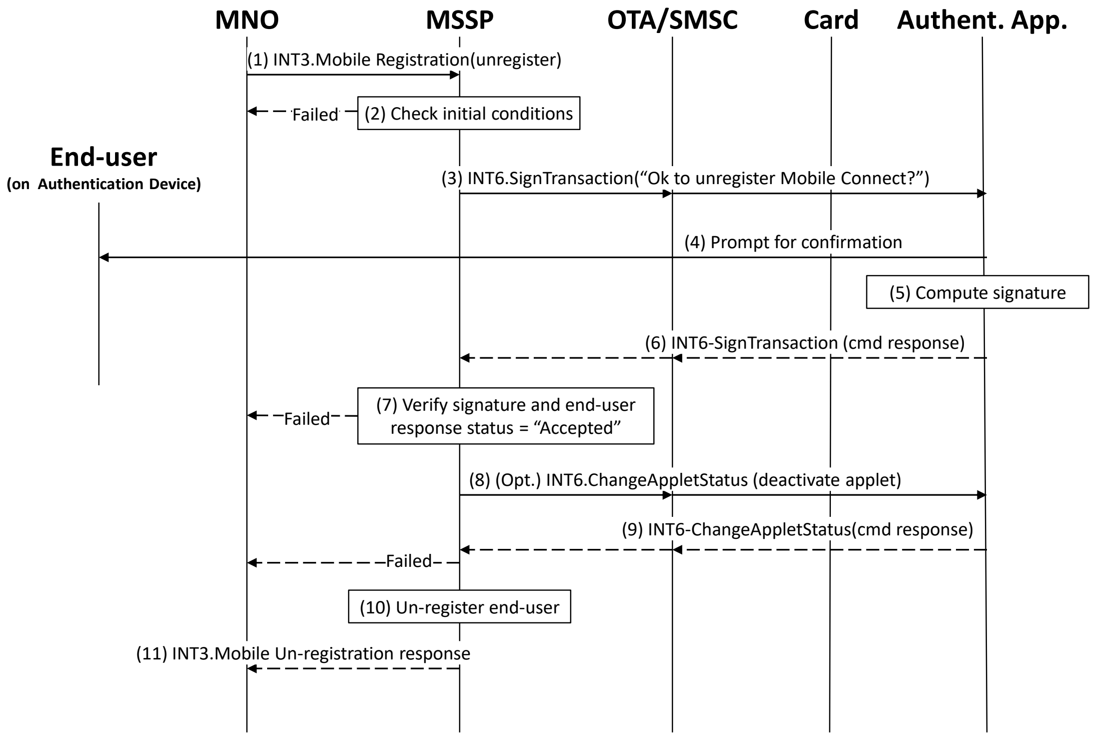

MNO
MSSP
OTA/SMSC
Card
Authent. App.
End-user
(on Authentication Device)
(1) INT3.Mobile Registration(unregister)
Failed
(2) Check initial conditions
(3) INT6.SignTransaction("Ok to unregister Mobile Connect?")
(4) Prompt for confirmation
(5) Compute signature
(6) INT6-SignTransaction (cmd response)
Failed
(7) Verify signature and end-user response status = "Accepted"
(8) (Opt.) INT6.ChangeAppletStatus (deactivate applet)
(9) INT6-ChangeAppletStatus(cmd response)
Failed
(10) Un-register end-user
(11) INT3.Mobile Un-registration response

Figure 16 Sequence flow describing Un-registration

Note: to simplify the figure, the INT4a/b and INT5/INT5b allowing communication between MSSP and Card Authentication Application are not illustrated. But this would be identical to what is presented in section 7.1.

Pre-conditions:

- End-user is already registered in the MSSP

(1) The MNO calls the "**INT3b-MobileRegistration**" function in the "Unregister" mode, with its relevant input data.

(2) The MSSP shall verify that the end-user is registered for the "Personal Code" authentication mode and that end-user is registered and is activated for Mobile Connect. If any of the conditions to be verified are not satisfied, the MSSP shall return a response indicating the failure, and the procedure shall end.

(3) The MSSP shall send the "**INT6-SignTransaction**" function to the Applet, requesting the end-user to confirm its un-registration to Mobile Connect

(4) The Applet shall request the end-user to either accept or cancel according to "Click-OK journey" section 3.1, or to confirm by entering its Personal Code according to "Personal Code journey" described in section 3.2.

(5) The Applet shall compute the signature response with the targeted Authentication Handler.

(6) The Applet shall return the response of the "**INT6-SignTransaction**" function to the MSSP. In case of error during this step, the MSSP shall return a response indicating the failure, and the procedure shall end.

(7) The MSSP shall verify the signature of the end-user's Applet is valid and that end-user has accepted the un-registration request. If any of the conditions to be verified are not satisfied, the MSSP shall return a response indicating the failure, and the procedure shall end.

(8) Optionally the MSSP may send a may send the "**INT6-ChangeAppletStatus**" function to the Applet to deactivate the Applet.

(9) The Applet shall return the response of the "**INT6-ChangeAppletStatus**" function to the MSSP. In case of error during this step, the MSSP shall return a response indicating the failure, and the procedure shall end.

(10) The MSSP shall update its local database to reflect the un-registration end-user.

(11) The MSSP shall return the response of the "**INT3b-MobileRegistration**" function indicating that the end-user has been registered.

Note: optionally the MSSP at step (8) may also delete the Applet on the end-user's mobile equipment to free space.

7.4 Changing card

The following figure describes the call flow corresponding to the change of a card requesting the end-user to be removed from the MSSP and registered again. This sequence is triggered by the Identity GW or the MNO's provisioning system (The figure only shows the ID GW).

This goal of this procedure is to get the end-user in the state than the one he was before card change:

- MSSP may have to load and install the Applet on the new card.
- MSSP shall re-install the Authentication Handler(s) corresponding to the signature profile(s) he was previously equipped with. The figure hereunder illustrates the case where the end-user was registered with the only LoA2 (Click-OK). It may be possible that MSSP has to re-install the LoA3; in that case the sequence would be added additional steps corresponding to the installation of the additional Authentication Handler and entering of the Personal Code.

All these steps (except optional Personal Code entering) shall be done silently to the end-user.

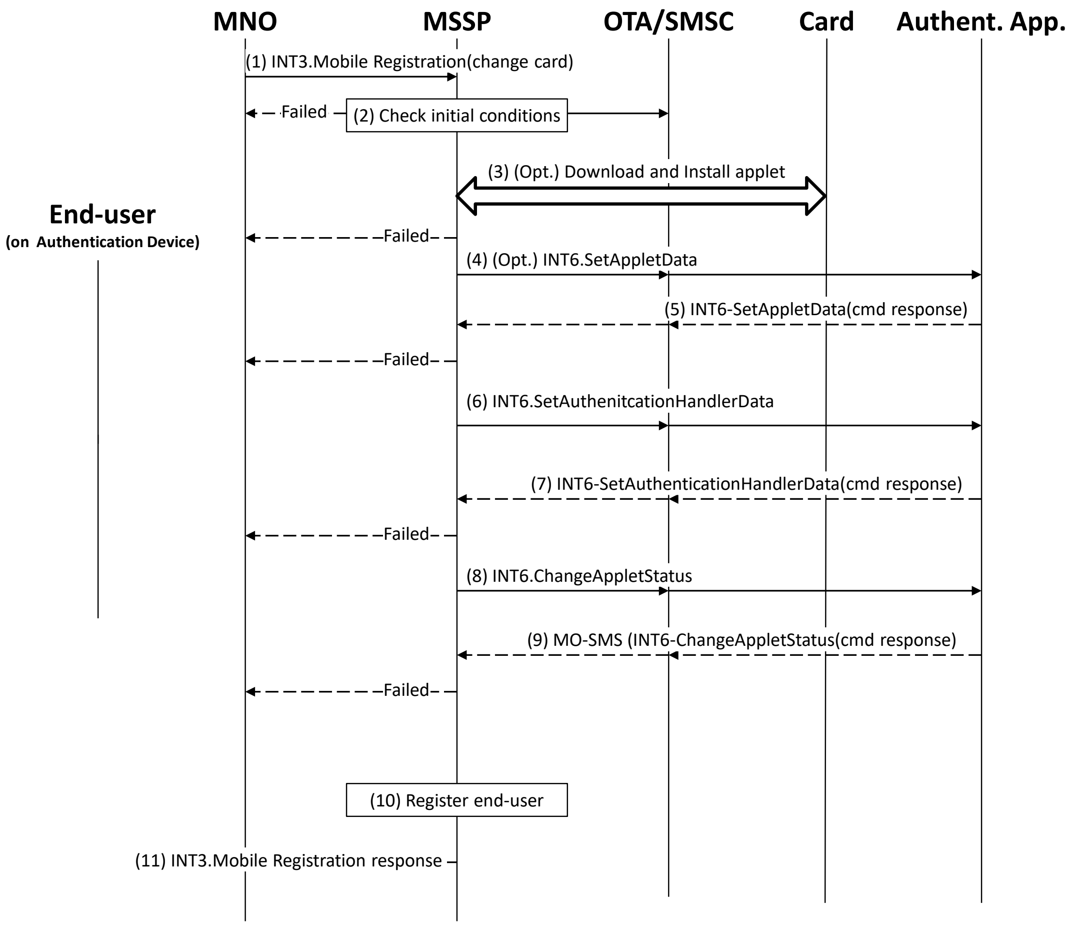

MNO
MSSP
OTA/SMSC
Card
Authent. App.
End-user
(on Authentication Device)
(1) INT3.Mobile Registration(change card)
Failed
(2) Check initial conditions
(3) (Opt.) Download and Install applet
Failed
(4) (Opt.) INT6.SetAppletData
(5) INT6-SetAppletData(cmd response)
Failed
(6) INT6.SetAuthenitcationHandlerData
(7) INT6-SetAuthenticationHandlerData(cmd response)
Failed
(8) INT6.ChangeAppletStatus
(9) MO-SMS (INT6-ChangeAppletStatus(cmd response)
Failed
(10) Register end-user
(11) INT3.Mobile Registration response

Figure 17 Sequence flow describing the change card

Note: to simplify the figure, the INT4a/b and INT5/INT5b allowing communication between MSSP and Card Authentication Application are not illustrated. But this would be identical to what is presented in section 7.1.

Note: It is assumed that the end-user keeps all its identities (including MSISDN) when the card is changed.

Pre-conditions:

- End-user is already registered in the MSSP
- The end-user's new Card is already provisioned within the OTA Platform and associated with the same (or new MSISDN)
- The end-user's new Card is able to host the Card Authentication Application

(1) The MNO calls the "**INT3b-MobileRegistration**" function in the "Change card" mode, with its relevant input data (the end-user mobile identity...).

(2) The MSSP shall check that the function request is acceptable. The check shall include at least the following steps:

- The request well formed (all data present...)
- the end-user is registered
- the end-user's new card (identified by the MSISDN) is provisioned in the OTA Platform

If any of the conditions to be verified are not satisfied, the MSSP shall return a response indicating the failure, and the procedure shall end.

(3) Optionally (depending if the new Card has already the Applet loaded and installed or not) the MSSP may trigger the load and installation of the Applet. This step is performed using OTA Platform functions and is out of scope of this specification. In case of error during this step, the MSSP shall return a response indicating the failure, and the procedure shall end.

(4) Optionally the MSSP may send the “**INT6-SetAppletData**” function to modify the Applet configuration.

(5) The Applet shall return the response of the “**INT6-SetAppletData**” function to the MSSP. In case of error during this step, the MSSP may return a response indicating the failure, and the procedure may end.

(6) The MSSP shall send the “**INT6-SetAuthenticationHandlerData**” function to the Applet to install the new Authentication Handler.

(7) The Applet shall return the response of the “**INT6-SetAuthenticationHandlerData**” function to the MSSP. In case of error during this step, the MSSP shall return a response indicating the failure, and the procedure shall end.

(8)The MSSP shall send the “**INT6-ChangeAppletStatus**” function to the Applet to activate both Applet and the newly installed Authentication Handler.

(9) The Applet shall return the response of the “**INT6-ChangeAppletStatus**” function to the MSSP. In case of error during this step, the MSSP shall return a response indicating the failure, and the procedure shall end.

(10) The MSSP shall update its local database to reflect the newly registered end-user.

(11) The MSSP shall return the response of the “**INT3b-MobileRegistration**” function indicating that the end-user has been registered.

7.5 Reset Personal Code flow

The following figure describes the call flow corresponding to the end-user Personal Code reset. This sequence is triggered by the MNO's provisioning system.

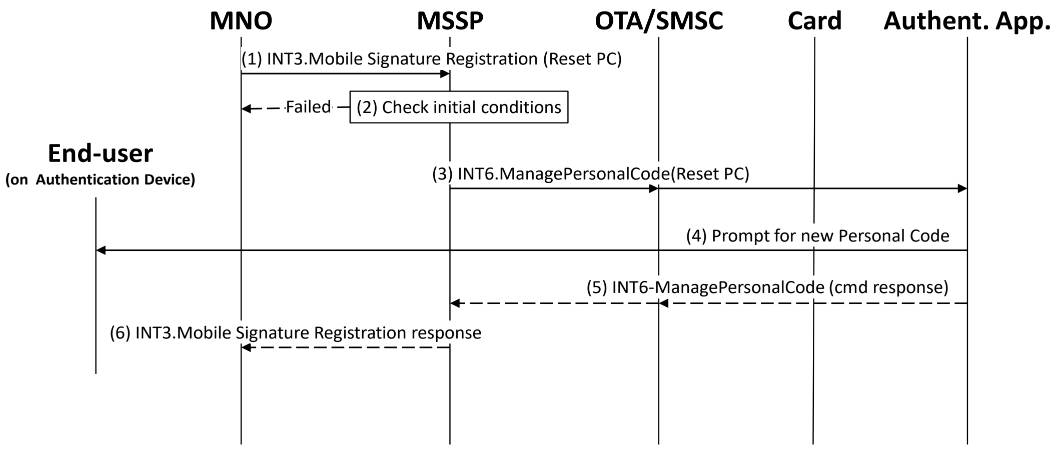

MNO
MSSP
OTA/SMSC
Card
Authent. App.
(1) INT3.Mobile Signature Registration (Reset PC)
Failed
(2) Check initial conditions
End-user
(on Authentication Device)
(3) INT6.ManagePersonalCode(Reset PC)
(4) Prompt for new Personal Code
(5) INT6-ManagePersonalCode (cmd response)
(6) INT3.Mobile Signature Registration response

Figure 18 Sequence flow describing the reset Personal Code

Note: to simplify the figure, the INT4a/b and INT5/INT5b allowing communication between MSSP and Card Authentication Application are not illustrated. But this would be identical to what is presented in section 7.1.

Pre-conditions:

- End-user is already registered in the MSSP
- End-user is registered for the "Personal Code" authentication mode
- End-user has been authenticated by the MNO by a mean out of scope of this document (see section 3.5).

(1) The MNO calls the "**INT3b-MobileSignatureRegistration**" function in the "Reset PC" mode, with its relevant input data.

(2) The MSSP shall verify that the end-user is registered for the "Personal Code" authentication mode and that end-user is activated for Mobile Connect. If any of the conditions to be verified are not satisfied, the MSSP shall return a response indicating the failure, and the procedure shall end.

(3) The MSSP shall send the "**INT6-ManagePersonalCode**" function to the Applet to reset the Personal Code stored on the Applet.

(4) The Applet shall request the end-user to enter its new Personal Code according to the end-user journey described in section 3.5.

(5) The Applet shall return the response of the "**INT6-ManagePersonalCode**" function to the MSSP.

(6) The MSSP shall return the response of the "**INT3b-MobileSignatureRegistration**" function to the MNO.

7.6 Unblock Personal Code flow

The following figure describes the call flow corresponding to the end-user Personal Code unblock. This sequence is triggered by the Identity GW or the MNO's provisioning system (only ID GW is illustrated in the figure).

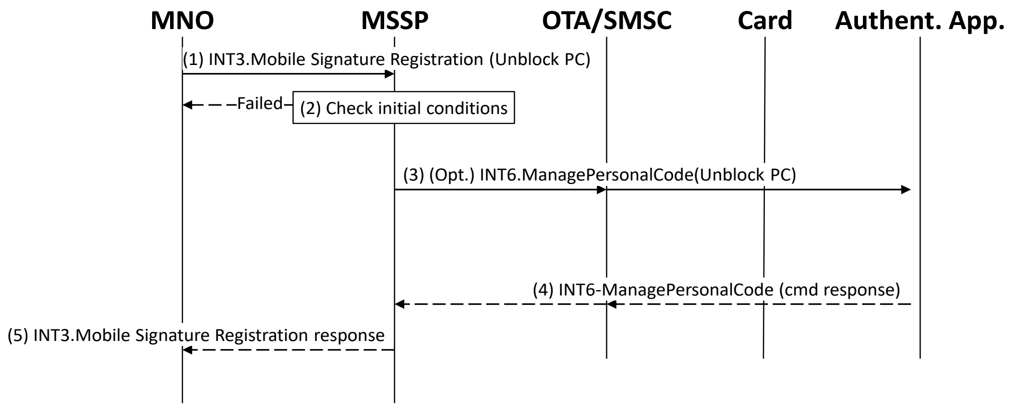

MNO
MSSP
OTA/SMSC
Card
Authent. App.
(1) INT3.Mobile Signature Registration (Unblock PC)
Failed
(2) Check initial conditions
(3) (Opt.) INT6.ManagePersonalCode(Unblock PC)
(4) INT6-ManagePersonalCode (cmd response)
(5) INT3.Mobile Signature Registration response

Figure 19 Sequence flow describing the unblock Personal Code

Note: to simplify the figure, the INT4a/b and INT5/INT5b allowing communication between MSSP and Card Authentication Application are not illustrated. But this would be identical to what is presented in section 7.1.

Pre-conditions:

- End-user is already registered in the MSSP
- End-user is registered for the "Personal Coode" authentication mode
- End-user has been authenticated by the MNO by a mean out of scope of this document (see section 3.5).

(1) The MNO calls the "**INT3b-MobileSignatureRegistration**" function in the "Unblock PC" mode, with its relevant input data.

(2) The MSSP shall verify that the end-user is registered for the "Personal Code" authentication mode and that end-user is activated for Mobile Connect. If any of the conditions to be verified are not satisfied, the MSSP shall return a response indicating the failure, and the procedure shall end.

(3) The MSSP shall send the "**INT6-ManagePersonalCode**" function to the Applet to unblock the Personal Code stored on the Applet. The applet can display a message to the user via DISPLAY TEXT.

(4) The Applet shall return the response of the "**INT6-ManagePersonalCode**" function to the MSSP.

(5) The MSSP shall return the response of the "**INT3b-MobileSignatureRegistration**" function to the MNO`.

8 Card Interfaces specifications

This section of the document describes the interfaces involving both Card and Card Authentication Application. This includes:

- INT5: Interface between the OTA platform and card
- INT6: Interface between the Authentication Server and Card Authentication Application

INT7 is out of scope of this specification. Nevertheless for the purpose of clarity, expectations for this interface are described see section 0.

8.1 INT5 and INT5b: OTA platform - Card

In the context of this specification the INT5 and INT5b interfaces allow the transport of functions defined in INT6.

This interface is defined between the ISD, or a dedicated SSD, in the UICC and the OTA Platform of the MNO owning the UICC (see also section 6.3). The same interface may also apply between a SIM card and the OTA Platform of the MNO. In that case the concept or Security Domain is not applicable but secure messaging over SMS remains, as defined in GSM 03.48[15] or can be use secure messaging over SMS to open a HTTPS session as described in GlobalPlatform Amendment B [27].

The INT6 functions shall be addressed to the SIM/UICC through a secure channel, SCP80 over SMS SMS or SCP81 over HTTPs established between the OTA Platform and the ISD.

To enable SCP80 (also called 03.48) or SCP81 (also called RAMoHTTP), the ISD, or a dedicated SSD, shall be personalized before issuance by the Card Manufacturer (in accordance with MNO specifications) with at least one key set, with a Key Version Number between ‘01’ to ‘0F’ for SCP80 following GlobalPlatform Card Specification UICC Configuration [8] or a Key Version Number between ’40’ to ‘4F’ for SCP81. We assume in this specification that these key sets are then loaded in the MNO OTA Platform by a mean that is out of scope of this specification.

The Card Authentication Application shall have a TAR to be addressable using secure messaging 03.48. The TAR can be assigned during installation phase (card manufacturing or OTA installation), and this specification doesn’t mandate a well-known fixed value. Nevertheless, the TAR has to be communicated to the Authentication Server.

8.1.1 SMS

The UICC shall support the sending of secure packets over SMS as defined in 3GPP TS 31.115 [10].

For a SIM card, the GSM 03.48 [15] shall be followed.

To avoid attack on premium number, it is recommended that UICC follows 3GPP TS 31.115 [10] v11.0.0 onwards. This latest evolution of the standards specifies that the card shall not answer to any incoming SM-MT that doesn’t include a valid signature.

The UICC shall support RAM over SMS as defined in 3GPP TS 31.116 [11] (and more generally in ETSI TS 102 226 [6]). This is required for some functions of the interface INT5 and the secure personalization of Card Authentication Application during enrolment procedure.

For a SIM card, these functions shall be realized using functions/mechanism out of scope of this specification.

According to ETSI TS 143.019 [26] (from v5.6.3 at least), section § 6.6 "Handler availability" indicates that "*The EnvelopeResponseHandler content must be posted before the first invocation of a ProactiveHandler.send method or before the termination of the processToolkit, so that the GSM Applet can offer these data to the ME (eg 9Fxx/9Exx/91xx). After the first invocation of the ProactiveHandler.send method the EnvelopeResponseHandler is no more available*".

Concretely, it means that it is impossible for the card to return a SCP80 PoR containing the response of a Command Packet depending of the processing of a Proactive session and secured with the keys of its related Security Domain using the standard *EnvelopeResponseHandler*.

Section 8.1.1.2.1 and 8.1.1.2.2 propose solutions to work around this issue. Solutions rely on the fact that the response of commands that require proactive session either doesn't include sensitive data (MANAGE_PC or PUT_DATA) or is already secured (SIGN_TRANSACTION). These solutions don't work for the Authentication Handlers based on secure messaging layer.

At the price of interoperability loss, the Card Authentication Application may still rely on proprietary API, if available, to return a standard PoR using the keys of its related Security Domain.

8.1.1.1 SMS exchange with SCP80 for commands not requiring proactive session

This case doesn't have any issue; the OTA Platform can send a SM-MT with a SCP80 envelop, applying security and requiring the response as a PoR as defined in section 8.1.2; SCP80 secured data containing one (or possibly more) of the following commands that doesn't makes use of proactive session during execution:

- Get applet data (GET_DATA)
- Set Applet data (PUT_DATA)
- Manage Personal Code (MANAGE_PC): Unblock PC mode
- Set Authentication Handler data (PUT_DATA)
- Change applet status (CHANGE_STATUS)
- Delete Authentication Handler (DELETE)

In response, the card applet may return a Proof of Receipt (PoR), secured with OTA keys of its related Security Domain following security level required in the SPI2 of the received Command Packet, and containing the command execution response (status code and additional data).

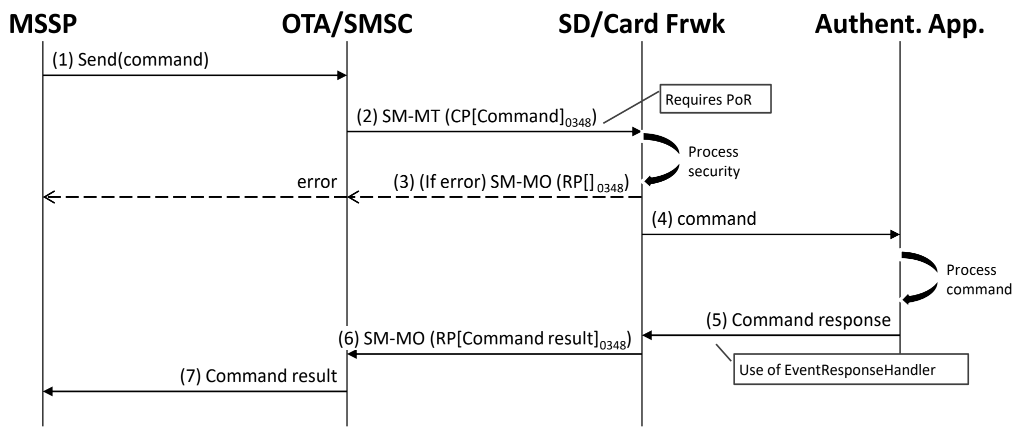


Figure 20 SMS exchange with SCP80 for commands w/o proactive session

The sequence here above represents a deployment where the Authentication Server (MSSP) relies on an OTA platform for the secure messaging. SMS exchanges would be the same for a deployment where the Authentication Server (MSSP) would embed the secure messaging.

8.1.1.2 SMS exchange with commands requiring proactive session

In case the MSSP sends a message containing command requiring the use of a proactive session, the card applet cannot send back the command execution response using the opened *EnvelopeResponseHandler()*; as a consequence we cannot have the same SMS exchange than the one described in section 8.1.1.1 .

To work around this issue (while minimizing the Applet size and without compromising Applet interoperability) the card Authentication applet shall return the command execution result response in:

- Either a binary SM-MO containing a SCP80 Command Packet with no security (SPI1=00h and SPI2=00h) and secured data field containing simply the applicative status word(s) and additional data(s) (if any); this SM-MT shall be addressed with a TP-DA value matching the TP-OA of the SM-MT it is the response to (matching the OTA Platform).

- Or a binary SM-MO containing directly the applicative status word(s) and additional data(s) (if any); this SM-MT shall be sent addressed directly to the MSSP with the TP-DA value configured in the Applet (see section 6.6.9).

Card Authentication shall implement at least one of the two methods. In case both methods are supported, the SCP80 Command Packet is the default mechanism and the Simple SM MO binary will be used in case the MSSP address is configured.

Authentication Server (MSSP) shall support both methods.

In both cases the structure of the response shall follow the section 8.2.4.

Within SIM Applet specification, the commands requiring proactive session are:

- SIGN TRANSACTION: this command returns response data that are all signed at application level by the ‘Message Authentication Code’ itself.

- Manage Personal Code (MANAGE_PC: Reset Personal Code mode): no sensitive data is returned

8.1.1.2.1 SMS exchange with response in a SCP80 Command Packet

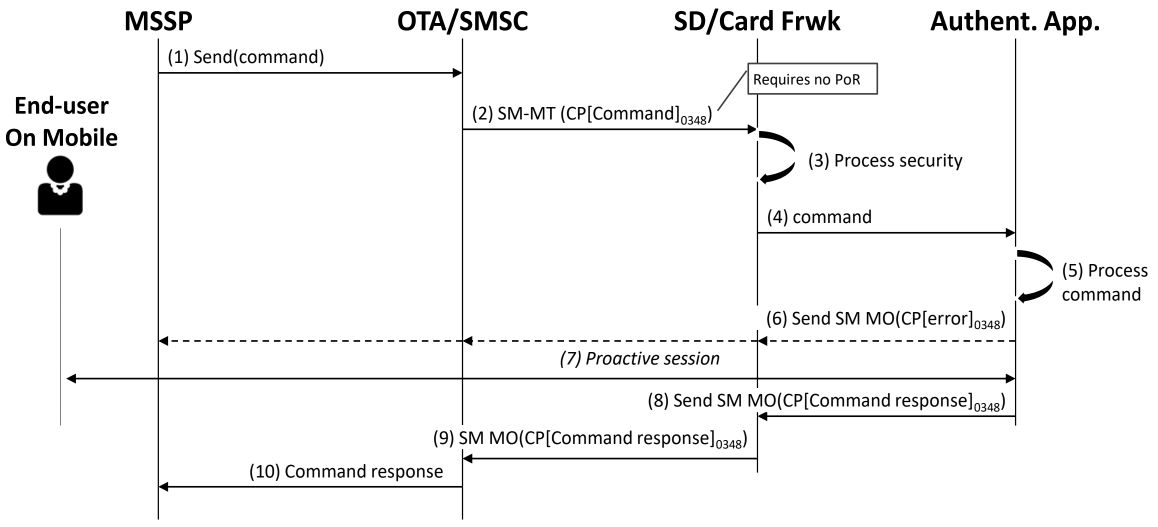


Figure 21 SMS exchange with response in a SCP80 Command Packet

The sequence here above represents a deployment where the Authentication Server (MSSP) relies on an OTA platform for the secure messaging. SMS exchanges would be the same for a deployment where the Authentication Server (MSSP) would embed the secure messaging.

(1) (2) The Authentication Server (MSSP) shall send a secure message targeting the Card Authentication Application, and containing a secured data packet with one (or more) command. The required security level applied to the SCP Command packet shall follow the section 8.1.1.1 except that it shall request **no** PoR (SPI2=00h).

(3) The Card framework/Security Domain shall process the security of the received SCP80 Command Packet.

Note that in case of security problem (counter, bad signature...) the server won't receive any response from the card. This may be seen as problematic from server side, but:

- Such an error shall not happen on a system in production, or at least should be very rare
- This minimize the number of SMS, minimizing the deployment cost and network usage
- And following latest security measures to avoid attacks on Premium numbers (see section 8.1.1), even if PoR is requested, card is not supposed to return error message when incoming message signature is invalid. So server cannot be ensured to receive an error message in all situation;

Nevertheless during integration/testing, or for a specific customer care action requiring some investigation, it may be desirable to request the PoR. If so, the further step (6) shall be a Response Packet (PoR) instead of a Command Packet and shall contain either a Status Word indicating an error (with possibly additional data) or a Status Word indicating '61 00' that the final command answer will be sent in another SM MO (step (8)). The PoR shall be secured as described in section 8.1.2.

System shall also allow such configuration individually for each message.

(4) Command(s) shall be forwarded to the Card Authentication Application.

(5) Card Authentication Application shall process the receives command(s)

(6) In case of application error during the processing of the command the Card Authentication Application shall send back a SCP80 Command Packet with no security (SPI1=00h and SPI2=00h) and secured data field containing the error (see 8.1.1.2).

(7) Card Authentication Application shall handle the proactive session required for execution of the received command(s)

(8) Card Authentication Application shall send back a SCP80 Command Packet with no security (SPI1=00h and SPI2=00h) and secured data field containing the result(s) of the command(s) execution (see 8.1.1.2).

8.1.1.2.2 SMS exchange with response in a simple SM MO binary

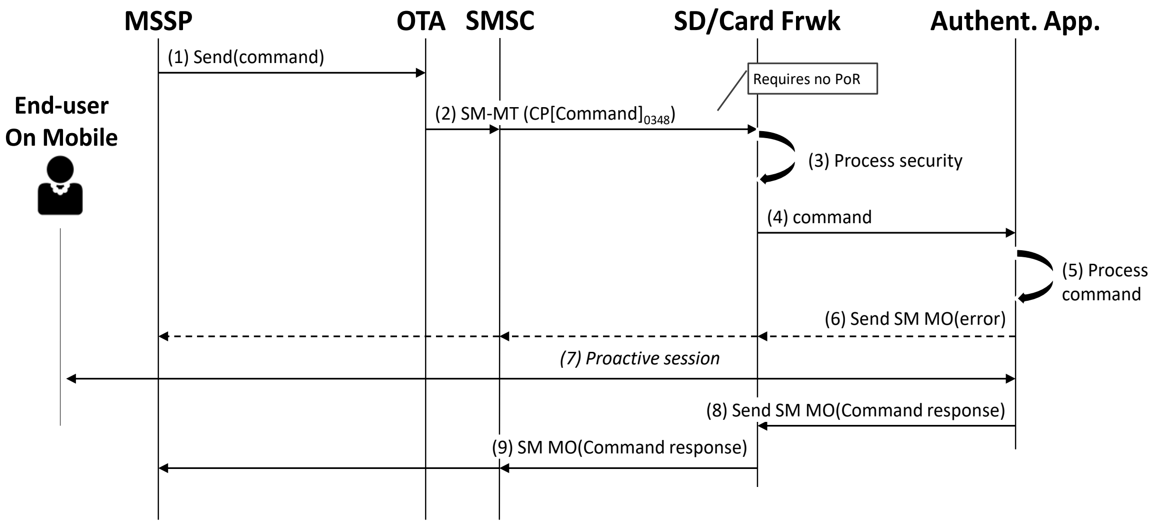


Figure 22 SMS exchange with response in a simple SM MO binary

The sequence here above represents a deployment where the Authentication Server (MSSP) relies on an OTA platform for the secure messaging in the MT way. SMS exchanges would be the same for a deployment where the Authentication Server (MSSP) would embed the secure messaging.

Sequence flow differs from the previous section in:

Step (6): In case of application error during the processing of the command the Card Authentication Application shall send back a binary SM-MO containing directly the error (see 8.1.1.2).

Step (8) (9): Card Authentication Application shall send back a binary SM-MO containing the result(s) of the command(s) execution (see 8.1.1.2).

8.1.2 Security level over SMS

The SM (MT or MO) can be sent with no SCP80 transport security set (SPI1 = 00h). In that case it is mandatory that the Card Authentication Application implements its own applicative transport security. The management of this applicative transport keys is specified as an optional feature.

Nevertheless, to simplify the Card Authentication Application implementation and reduce its size, it is recommended to rely on SCP80 built in security. In that case it is recommended to apply at least:

- the SM (MT or MO) shall make use of a Cryptographic Checksum (CC) with Triple DES in outer-CBC mode, ciphering with Triple DES in outer-CBC mode and counter value higher (two keys (112 bits) as minimum key length, SPI1=16h). This is to be considered as a minimum-security level; AES algorithm may be considered for SIM/UICC able to support it.

- If a Proof Of Receipt (PoR) is requested (SPI2=39h) a Cryptographic Checksum (CC) with Triple DES in outer-CBC mode, ciphering with Triple DES in outer-CBC mode (two keys (112 bits) as minimum key length, SPI1=36h). This is to be considered as a minimum security level; AES algorithm may be considered for SIM/UICC able to support it.

8.1.3 Key wrapping over SMS

This specification covers the case where the Card Authentication Applet can be loaded and installed by SMS OTA over the INT5. This includes also the step where the authentication key of an Authentication Handler is loaded in the Card Authentication Applet.

According to the NIST SP 800-57 Part 1 [24], the key for key wrapping shall have at least the same strength than the wrapped key, e.g. it means that an AES 128 bits key shall be wrapped using at least an AES 128 bits key.

This doesn’t bring particular constraints on the card for Authentication Handler based on Triple DES and OATH as the Triple DES for securing SM is defined for a long time. Nevertheless, in case of Authentication Handler based on AES, the SM-MT used to transport the authentication key of the Authentication Handler shall be secured by at least and AES key of same length. The AES for SCP80 messages is defined only since ETSI release 8.1, 2009-01, a quite new standard release that limits the range of cards that supports it.

Note: This means that the Card Authentication Applet may rely on the cryptographic functions provided by the card OS (instead of re-implementing them internally), as the same cryptographic functions are already mandated at OS level for secure messaging purpose.

8.1.4 INT5b: SCP81 (RAMoHTTP)

The UICC should support to have a SCP81 channel between the UICC and OTA as described in GlobalPlatform Amendment B [27].

The UICC will be triggered by a SMS from the OTA to open a HTTPs channel (SCP81) with OTA. This SMS will support the definition 3GPP TS 31.115 [10] and 3GPP TS 31.116 [11] (and more generally in ETSI TS 102 226 [6]). The GlobalPlatform Amendment B [27] describes the push message and the ETSI 102 223 [4] the different parameters for the Open Channel command.

These SCP81 connectivity parameters could be concatenated with the INT6 instruction sent in the initial SMS MT.

The POST message is used by the Authentication Applet to transmit the INT6 command response as a format string in the body of the request to the OTA Platform.

The POST request defined in 7.1.2 shall contain a specific header field “X-Admin-Targeted-Server” (in addition to the header fields defined in GP Amendment B [27]) containing the MSSP Address defined in 6.6.9.
Upon reception of this POST request, the OTA Platform shall deliver the INT6 command response contained in the HTTP body to the MSSP identified by the header field “X-Admin-Targeted-Server”

The OTA Platform, after having successfully delivered the INT6 command response to the identified MSSP, shall reply with a HTTP response with a status 204 (No content) as defined in GP Amendment B [27] to end the remote session

For a SIM card, these functions shall be realized using functions/mechanism out of scope of this specification.

Depending on the OS the "SM-MT (INT6+Optional (SCP81 parameters))" is sent to the Admin Agent and forwarded to the Authentication Application (as outlined in the figure below) or it might be directly sent to the Authentication Application. In the latter case step (3) and (4) can be skipped.

Likewise for step (6), depending on the OS the Admin Agent can be triggered by the Authentication Application directly (as outlined in the figure below). Alternatively the Admin Agent can also be triggered also by the OTA.

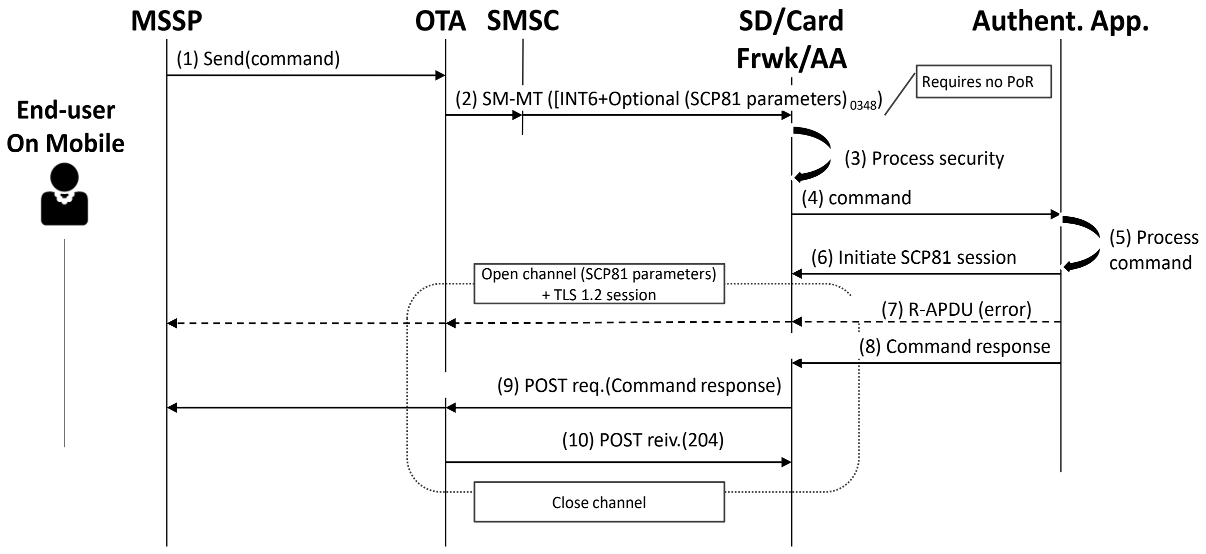


Figure 23 : RAMoHTTP

8.2 INT6: Authentication Server (MSSP) – Card Authentication Application

This section contains the technical descriptions of those functions between the Authentication Server (MSSP) and the Card Authentication Application. PKI is reserved for future release.

Function/Command	Ins	Description
SIGN_TRANSACTION	‘A1’	To generate a signature other a transaction. By the way the same function can be used to authenticate the end-user.
MANAGE_PC	‘B1’	To unblock or reset the Personal Code
GET_DATA	‘B2’	To get applet data
PUT_DATA	‘B3’	To put applet data as well as Authentication handler data or E2E transport key data
CHANGE_STATUS	‘B4’	To activate/deactivate the applet, the E2E transport key or an Authentication Handler individually
DELETE	‘B5’	To delete an Authentication Handler
SCP81 HTTPs connectivity parameters	‘81’	Optional instruction to provide the SCP81 parameters as described in GlobalPlatform Amendment B [27]

Table 8 Functions and commands overview

The Applet shall support to receive several commands in a single SMS.

8.2.1 General coding rules for commands

Commands defined in this section are APDU following ISO/IEC 7816-4 structure.

TLVs in commands shall be managed in the following way:

- Tag order is not significant. Applet shall accept any tag order.
- Tags that are out of the specification shall be simply ignored by the Applet
- Optional tags not supported by the applet shall be simply ignored.
- Duplicated tags (same tag present several time). Recommended behavior is to handle the first one (in the tag order) and ignore the following. Applet could also return an error '6A 80'.

8.2.2 General error conditions

Any of the commands defined in this document may return one of the general error conditions defined in ISO/IEC 7816-4 [3] section 5.1.3 and GlobalPlatform Card Specifications section [7] section 11.1.3 - General Error conditions. In particular:

SW1	SW2	Meaning
'64'	'00'	(Execution error) No specific diagnosis
'67'	'00'	(Checking error) Wrong length in Lc
'69'	'82'	(Checking error) Security Status not satisfied
'69'	'85'	(Checking error) Conditions of use not satisfied
'6A'	'86'	(Checking error) Incorrect P1 P2
'6D'	'00'	(Checking error) Invalid instruction
'69'	'86'	(Checking error) Command not allowed
'6A'	'80'	(Checking error) Incorrect data
'6A'	'81'	(Checking error) Function not supported

Figure 24 : General error conditions

Applet shall check that fields with a length well defined in this document (length is not variable), e.g. 'TransactionId' are received with the expected length. If not the Applet shall return an error '6A 80' without additional data. For those commands requiring 'TransactionId', in case it isn't present, the applet shall have this behavior too.

In case of wrong length in Lc, the behavior must be similar to fields with incorrect length, only the status word ('67 00') must be returned without additional data.

In case of receiving an empty message, the Applet shall return an error '64 00'.

8.2.3 General user interaction considerations

When a command requires the user interaction, a STK proactive command will be executed which may have several possible responses. The Applet shall interpret the responses according to:

- **RES_CMD_PERF**: user clicked the "accept/ok" button. The applet shall execute what the command/flow defines for this situation.
- **RES_CMD_PERF_BACKWARD_MOVE_REQ**: user clicked "back/cancel" button. The applet shall execute what the command/flow defines for this situation.
- **RES_CMD_PERF_HELP_INFO_REQ**: ignore and throw the current proactive command again.

- **RES_CMD_PERF_NO_RESP_FROM_USER**: the timeout for the user response has been reached. The applet shall execute what the command/flow defines for this situation.
- Any other response shall be considered as a cancel.

8.2.4 Remote APDU format

The secure message may contain a single command as well as several commands, which may be grouped in a single secure message. In both cases the commands shall be sent using the Compact Remote command structure (ETSI 102 226 section 5.1.1) and the response shall be sent following the Compact Remote response structure (ETSI 102 226 section 5.1.2). This response structure apply for all the commands defined in this document, regardless if they require the execution of Sim Toolkit [10] proactive commands, hence independently of the mechanism used to reply among the ones defined in section “8.1.1 SMS” or “8.1.4 SCP81: RAMoHTTP”.

Although the Compact Remote specification states that case 4 commands sent in the request should be followed by a GET RESPONSE command in order to receive in the response the output data generated by the case 4 command execution, that is not required in the interaction protocol with the SIM applet hereby described. As a result, the GET RESPONSE command should not be used and case 4 commands must be processed as if they were always followed by a GET RESPONSE command with P3 = Le = ‘00’, meaning all the output data must be sent in the response. This measure makes it easier to process incoming commands (which means less applet size), and saves space for the data field of the case 4 commands.

Optionally the Applet may support to receive commands in Expanded Remote command structure (ETSI 102 226 section 5.2.1). In that case the response shall be sent following the Expanded Remote response structure (ETSI 102 226 section 5.2.2).The general case by SMS when several commands are grouped in a single secure message have some limitations due to the behavior of SIM Toolkit [10] when proactive commands must be executed to require the user interaction. On those cases, the PoR is sent as soon as the proactive command is initiated, impeding to continue the processing of further commands. So, when one of the commands requiring user interaction is included in a single SMS with other commands, it must be the last command of the list. In case any other commands are included in the list after a command requiring user interaction, those remaining command would be simply ignored.

Given the complications that the execution of SIM Toolkit proactive commands introduce, it is strongly recommend to send those commands requiring user interaction within their own SMS. In the current specification there are two commands which apply: Sign transaction and Reset Personal Code (one of the cases of Manage Personal Code).

8.2.5 Transaction ID

All the commands defined in this INT6 are using a Transaction ID (Tag ’01). The Transaction ID is used to correlate an Authentication Server (MSSP) request with a Card Authentication Application response.
The MSSP is responsible to generate the value for each command sent to the card. For security reason the Transaction ID shall be a pure random value.
The same value must be provided by the Card Authentication Application within the response data.

8.2.6 Sign transaction

Function name: SignTransaction

Provider: Card Authentication Application

Related procedure: Authentication/Signature request

Description: This function allows the Authentication Server to request Card Authentication Application to handle the user's signature of a transaction represented by a date/time and a message.

By the way, this function can be used when only a simple authentication of the end-user is required.

Depending of the targeted Authentications Handler, the end-user will be requested a simple 'Click-ok' or a 'Personal Code input'. If 'Personal Code input' is required, this function allows selecting a 'one step' or 'two step' user journey as described in the section 3.2.

Input Data

Name	Description
TransactionID	See section 8.2.5 The TransactionID shall be part of the signature.
TransactionDateTime	Indicates the time stamp of the transaction. The time reference may come from the Authentication Server (MSSP) itself or get by the Authentication Server from a Time server (but this is out of scope of this specification). The TransactionDateTime shall be part of the signature.
Message	The message to be displayed to the user. The Message shall be part of the signature.
Personal code verification mode	A value indicating to the applet the Personal code verification mode (if applicable): - One-step: Personal code has to be verified in a One-step user-journey - Two-steps: Personal code has to be verified in a Two-steps user-journey

Table 9 : Sign Transaction Input data

Output data

Name	Description
TransactionID	The value as provided in the Input data.
TransactionDateTime	The value as provided in the Input data.
Authentication handler type	The type of the authentication handler used for this signature transaction.
Message Authentication Code	The computed Message Authentication Code.

Table 10 : Sign Transaction Output data

Specific error cases

- APPLET_NOT_OPERATIONAL: Applet installation and personalization has not been finalized or is currently deactivated
- SERVICE_NOT_OPERATIONAL: Authentication Handler installation and personalization has not been finalized (e.g. Authentication key not there, personal code not set...) or Authentication Handler is deactivated
- INVALID_AUTH_HANDLER_ID: The targeted Authentication Handler doesn't exist
- SIGNATURE_TRANSACTION_REJECTED: the signature transaction has been cancelled by the user.
- SIGNATURE_TRANSACTION_EXPIRED: the signature transaction expired without action by the user.
- PCODE_BLOCKED: the Personal code is blocked
- PCODE_NOT_CHANGED: Personal code must be changed before authentication request can be answered

8.2.6.1 Command description

Command message:

The SIGN TRANSACTION command message is coded according to the following table:

Code	Value	Meaning
CLA	‘0X’	See ISO/IEC 78116-4 [3]
INS	‘A1’	SIGN_TRANSACTION
P1	‘XX’	See coding of P1
P2	‘XX’	Identifier of the Authentication Handler to use.
Lc	‘XX’	Length of the data field
Data	‘XXX...’	See table hereunder
Le	‘00’	

Table 11 : Sign Transaction command message

Coding of P1:

Indicates the Personal Code verification mode. It is relevant only when the targeted Authentication Handler applies for Personal Code verification.

‘01’: one-step journey

‘02’: two-steps journey

Others value are RFU (If one of this RFU value is received by the applet, the applet shall apply the one-step journey as the default behavior)

Data field:

Tag	Length	Value	MOC
‘01’	4	Transaction ID	M
‘02’	4	Transaction date time. Time elapsed in seconds since midnight UTC of January 1, 1970 [UT] coded on 4 bytes.	M
‘8D’	0-X	Message. A text string to be displayed with following structure: - First byte to indicate the Data coding scheme of the message to be displayed. - Remaining bytes: message to be displayed	M

Table 12 : Sign Transaction Command message Data field

A null Text string (tag ’8D’) shall be coded with Length = '00', and no Value part.

The Data coding scheme byte is coded as for SMS Data coding scheme defined in TS 102 223 [4].

The following Data coding scheme values shall be supported:

- ’00': GSM default alphabet 7 bits packed;
- '04': GSM default alphabet 8 bits;
- '08': UCS2.

Others Data coding scheme may be supported.

8.2.6.2 Interoperability issues

The applet may use the underlying GSM Sim Tool Kit proactive commands GET INPUT or DISPLAY TEXT for user interaction depending on the targeted authentication handler “Personal Code Input” or “Click-ok”. These commands limit the maximum text length to be displayed because of a maximum total length of the APDU of 255 bytes. Taking into account header and optional header and that two concatenated SMS can transport a total of 220 bytes of encoded text to be displayed with the "SIGN TRANSACTION" command, we recommend for interoperability a maximum of 2 concatenated SMS transporting a 220 bytes maximum message to be displayed of encoded text (first byte of the Message is the Data Coding Scheme).

Interoperability Recommendations:

- Applet should be able to process the APDU payload of 2 concatenated SMSs.

- The encoded text to be displayed should not exceed 220 bytes

A maximum of 220 bytes for the encoded text to be displayed leads to the following maximum character length for the Data Coding schemes supported:

'00': GSM default alphabet 7 bits packed: 251 characters

'04': GSM default alphabet 8 bits: 220 characters

'08': UCS2 : 110 characters

Interoperability with v 2.1.1 or earlier versions of this specification:

The main impacts on MNOs implementing the support for 2 concatenated SMS are derived from the fact that most of the old versions of the SIM applet (compliant with v2.1.1 or earlier of this specification) will not be able to handle the bigger amount of data they could now receive, as they are expected to be optimized in size to increase the compatibility with the different SIM cards. A new version of the applet will then be required, with larger internal buffers that cope with the new size limit of the messages received.

As a result, the impacts on a MNO that decides to add this feature depend on whether it already has SIM applets deployed and working in a certain market or not:

- If there are not SIM applets in use:
    - o The MNO will need to make sure the version of the applet they install in the SIM cards supports the new feature, as explained above.
    - o The MNO will need to make sure the MSSP is configured to process signature requests carrying messages to be displayed as big as the new limit imposed by the use of 2 concatenated messages. The same with any other server-side component that may be affected.
- If there are already SIM applets on the field:
    - o The MNO will need to distinguish two types of users of the SIM applet, those with the new version and those with the legacy version. The MSSP would need to be aware of that division and take it into account when processing a signature request for each particular user, in order to apply the former limit or the new limit for the message to be displayed.
    - o The MNO will need to define a migration strategy for users of the legacy version of the applet (e.g. having the new version preinstalled in new SIM cards or having the old version replaced by the new one by means of the OTA platform or a combination of both).
    - o Once the legacy version of the applet is replaced by the new one in a user's SIM card, it will not be possible to process signature requests until the configuration of the new applet is launched from the server (activating the authentication handlers, setting the keys, etc.) and the user is asked to define the Personal Code again (in case LoA 3 wants to be used). So the MNO will need to be aware of the precise moment the applet has been updated for each

particular user, and coordinate the other parties involved, MSSP and final user, so that the authenticator works again as soon as possible.

Note that these points, especially the last two, pose an impact both in the technical and in the user experience sides. For example, there is an inherent technical risk in the management of the applets in the SIM cards via OTA, as it is prone to errors due to variations in the coverage of mobile phones or some other factors, so it is advisable to avoid it whenever possible, and the orchestration of that management with the MSSP and the user can be technically complex too, depending on the systems involved and how they interface with one another. But also, from a user experience perspective, it can be difficult to convey in the right way that the SIM Applet Authenticator is not going to be available for a certain period of time, until it is reconfigured, and get the user involved in the process, resetting the Personal Code when requested.

MNOs must consider all these factors to make the decision if, when and how they want to implement the 2 concatenated SMS feature in the different markets where they are present.

8.2.6.3 Command response processing

Processing for Simple applet:

The applet shall return Transaction ID, the transaction data-time and authentication handler type in the response.

The signature of this data is ensured by the secure messaging layer.

Processing for 3DES-CBC

The processing to compute the 3DES-CBC MAC signature is described in section Annex B.

Processing for AES-CMAC

The processing to compute the AES-CMAC is described in section Annex C.

Processing for OATH OCRA

This function corresponds to the One-Way Challenge-Response as described in the RFC 6287 [2], section 7.1. Refer to Annex A for signature computation.

Response message:

Data Field Returned in the Response Message

Tag	Length	Value	MOC
‘01’	4	Transaction ID. This shall be the same value as the one received in the command message.	M
‘02’	4	Transaction date time. Time elapsed in seconds since midnight UTC of January 1, 1970 [UT] coded on 4 bytes. This shall be the same value as the one received in the command message.	M
‘10’	1	Authentication handler type (B1-B8).	M
‘11’	0-XX	Message Authentication Code	C

Table 13: Command Response data fields

The ‘Transaction Id’ (tag ’01’) and ‘Transaction date time’ (tag ‘02’) of the command message shall be returned back within the response message to allow MSSP pairing with the initial authentication request, including the cases in which the command returns and error.

The Message Authentication Code field shall contain the result of the computation process implemented by the Authentication Handler. This field may be missing if the Authentication used is one of the '"Click OK" with authentication based on secure messaging layer' or '"Personal Code input" with authentication based on secure messaging layer'.

Message Authentication Code field must not be included in case of error to avoid potential attacks.

Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may either return a general error condition as listed in section 8.2.2 or one of the following error conditions.

SW1	SW2	Meaning
'69'	'86'	Command not allowed. Indicates APPLET_NOT_OPERATIONAL
'69'	'85'	Conditions of use not satisfied. Indicates SERVICE_NOT_OPERATIONAL
'69'	'90'	Indicates PC_CODE_BLOCKED
'69'	'91'	Indicates PC_CODE_NOT_CHANGED
'65'	'03'	Indicates SIGNATURE_TRANSACTION_REJECTED
'65'	'04'	Indicates SIGNATURE_TRANSACTION_EXPIRED
'65'	'AA'	Indicates INVALID_AUTH_HANDLER_ID

Table 14 : Processing State response message

8.2.7 Manage Personal Code

Function name: ManagePersonalCode

Provider: Card Authentication Application

Related procedure: Reset Personal Code/Unblock Personal Code

Description: This function allows managing the Personal Code. Two actions are possible:

- Unblock the user's Personal Code
- Reset the user's Personal Code

There is only one Personal Code defined for the Applet. The same Personal Code is used whatever the Authentication Handler that requests it.

This function shall be used to initialize the Personal Code before the first use of an Authentication Handler requiring a Personal Code.

Input Data

Name	Description
TransactionID	See section 8.2.5
Mode	A value indicating the action to perform: - Unblock - Reset (allow also first initialization)

Table 15 : Manage Personal Code – Input Data

Output data

Name	Description
TransactionID	The value as provided in the Input data.

Table 16 : Manage Personal Code – Output data

Specific error cases

- SERVICE_NOT_OPERATIONAL: applet installation and personalization has not been finalized (e.g. seed not there, personal code not set...)
- PCODE_NOT_BLOCKED: the Personal Code is not blocked
- PCODE_NOT_RESET: the Personal Code is not reinitialized (procedure failed), including the case where the Personal Code has not been created (first creation).

8.2.7.1 Command description

Command message:

The MANAGE_PC command message is coded according to the following table:

Code	Value	Meaning
CLA	'0X'	See ISO/IEC 78116-4 [3]
INS	'B1'	MANAGE_PC
P1	'00' '01'	Unblock PC (optional mode) Reset PC
P2	'XX'	Not used
Lc	'06'	In case Transaction ID is added to data. Not present otherwise
Data		See table hereunder
Le	'00'	Only present in case of TransactionID expected in the response by the MSSP

Table 17 : The MANAGE_PC command message

Data field:

Tag	Length	Value	MOC
'01'	4	Transaction ID. The Authentication Server (MSSP) may use this field or not.	O

Table 18: Command message Data field

Processing:

On reception of this command the applet shall:

- If P1='00', unblock the Personal Code of the user (Personal Code value remains unchanged). This mode is optional to reduce the final Applet code size, as the RESET_PC mode includes basically the unblocking of the PC but with a different end-user journey (end-user is always prompted to enter a PC that can new or distinct). If not supported the Card Authentication Application shall return a '6A 81' code.

- If P1='01', reinitialize the Personal Code by executing the end user journey described in section 3.5, or 3.3 in case of first creation. The maximum number of attempts mentioned in this journey is the same that the one used for the Sign transaction, Section 6.6.5 "Max Personal Code attempts"

Response message:

Data Field Returned in the Response Message

Tag	Length	Value	MOC
'01'	4	Transaction ID. This shall be the same value as the one received in the command message. If no value was received, then no value shall be returned.	O

Table 19 Command message response

Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may either return a general error condition as listed in section 8.2.2 or one of the following error conditions.

SW1	SW2	Meaning
'69'	'85'	Conditions of use not satisfied. Indicates SERVICE_NOT_OPERATIONAL
'62'	'01'	Indicates PC_CODE_NOT_BLOCKED
'65'	'02'	Indicates PC_CODE_NOT_RESET

Table 20 Command message Processing State response message

8.2.8 Get applet data

Function name: GetAppletData

Provider: Card Authentication Application

Related procedure: No specific procedure; can be used when required

Description: This function allows retrieving information about the targeted applet:

- Supported Authentication Handler types
- GSMA version
- Activation flag
- Installation date
- Max PC attempt
- PC length
- MSSP Address
- List of created Authentication Handler
- E2E transport key flag
- E2E transport key details
- Applet Release
- Applet Locale

Each of this data is described in section 6.6.

Input Data

Name	Description
TransactionID	See section 8.2.5

Table 21 : Get applet input data

Output data

Name	Description
TransactionID	The value as provided in the Input data.
Supported Authentication Handler types	See section 6.6.
GSMA version	See section 6.6.
Activation flag	See section 6.6.
Installation date	See section 6.6.
Max PC attempt	See section 6.6.
PC length	See section 6.6.
E2E transport key flag	See section 6.6.
E2E transport key type	Indicates the transport key type stablished.
MSSP Address	See section 6.6.
List of created Authentication Handler	For each handler indicates the handler identifier, the handler type, and the handler state.
Applet Release	See section 6.6.
Applet Locale	See section 6.6.

Table 22 : Get applet output data

Specific error cases

None

8.2.8.1 Command description

Command message:

The GET_DATA command message is coded according to the following table:

Code	Value	Meaning
CLA	‘0X’	See ISO/IEC 78116-4 [3]
INS	‘B2’	GET_DATA
P1	‘XX’	See coding of P1
P2	‘XX’	Not used
Lc	‘06’	In case Transaction ID is added to data. Not present otherwise
Data		See table hereunder
Le	‘00’	

Table 23 : Get data command message

Coding of P1:

The different P1 values allow to specify what applet information the MSSP wants to receive:

- ‘00’: send all the information
- ‘01’: send standard information only (skip ‘FF’ tag)
- ‘02’: send proprietary information (send only ‘FF’ tag)
- ‘03’: send standard information without authenticators (skip ‘B0’ and ‘FF’ tags)
- ‘04’: send only the authenticators information (send only ‘B0’ tag)

The applet must support P1=’00’, all the other values are optional.

Data field:

Tag	Length	Value	MOC
‘01’	4	Transaction ID. The Authentication Server (MSSP) may use this field or not.	O

Table 24 : Get data command message data field

Response message:

The command returns a list of TLVs.

Data Field Returned in the Response Message

Tag	Length	Value	MOC
‘01’	4	Transaction ID. This shall be the same value as the one received in the command message. If no value was received, then no value shall be returned.	O
‘A0’	2	Supported Authentication Handler types (see section 6.6.1 for coding of the 2 bytes)	M
‘A1’	2	GSMA version	M
‘A2’	1	Applet Activation flag	M
‘A3’	4	Installation date	M
‘A4’	1	Max PC attempt	M
‘A5’	1	PC length	M
‘A7’	var	MSSP Address	C
‘A8’	1	E2E transport key flag	M
‘A9’	1	E2E transport key type (‘82’= 3DES-CBC key type, ‘88’= AES-CMAC key type)	C
‘B0’	var	Tag for list of Authentication Handler descriptors. The ‘B0’ content data is a list of Authentication Handler type tag (as defined by 8.2.10.1)	M
‘Bx’	6	Authentication Handler type tag (x=1 to 8). B1: “’Click OK’ with authentication based on secure messaging layer” B2: “’Personal Code input’ with authentication based on secure messaging layer” B3: “’Click OK’ with authentication based on 3DES-CBC” B4: “’Personal Code input’ with authentication based on 3DES-CBC” B5: “’’Click OK’ with authentication based on OATH OCRA” B6: “Personal Code input’ with authentication based on OATH OCRA” B7: “’Click OK’ with authentication based on AES-CMAC” B8: “’Personal Code input’ with authentication based on AES-CMAC” The content of this tag are the two next defined tags.	O
‘AA’	1	Authentication Handler identifier	M
‘AB’	1	Authentication Handler state (‘00’= deactivated, ‘01’ =activated)	M
…			
‘C0’	15	Applet Release	O
‘D1’	9	MC_RID: 5 bytes MC_PIX : 4 bytes	C

Tag	Length	Value	MOC
‘D2’	2	Version Data structure: major version (first byte) + minor version (second byte)	C
‘C1’	6	Applet Locale: Language used (2bytes) + Script used (2 bytes) + Country of use (2bytes)	O
‘FF’	var	Tag reserved to describes proprietary features. This tag is optional; may contains additional tags which meaning is manufacturer dependent.	O

Table 25 : The Get data Response message

The MSSP address, Applet Release and Applet Locale fields are optional. These values may not be returned if not supported by the Card Authentication Application. See section 8.2.9 for coding of this value.

The ‘B0’ tag is mandatory in case the P1 value indicates that authentication handlers data must be included. When no Authentication Handler is not yet created, the command shall return a ‘B0 00’ value indicating an empty list.

Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may return a general error condition as listed in section 8.2.2. In case that the provided P1 value isn’t supported by the applet, ‘6A81’ (Function not supported) must be send.

In case the amount of information to return is too big to be included in one single SMS, it must be truncated and this situation shall be indicated by status bytes '62' ’00’.

Note: the amount of data available to send as response in one single SMS is variable, depending on the parameters configured in the incoming 3.48 SMS containing the Get Applet data command. The applet must analyse those parameters to guarantee the correct truncation of the data.

8.2.9 Set Applet data

Function name: Set Applet data

Provider: Card Authentication Application

Related procedure: First registration of an end-user/Un-registering and end-user

Description: This function allows setting personalization data required for the applet to be functioning:

- Installation date
- Max PC attempt
- PC length
- MSSP address

Each of this data is described in section 6.6.

Input Data

Name	Description
TransactionID	See section 8.2.5
Installation date	See section 6.6.4
Max PC attempts	See section 6.6.5
PC length	See section 6.6.6
MSSP Address	See section 6.6.9

Table 26 : Set Applet data – input

Output Data

Name	Description
TransactionID	The value as provided in the Input data.

Table 27 : Set Applet Data - Output

Specific error cases

- INVALID_INSTALL_DATE: the provided value is invalid
- INSTALL_DATE_ALREADY_FIXED: the installation date is already set in the Applet
- INVALID_MAX_PC_ATTEMPT: the provided value is invalid
- MAX_PC_ATTEMPT_ALREADY_FIXED: the max PC attempts value is already set and cannot be changed because and Authentication Handler requiring a PC is already created
- INVALID_PC_LENGTH: the provided value is invalid
- INVALID_MSSP_ADDRESS: the provided value is invalid

8.2.9.1 Command description

Command message:

The PUT_DATA command message is coded according to the following table:

Code	Value	Meaning
CLA	‘0X’	See ISO/IEC 78116-4 [3]
INS	‘B3’	PUT_DATA
P1	‘00’	Indicates applet data
P2	‘XX’	Not used
Lc	‘XX’	Length of the data field
Data	‘XXX...’	See table hereunder
Le	‘00’	Only present in case of TransactionID expected in the response by the MSSP

Table 28: The put data command message

Data field:

Tag	Length	Value	MOC
‘01’	4	Transaction ID. The Authentication Server (MSSP) may use this field or not.	O
‘A3’	4	Installation date	O
‘A4’	1	Max PC attempts	O
‘A5’	1	PC length	O
‘A7’	var	MSSP address	O

Table 29 : The PUT_DATA data field

The MSSP Address shall contain the structure defined in section 6.6.9:

- 1 Byte for Length
- 1 Byte for TON API
- 0 to 10 Bytes Dialing number string

Both values shall be coded according to for TS 102 223 [4].

Processing

This command allows putting any combination of the listed data in the data field.

If any of the provided value is invalid the full command shall be rejected and none of the values shall be updated.

The 'Installation data' shall be set only once. Further change of installation date shall be rejected.

The 'Max PC attempts' shall be set before the creation of the first Authentication Handler requiring a Personal Code. Any further change of 'Max PC attempts' after creation of a Personal Code shall be rejected. The MSSP shall reset the Personal Code through a MANAGE PERSONAL CODE command after a Max PC attempts modification to be taken into account.

The 'PC length' can be changed at any time. But when changed, the MSSP shall reset the Personal Code through a MANAGE PERSONAL CODE command after a PC length modification to be taken into account. . In case the provided 'PC length' doesn't change compared to the already set value, the Applet shall have the same behavior as if there was no 'PC length' value.

The 'MSSP Address' can be changed at any time.

Response message:

Data Field Returned in the Response Message

Tag	Length	Value	MOC
'01'	4	Transaction ID. This shall be the same value as the one received in the command message. If no value was received, then no value shall be returned.	O

Table 30 PUT_DATA response message

Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may either return a general error condition as listed in section 8.2.2 or one of the following error conditions.

SW1	SW2	Meaning
'65'	'A3'	Indicates INVALID_INSTALL_DATE
'65'	'A8'	Indicates INSTALL_DATE_ALREADY_FIXED
'65'	'A4'	Indicates INVALID_MAX_PC_ATTEMPT
'65'	'A9'	Indicates MAX_PC_ATTEMPT_ALREADY_FIXED
'65'	'A5'	Indicates INVALID_PC_LENGTH
'65'	'A7'	Indicates INVALID_MSSP_ADDRESS

Table 31 PUT_DATA Processing State response message

8.2.10 Set Authentication Handler data

Function name: Set Authentication Handler data

Provider: Card Authentication Application

Related procedure: First registration of an end-user

Description: This function allows setting specific Authentication Handler data field. It may be an update of data of an already created Authentication Handler, as well as setting data for a new Authentication handler.

The data to set depends on the type of the Authentication Handler to update/create.

This function can be used by the Authentication Server at any time and whatever the applet status (activated or deactivated).

Input Data for Authentication Handler based on secure messaging layer

None

Input Data for Authentication Handler based on 3DES-CBC or AES-CMAC

Name	Description
TransactionID	See section 8.2.5
Authentication key	The authentication key is used to compute the authentication data. The key shall be 112 bits (2keys) minimum length for 3DES-CBC. The key shall be 128 bits length for AES-CMAC.

Table 32: Set Authentication Handler based on 3DES-CBC or AES-CMAC input data

Input Data for Authentication Handler based on OATH OCRA

Name	Description
TransactionID	See previous table
Authentication key	The authentication key is used to compute the authentication data. The Authentication key corresponds to the Key (K) designated in RFC 4226 [1] and in RFC 6287 [2]. The key shall be 20 bytes minimum length.
Counter	The counter is used to compute the authentication data. Setting this parameter is optional. If the parameter is not set at creation time, the applet shall supply a default value of '0x0000000000000000'. If the parameter is not used, then it is not concatenated. The Counter corresponds to the Counter (C) designated in RFC 4226 [1] and in RFC 6287 [2] and is consequently an unsigned 8-byte value.

Table 33 Input Data for Authentication Handler based on OATH OCRA

Output Data

Name	Description
TransactionID	The value as provided in the Input data.

Table 34 : output for Authentication Handler

Specific error cases

- INVALID_AUTH_HANLDER_TYPE: the provided Authentication Handler type is invalid (type not exist or not supported)
- INVALID_AUTH_HANLDER_ID: the provided Authentication Handler identifier is invalid or an Authentication handler with same identifier already exist
- INVALID_AUTH_KEY: the provided authentication key is invalid (wrong length...)
- INVALID_COUNTER: the provided value is invalid
- MAXIMUN_AMOUNT_OF_AUTHENTICATORS_ALREADY_REACHED: the applet implementations may limit the maximum number of authentication handler that can be created. There may be different reasons to select this parameter like maintain the memory consumption among certain levels or to return all the applet data in one single Get applet

data command. The maximum number of authenticators support by the applet is an implementation detail.

8.2.10.1 Command description

__Command message:__

The PUT_DATA command message is coded according to the following table:

Code	Value	Meaning
CLA	‘0X’	See ISO/IEC 78116-4 [3]
INS	‘B3’	PUT_DATA
P1	‘01’ ‘02’	Update Authentication Handler data Create Authentication Handler with data
P2	‘XX’	Not used
Lc	‘XX’	Length of the data field
Data	‘XXX...’	See table hereunder
Le	‘00’	Only present in case of TransactionID expected in the response by the MSSP

Table 35 : PUT_DATA command message

Data field:

Data field for an Authentication Handler of type “’Click OK’ with authentication based on secure messaging layer”:

Tag	Length	Value	MOC
‘01’	4	Transaction ID. The Authentication Server (MSSP) may use this field or not.	O
‘B1’	3	Authentication Handler type tag indicating a “’Click OK’ with authentication based on secure messaging layer”. The content of this tag is the next defined tag.	M
‘AA’	1	Authentication Handler identifier	M

Table 36 Data field for “Click Ok” authentication

Data field for an Authentication Handler of type “’Personal Code input’ with authentication based on secure messaging layer”:

Tag	Length	Value	MOC
‘01’	4	Transaction ID. The Authentication Server (MSSP) may use this field or not.	O
‘B2’	3	Authentication Handler type tag indicating a “’Personal Code input’ with authentication based on secure messaging layer”. The content of this tag is the next defined tag.	M
‘AA’	1	Authentication Handler identifier	M

Table 37 : Data Field for “Personal Data Code Input”

Data field for an Authentication Handler of type “’Click OK’ with authentication based on 3DES-CBC”:

Tag	Length	Value	MOC
‘01’	4	Transaction ID. The Authentication Server (MSSP) may use this field or not.	O
‘B3’	var	Authentication Handler type tag indicating a “’Click OK’ with authentication based on 3DES-CBC”. The content of this tag are the two next defined tags.	M
‘AA’	1	Authentication Handler identifier	M
‘AC’	var	Authentication key	M

Table 38: Data field for Authentication Handler of type "Click OK"

Data field for an Authentication Handler of type "'Personal Code input' with authentication based on 3DES-CBC":

Tag	Length	Value	MOC
'01'	4	Transaction ID. The Authentication Server (MSSP) may use this field or not.	O
'B4'	var	Authentication Handler type tag indicating a "'Personal Code input' with authentication based on 3DES-CBC". The content of this tag are the two next defined tags.	M
'AA'	1	Authentication Handler identifier	M
'AC'	var	Authentication key	M

Table 39 : Data field for "Persaonal Code Input" / based on 3DES-CBC

Data field for an Authentication Handler of type "'Click OK' with authentication based on OATH OCRA":

Tag	Length	Value	MOC
'01'	4	Transaction ID. The Authentication Server (MSSP) may use this field or not.	O
'B5'	var	Authentication Handler type tag indicating a ""Click OK' with authentication based on OATH OCRA". The content of this tag are the three next defined tags.	M
'AA'	1	Authentication Handler identifier	M
'AC'	var	Authentication key	M
'AD'	8	Counter	O

Table 40 : Data field for "Click OK" authentication handler type / based on OATH OCRA

Data field for an Authentication Handler of type "'Personal Code input' with authentication based on OATH OCRA":

Tag	Length	Value	MOC
'01'	4	Transaction ID. The Authentication Server (MSSP) may use this field or not.	O
'B6'	var	Authentication Handler type tag indicating a "Personal Code input' with authentication based on OATH OCRA". The content of this tag are the three next defined tags.	M
'AA'	1	Authentication Handler identifier	M
'AC'	var	Authentication key	M
'AD'	8	Counter	O

Table 41 Data field for "peraonal Code Input" / based on OATH OCRA

Data field for an Authentication Handler of type "'Click OK' with authentication based on AES-CMAC":

Tag	Length	Value	MOC
'01'	4	Transaction ID. The Authentication Server (MSSP) may use this field or not.	O
'B7'	var	Authentication Handler type tag indicating a "'Click OK' with authentication based on AES-CMAC". The content of this tag is the next defined tag.	M
'AA'	1	Authentication Handler identifier	M
'AC'	var	Authentication key	M

Table 42 Data field for "Click OK" Authentication Handler based on AES-CMAC

Data field for an Authentication Handler of type "'Personal Code input' with authentication based on AES-CMAC":

Tag	Length	Value	MOC
‘01’	4	Transaction ID. The Authentication Server (MSSP) may use this field or not.	O
‘B8’	var	Authentication Handler type tag indicating a “’Personal Code input’ with authentication based on AES-CMAC”. The content of this tag are the two next defined tag.	M
‘AA’	1	Authentication Handler identifier	M
‘AC’	var	Authentication key	M

Table 43 Data field for “Personal Code Input” / based on AES-CMAC

Processing

The command can contain only one tag BX (meaning that the command can only create or update one handler).

In ‘Update’ mode the ‘Authentication Handler identifier’ identifies the handler to update. The Authentication Handler shall exist and the type shall match, else an error shall be returned.

In ‘Create’ mode the Authentication Handler identifier allows to set the handler identifier value.

If any of the provided value is invalid the full command shall be rejected and none of the values shall be updated.

<u>Response message:</u>

Data Field Returned in the Response Message

Tag	Length	Value	MOC
‘01’	4	Transaction ID. This shall be the same value as the one received in the command message. If no value was received, then no value shall be returned.	O

Table 44 Data field in the Response PUT_DATA

Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may either return a general error condition as listed in section 8.2.2 or one of the following error conditions.

SW1	SW2	Meaning
‘65’	‘BB’	Indicates INVALID_AUTH_HANLDER_TYPE
‘65’	‘AA’	Indicates INVALID_AUTH_HANLDER_ID
‘65’	‘AC	Indicates INVALID_AUTH_KEY
‘65’	‘AD’	Indicates INVALID_COUNTER
‘65’	‘AE’	Indicates MAXIMUN_AMOUNT_OF_AUTHENTICATORS_ALREADY_REACHED

Table 45 Processing State returned PUT_DATA

8.2.11 Change applet status

Function name: ChangeAppletStatus

Provider: Card Authentication Application

Related procedure: First registration of an end-user/ Un-registering an end-user

Description: This function allows:

- Activating or deactivating the applet.
- Activating or deactivating an Authentication Handler
- Activating or deactivating the E2E transport key

This function may be sent to the applet at any time. But activating the applet, or an Authentication Handler or E2E transport key, requires that all mandatory data have been previously set; else activation shall fail.

Input Data

Name	Description
TransactionID	See section 8.2.5
ActivationFlag	Activation flag

Table 46 ChangeAppletStatus Input Data

Output Data

Name	Description
TransactionID	The value as provided in the Input data.

Table 47 ChangeAppletStatus Output data

Specific error cases

- MISSING DATA: the applet, the handler or the E2E transport key cannot be activated because missing some personalization data
- INVALID_AUTH_HANLDER_ID: the provided Authentication Handler identifier is invalid

8.2.11.1 Command description

Command message:

The CHANGE_STATUS command message is coded according to the following table:

Code	Value	Meaning
CLA	‘0X’	See ISO/IEC 78116-4 [3]
INS	‘B4’	CHANGE_STATUS
P1	‘XX’	See coding of the P1
P2	‘XX’	See coding of the P2
Lc	‘06’	In case Transaction ID is added to data. Not present otherwise
Data		See table defined in data field section
Le	‘00’	Only present in case of TransactionID expected in the response by the MSSP

Table 48 CHANGE_STATUS command message

Coding of the P1:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	0	-	-	-	-	-	-	Change applet status
0	1	-	-	-	-	-	-	Change Authentication Handler status
0	0	1						Change E2E Transport status
-	-	-	-	-	-	-	0	Deactivate the applet, E2E transport key or handler (according to b8b7b6)
-	-	-	-	-	-	-	1	Activate the applet, E2E transport key or handler (according to b8b7b6)

Table 49 CHANGE_STATUS coding of the P1

Some examples of P1 values:

‘80’ – Deactivate the Applet
‘81’ – Activate the Applet
‘40’ – Deactivate the Authentication Handler (identifier is given by P2)
‘41’ – Activate the Authentication Handler (identifier is given by P2)
‘20’ – Deactivate E2E transport key
‘21’ – Activate E2E transport key
‘E0’ – Deactivate Applet, Authentication Handler and E2E transport key
‘E1’ – Activate Applet, Authentication Handler and E2E transport key

Coding of the P2:

If P1 indicates to change the status of an Authentication handler, P2 is used to contain the Authentication Handler identifier.

P2 is not used when P1 doesn’t indicate a change in the status of an Authentication handler.

Data field:

Tag	Length	Value	MOC
‘01’	4	Transaction ID. The Authentication Server (MSSP) may use this field or not.	O

Table 50 CHANGE_STATUS Data field

Processing

The command can be used to simultaneously change the state of the Applet, the state of one Authentication Handler and the state of the E2E transport key flag.

In case one of the requested status change cannot be done (conditions not satisfied), the Applet shall reject the full command without doing any of the requested change and return the first error condition encountered).

The applet can be activated or deactivated without checking any condition.

An Authentication Handler can be deactivated without checking any condition.

An Authentication Handler can only be activated if all necessary personalization data have been provided.

The Applet shall reject the activation of any Authentication Handler requiring a PC until the PC is set by the end-user after a successful execution of MANAGE_PC command.

The E2E transport key can be deactivated without checking any condition.

The E2E transport key can only be activated if it has been previously defined, as described in “Set E2E transport key” section.

In case the received command asks to change the status of something that has already the requested status (e.g. activate the applet when the applet is already activated), the Applet shall simply do nothing and return a SW = 90 00.

Response message:

Data Field Returned in the Response Message

Tag	Length	Value	MOC
‘01’	4	Transaction ID. This shall be the same value as the one received in the command message. If no value was received, then no value shall be returned.	O

Table 51 CHANGE_STATUS data field response

Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may either return a general error condition as listed in section 8.2.2 or one of the following error conditions.

SW1	SW2	Meaning
‘69’	‘84’	Indicates MISSING DATA
‘65’	‘AA’	Indicates INVALID_AUTH_HANDLER_ID

Table 52 CHANGE_STATUS processing state response

8.2.12 Delete Authentication Handler

Function name: DeleteAuthenticationHandler

Provider: Card Authentication Application

Related procedure: Un-registering an end-user

Description: This function allows deleting an Authentication Handler.

This function may be sent to the applet at any time. This function shall be accepted by the applet whatever its state or the state of the targeted Authentication Handler.

Input Data

Name	Description
TransactionID	See section 8.2.5
Authentication Handler id	Identifier of the Authentication Handler to delete

Table 53 Delete Authentication handler input data

Output Data

Name	Description
TransactionID	The value as provided in the Input data.

Table 54 Delete Authentication handler output data

Specific error cases

- INVALID_AUTH_HANLDER_ID: the provided Authentication Handler identifier is invalid (e.g. the handler doesn’t exist)

8.2.12.1 Command description

Command message:

The DELETE command message is coded according to the following table:

Code	Value	Meaning
CLA	‘0X’	See ISO/IEC 78116-4 [3]
INS	‘B5’	DELETE
P1	‘XX’	Not used

P2	‘XX’	Authentication Handler identifier
Lc	‘06’	In case Transaction ID is added to data. Not present otherwise
Data		See table hereunder
Le	‘00’	Only present in case of TransactionID expected in the response by the MSSP

Table 55 DELETE command message

Data field:

Tag	Length	Value	MOC
‘01’	4	Transaction ID. The Authentication Server (MSSP) may use this field or not.	O

Table 56 DELETE command data field

Processing

The applet shall return an error if the Authentication Handler identified by its identifier doesn’t exist.

The Authentication Handler can be deleted whatever the Applet state or the Authentication Handler state.

The Applet shall also delete all data related to the deleted Authentication Handler.

The Applet shall release all the resources used by the deleted Authentication Handler.

Response Message:

Data Field Returned in the Response Message

Tag	Length	Value	MOC
‘01’	4	Transaction ID. This shall be the same value as the one received in the command message. If no value was received, then no value shall be returned.	O

Table 57 Delete authentication handler data field returned in response

Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may either return a general error condition as listed in section 8.2.2 or one of the following error conditions.

SW1	SW2	Meaning
‘65’	‘AA’	Indicates INVALID_AUTH_HANLDER_ID

Table 58 Delete authenticaitno handler processing state returned in the response

8.2.13 Set E2E transport key

Function name: Set E2E transport key

Provider: Card Authentication Application

Related procedure: XXX

Description: This function allows setting specific E2E transport key among the MSSP and the applet, to create and additional security layer due to lack of OTA security or to increase the security provided by OTA.

Input Data for Authentication Handler based on 3DES-CBC or AES-CMAC

Name	Description
E2E key type	3DES-CBC or AES-CMAC
Encryption key	The encryption key is used to encrypt/decrypt the E2E channel among the MSSP and the applet. The key shall be 112 bits (2keys) minimum length for 3DES-CBC. The key shall be 128 bits length for AES-CMAC.

Table 59 : Set E2E Transport key input data based on EDES-CBC or AES-CMAC

Output Data

None

Specific error cases

- INVALID_E2E_KEY_TYPE: the provided E2E key type is invalid (type doesn't exist or isn't supported)
- INVALID_AUTH_KEY: the provided E2E key is invalid (wrong length...)

8.2.13.1 Command description

Command message:

The PUT_DATA command message is coded according to the following table:

Code	Value	Meaning
CLA	'0X'	See ISO/IEC 78116-4 [3]
INS	'B3'	PUT_DATA
P1	'03'	Create or Update E2E transport key
P2	'82' '88'	3DES-CBC key type AES-CMAC key type
Lc	'XX'	Length of the data field
Data	'XXX...'	See table hereunder
Le	'00'	Only present in case of TransactionID expected in the response by the MSSP

Table 60 Set E2E transport key PUT_DATA command message

Data field:

Tag	Length	Value	MOC
'01'	4	Transaction ID. The Authentication Server (MSSP) may use this field or not.	O
'AC'	var	Encryption key	M

Table 61 Set E2E transport key data field

Processing

In case there wasn't a transport key previously set, the provided values are stablished but the applet won't start to use E2E transport encryption until the E2E transport key flag is activated.

The E2E transport key flag must be deactivated to allow the execution of this command to avoid problems of synchronization in case of errors or missing responses, in which the server may have uncertainty about what transport key to use (the previous one if it was already configured or the new one).

If any of the provided value is invalid the full command shall be rejected and none of the values shall be updated.

The algorithm parameters used to encrypt/decrypt are the same as the ones described in Annex B and Annex C for signature purposes.

<u>Response message:</u>

Data Field Returned in the Response Message

'01'	4	Transaction ID. This shall be the same value as the one received in the command message. If no value was received, then no value shall be returned.	O

Table 62 Set E2E transport key data field returned in the response message

Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

In case of try to execute this command when the E2E transport key flag is activated, a general error "Conditions of use not satisfied" must be returned.

This command may either return a general error condition as listed in section 8.2.2 or one of the following error conditions.

SW1	SW2	Meaning
'65'	'BB'	Indicates INVALID_E2E_KEY_TYPE
'65'	'AC	Indicates INVALID_E2E_KEY

Table 63 Set E2E transport key processing state returned in the Response message

8.3 INT7: OTA Platform - Card

This interface is out of scope of this specification. Nevertheless, for the purpose of clarity, expectations for this interface are described in this section.

The INT7 shall rely on the secure messaging provided by INT5.

The INT7 interface shall provide the necessary functions for the Card Authentication Application issuance. This includes:

- Functions to audit the card state (not exhaustive):
  - Retrieve if the Applet package is loaded or not
  - Retrieve if the Applet is installed and selectable or not (note: the configuration of the Applet is realized using dedicated functions described in INT6)
  - Retrieve the amount of free non-volatile memory
  - Retrieve the amount of free volatile memory
- Functions to load the Applet package
- Functions to install the Applet

For an UICC platform, all these functions may be realized using GlobalPlatform commands as defined in [7].

9 Off-card Interfaces specifications

This section of the document describes the interfaces involving off-card entities. This includes the MSSP interfaces:

- INT3a: Interface between Identity Gateway and MSSP
- INT3b: Interface between MNO System and MSSP.

Others Interfaces INT1 and INT2 are addressed in separated documents.

9.1 MSSP interface description

The MSSP interface is a combination of

- The Web Service interface defined in ETSI TS 102 204 [21] for signature and registration procedures as defined in section 7. INT3a is basically a subset of this interface.

- A non-specified interface to access several additional procedures (the ones specified in section 7, except for signature and registration).INT3b includes this interface and the remaining function related with registration defined in ETSI TS 102 204 and not covered by INT3a.

This section provides recommendations or precisions regarding ETSI TS 102 204 [21] implementation. And, if not specified differently, information mentioned in ETSI TS 102 204 [21] shall apply.

The MSSP is composed of the following functions:

Function/Command	Related messages	Description
Mobile signature	SigREQ – STD SigRESP – STD	To get a mobile signature of a displayable text from an end-user
Mobile Signature status query	StatREQ – STD StatRESP – STD	To get information on a mobile signature request (executing, finished, expired...)
Mobile Signature profile	ProfREQ – STD ProfRESP – STD	To get the signature capabilities for a specified end-user.
Mobile Signature registration	RegREQ – STD RegRESP – STD	To inform the MSSP about a new end-user willing to register to the Mobile Connect Service.

Table 64 MSSP functions

The ‘Mobile Signature receipt’ function is not required in the context of Mobile Connect.

In addition, the function ‘Mobile Signature handshake’ is not required in the context of Mobile Connect.

Note: In Mobile Connect context, the entities connected to the MSSP are the Identity GW (that sits between the Application Provider and the MSSP) or the MNO. Nevertheless, the ETSI specifications use the term ‘Application Provider’ (rather than ‘Service Provider’) as the entity using the services of MSSP hence that will be retained in the following sections for consistency.

9.1.1 End-user representation in MSSP

- End-user life-cycle
- Description of the used identifiers:
  - MSISDN is mandatory; as also used as card identifier in the OTA Platform and communication address over SMS
  - Others identifiers may also be given at registration time.
- Rules when registering for signature profile:

- End-user may be registered for a single signature profile or several signature profiles at the same time; and can be registered for additional signature profile(s) at any time later.
- At the first registration the end-user shall be asked to confirm he accepts to register to Mobile Connect on its mobile (using the Mobile Connect service). This confirmation is two- fold:
    - Acts as legal acceptation signature
    - Tests the Applet and Authentication Handler are well installed and configured
- At later registration time MSSP may ask a test signature (with an appropriate message) to verify Authentication Handler is well installed and configure

9.1.2 General description of the interface based on ETSI TS 102 204

The MSSP interface, at least for the part based on ETSI TS 102 204, is specified as a set of web services. All these MSSP's functions are implemented over a simple request - response message exchange pattern through methods.

The functions are defined by the means of the Web Services Description Language WSDL Version 1.1 [22] as defined in ETSI TS 102 204 [21] clause 7

The messages to invoke web services defined in this section are bind to the network protocol HTTP & SOAP 1.2 as defined in ETSI TS 102 204 [21] clause 10.

9.1.2.1 Messaging modes for Mobile Signature request

ETSI TS 102.204 [21] defines three messaging modes for requesting a Mobile Signature function execution:

- Asynchronous server-server mode
- Asynchronous client-server mode
- Synchronous mode

In order to improve performance and avoid latency and lower resources usage (network or internal resources of the Identity GW and MNO System), it is recommended to use the asynchronous server-server mode.

If the connection from the Authentication Server (MSSP) to the Identity GW or MNO System is not possible due to deployment constraints, then the asynchronous client-server mode shall be considered / preferred.

The synchronous mode is definitively not recommended because of the asynchronous messaging that is used behind the authentication request (SMS exchange between the MSSP and the Card Authentication Applet).

Authentication server (MSSP) may support only a subset of these possible modes.

9.1.2.2 Message abstract type

This section describes all the common fields that will be systematically contained in all the messages defined for the functions of MSSP interface. This section provides addition information compared to ETSI TS 102.204 [21] section 8.1.

Common fields for a request message:

Name		Description	Mandatory
MajorVersion		Major version of the MSSP protocol as defined in ETSI TS 102 204 (and not related to the Mobile Connect version); value to set is '1'.	M
MinorVersion		Minor version of the MSSP protocol as defined in ETSI TS 102 204 (and not related to the Mobile Connect version); value to set is '1'.	M
AP_Info	AP_ID	The identifier of the Application Provider as defined in the MSSP system. This identifier is given to the AP when registered to the MSSP. For example, "http://mss.orange.com/mssURI/IDGTW"	M
	AP_TransID	A request identifier set by the AP and that the MSSP has to deliver back to the AP in the response message. This allows AP to match responses with requests.	M
	AP_PWD	Password of the AP. When the AP registers to the MSSP, he gets a password that MUST be used in all the messages in order to be authenticated.	M
	Instant	Timestamp of the message set and sent by the AP. This timestamp value shall be used to set the 'Transaction date time' (tag '02') value of the SIGN TRANSACTION command message (section 8.2.6.1)	M
	AP_URL	Contains the URL of the of the Mobile Signature Notification method of the AP. This parameter has to be set when the message mode selected by the AP is 'Asynchronous server-server'.	O
MSSP_Info	MSSP_ID	Identifier of the targeted MSSP. This specification requires that the URI shall at least be provided.	M
	Instant	Used only in case of a response	O

Table 65 MSSP request message

Common parameters for a response message:

Name		Description	Mandatory
MajorVersion		Same value as the one received in the request.	M
MinorVersion		Same value as the one received in the request.	M
Ap_Info	AP_ID	MSSP shall copy the same value as the one received in the request.	M
	AP_TransID	MSSP shall copy the same value as the one received in the request.	M
	AP_PWD	Shall be set to empty	M
	Instant	MSSP shall copy the same value as the one received in the request.	M
	AP_URL	MSSP shall copy the same value as the one received in the request, if any.	O
MSSP_Info	MSSP_ID	Identifier of the sending MSSP.	M
	Instant	The MSSP MUST mention the time and date when the message was created.	M

Table 66 MSSP response message

In the context of this specification, the AP_Info contains either the ID GW information (for a Mobile signature request message) or the information of another MNO system (for all other request types).

9.1.2.3 Guidelines for error code

This specification follows error handling defined in ETSI TS 102 204 [21] section 10.6.

The Annex H provide rules for the MSSP to map an error code coming from the Card Authentication Application into a SOAP fault sub code value.

The following table provides the list of possible SOAP Fault Sub code per function.

SOAP Fault Sub code	Raised by function	Additional comment
101 WRONG_PARAM	Mobile signature	
	Mobile Signature Status Query	No pending request matching the TransactionID
	Mobile Signature Profile Query	
	Mobile Signature Registration	
102 MISSING_PARAM	Mobile signature	
	Mobile Signature Status Query	
	Mobile Signature Profile Query	
	Mobile Signature Registration	
103 WRONG_DATA_LENGTH	Mobile signature	
104 UNAUTHORIZED_ACCESS	Mobile signature	
	Mobile Signature Status Query	
	Mobile Signature Profile Query	
	Mobile Signature Registration	
105 UNKNOWN _CLIENT	Mobile signature	The end-user is not registered yet
	Mobile Signature Profile Query	
	Mobile Signature Registration	The end-user is not a MNO subscriber
107 INAPPROPRIATE_DATA	Mobile signature	
108 INCOMPATIBLE_INTERFACE	Mobile signature	
	Mobile Signature Status Query	
	Mobile Signature Profile Query	
	Mobile Signature Registration	
109 UNSUPPORTED_PROFILE	Mobile signature	Requested signature profile is not supported by the MSSP.
208 EXPIRED_TRANSACTION	Mobile signature	
	Mobile Signature Registration	
209 OTA_ERROR	Mobile signature	
	Mobile Signature Registration	
210 UNAVAILABLE_USER _PROFILE	Mobile signature	Requested signature profile is not supported (or installed) by this end-user.
402 PC_NR_BLOCKED	Mobile signature	
406 PB_SIGNATURE_PROCESS	Mobile signature	
407 REGISTRATION_NOK	Mobile signature Registration	
410 APPLICATION_EXEC_ERROR	Mobile signature	
	Mobile Signature Registration	
411 PC_NOT_RESET	Mobile signature Registration	
504 OUTSTANDING_TRANSACTION	Mobile Signature Status Query	Request still in progress

900 INTERNAL_ERROR	Mobile signature	Any kind of error not covered by the other codes.
	Mobile Signature Status Query	
	Mobile Signature Profile Query	
	Mobile Signature Registration	

Table 67 SOAP fault error codes

9.2 INT3a: Identity GW - MSSP

This interface is defined between the Identity GW and the Authentication Server (MSSP). This interface is based on the Web Service interface defined in ETSI TS 102 204 as defined in the previous subsection.

INT3a includes the subset of functions related with Mobile Signature

9.2.1 Mobile Signature function

Provider: MSSP

Related procedure: Authentication/signature request (section 7.1)

This function corresponds to the Mobile Signature function as defined in the ETSI TS 102.204 [21] document (section 6.1).

In the context of Mobile Connect, this function allows triggering one of the end-user journeys "Click-Ok" or "Personal Code" as described in section 3. The end-user shall have been previously registered in the MSSP (see function in section 9.3.1). The Application Provider may know about end-user signature capabilities by using the function "Mobile Signature profile".

This function shall trig (if AP request is acceptable, e.g. end-user registered, request well formed...) the sending of a SIGN_TRANSACTION message (see section 8.2.6.1) to the Card Authentication Applet of the specified end-user.

9.2.1.1 Mobile signature request message (SigReq-STD)

The following table provides the parameters fields that are part of a message to request the execution of the Mobile Signature function to the MSSP.

Name	Description	MOC
MobileUser	In the context of Mobile Connect, the AP shall provide either: - A MSISDN in the international format. I.e. +33612345678. - Or a 'UserIdentifier' like 'userid42@myoperator.com'.	M
DataToBeSigned	Contains the data to be signed. This value has to be used to set the 'Message' parameter 'in the Sign transaction' function defined in 8.2.6. In the context of Mobile Connect, the MSSP is only required to support- 'MimeType' with the value 'text/plain'. The MSSP shall be able to encode the received data into a 'Data encoding schema' as required by the Card Authentication Applet. See note after this table. If 'MimeType' and 'Encoding' attributes are not specified, MSSP shall consider default values as 'text/plain' and '7bit'.	M
DataToBeDisplayed	This parameter is optional in ETSI 102 204. For Mobile Connect, this parameter is not used by the MSSP as the data to be displayed is the same as the data to be signed.	Ignored if present

SignatureProfile	This parameter is used to indicate which Level Of Assurance to apply for the signature. The possible Level Of Assurances are represented by URI and they are defined in section 9.2.3.	M
AdditionalServices	Mobile Connect doesn't mandate any added-value service. MSSP is free to define additional added-value service.	O
MSS_Format	Not used	Ignored if present
KeyReference	Not used	Ignored if present
SignatureProfileComparison	Not used	Ignored if present
ValidityDate	Expiration date of the request.	O
TimeOut	Not used	Ignored if present
MessagingMode	One of the following values: - 'asynchServerServer' for Asynchronous server-server mode (recommended) -'asynchClientServer' for Asynchronous client-server mode - 'synch' for Synchronous communication mode (shall be avoided) See section 9.1.2.1	M

Table 68 Mobile Signature request message

In order to keep the integrity of the DataToBeSigned received from the Application Provider, the MSSP shall not perform any automatic translation of the text. The Application provider is supposed to send the DataToBeSigned in the language expected by the end-user.
Similarly, the MSSP shall not perform complex encoding transformation on the DataToBeSigned. This specification requires supporting the following encoding transformations:

DataToBeSigned encoding	Resulting data Coding Scheme (DCS) in SIGN_TRANSACTION command
'7bit' (short line of US-ASCII data)	'00' (GSM default alphabet 7 bits packed including default alphabet extension table using the special escape character '0x1B')
'8bit' (short line of possible non-ASCII characters (octets with the high-order bit set)	'04': GSM default alphabet 8 bits (The final presentation to the end-user will depend on the device localisation)
'UTF-8'	'04': GSM default alphabet 8 bits, for code points between 0x00 and 0x7F
'UTF-16'	'08' (UCS-2) for code points between 0x0000 and 0xFFFF

Table 69 Encoding schemes

Additional encoding transformations are out of scope of this document.
If the MSSP receives a DataToBeSigned with an unsupported encoding, or characters out of supported code point mapping, the MSSP shall return a '107 INAPROPRIATE_DATA' error code.

9.2.1.2 Mobile signature response message (SigResp-STD)

The following table provides the fields that are part of a message that is a response of an execution request of the Mobile Signature function by the MSSP.

Name	Description	MOC
MobileUser	Copied directly from the SigReq-STD	M

Status	1) In case of synchronous mode: A valid signature will be indicated with a 502 VALID_SIGNATURE status (see ETSI 102 204 [21] Annex C). In any case an invalid signature is 503 INVALID_SIGNATURE. Any problem related to card Authentication Application like an invalid or blocked PC are returned as SOAP Fault with the appropriate sub error code (see section 9.1.2.3 and Annex H). 2) In case of asynchronous mode: Given that in this mode it's not possible to return a SOAP Fault, sub error codes (see section 9.1.2.3 and Annex H) are used directly as Status values. For example, a user cancel event would be indicated with status 401 USER_CANCEL This specification doesn't mandate any specific value for the optional fields "StatusMessage" and "StatusDetail". These fields can even be omitted.	M
SignatureProfile	The SignatureProfile that was finally used including the signature algorithm: http://uri.gsma.com/mobileconnect/LoA2/none#SCP80 http://uri.gsma.com/mobileconnect/LoA2/CBCMAC3DES http://uri.gsma.com/mobileconnect/LoA2/OATH-OCRA http://uri.gsma.com/mobileconnect/LoA3/none#SCP80 http://uri.gsma.com/mobileconnect/LoA3/CBCMAC3DES http://uri.gsma.com/mobileconnect/LoA3/OATH-OCRA	O
MSS_Signature	Mobile signature (including fields returned and signed by the Applet) computed and returned by the Card Authentication Application.	O
MSSP_TransID	Transaction number created by the MSSP for this transaction. The MSSP is free to generate the MSSP_TransID with the value it wants (content, length...). Only constraint is to be unique. The MSSP shall ensure the traceability with the value put in the 'TransactionID' field of the related SIGN_TRANSACTION command sent to the Card Authentication Application. This field shall be present when the request is valid and accepted by the MSSP. In case of rejected request, this field shall contain a "null" string value in the response.	C

Table 70 Mobile Signature response messages

9.2.2 Mobile Signature Status Query function

Provider: MSSP

Related procedure: Authentication/signature request (section 7.1), registration request (section 7.2)

This function corresponds to the Mobile Signature Status Query function as defined in the ETSI TS 102.204 [21] document (section 6.2).

In addition, this function can also be used to retrieve the status of a Mobile Registration function which may, as for Mobile Signature request, be executed in an asynchronous mode.

9.2.3 Mobile Signature Profile Query function

Provider: MSSP

Related procedure: Authentication/signature request (section0)

This function corresponds to the Mobile Signature Profile Query function as defined in the ETSI TS 102.204 [21] document (section 6.3).

This function can be used to retrieve the signature/authentication capabilities that are currently supported by the specified end-user, meaning already installed and that an AP can use for a further Mobile Signature request (SigReq-STD).

Here is the list of Mobile Signature profiles (as an URI) supported in Mobile Connect:

- http://uri.gsma.com/mobileconnect/LoA2: matches the "Click-Ok" end-user journey, Level Of Assurance 2
- http://uri.gsma.com/mobileconnect/LoA3: matches the "Personal Code" end-user journey, Level Of Assurance 3

9.2.3.1 Mobile signature Profile Query request message (ProfReq-STD)

The following table provides the parameters fields that are part of a message to request the execution of the Mobile signature Profile Query function to the MSSP.

Name	Description	MOC
MobileUser	See Mobile Signature request message.	M

Table 71 Mobile Signature profile query request message [profReq-STD]

9.2.3.2 Mobile signature Profile Query response message (ProfResp-STD)

The following table provides the fields that are part of a message that is a response of an execution the Mobile signature Profile Query function by the MSSP.

Name	Description	MOC
MobileSignatureProfile	A list of profiles supported currently by the end-user (possible profiles are defined in 0)	M
Status	Current Status of the transaction	M

Table 72 Mobile Signature Profile Query response message [ProfResp-STD]

9.3 INT3b: MNO - MSSP

This interface is defined between the MNO System and the Authentication Server (MSSP). This interface has two clearly differentiated parts:

- One based on the Web Service interface defined in ETSI TS 102 204 as defined in the previous subsection, specifically the function related with Mobile Signature

- Other part which is a non-specified interface to access several additional procedures (all the specified procedures in section 7 except signature and registration).

Only the first part will be described in this document, the second part is matter of internal decision by each MNO.

9.3.1 Mobile Signature Registration function

Provider: MSSP

Related procedure: Registering an end-user (section 7.2), Un-registering an end-user (section7.2), Changing card (section 7.4), Reset Personal Code (section 7.5), Unblock Personal Code (section7.6)

This function corresponds to the Mobile Signature registration function as defined in the ETSI TS 102.204 [21] document (section 6.4).

In the context of Mobile Connect, this function is also used to provide the following features:

- Unregister and end-user. In this mode the function may either only deactivate the Authentication/Signature capability of the end-user, or delete the Authentication/Signature capability of the end-user (by removing the application on the card , and removing any reference of the end user in the MSSP)
- Reset PC. In this mode the function allows to send a message to the Card Authentication Application in order to reset the Personal Code of the end-user (see section 3.5).
- Unblock PC. In this mode the function allows to send a message to the Card Authentication Application in order to unblock the Personal Code of the end-user (see section 3.4).

This function has to perform OTA operations in most of the cases. So unlikely to ETSI TS 102 204, this function may also be executed in asynchronous way. This function shall support to be executed in the messaging modes described in section 9.1.2.1.

9.3.1.1 Mobile signature registration request message (RegReq-STD)

The following table provides the parameters fields that are part of a message to request the execution of the end-user registration (or un-registration, reset Personal Counter or unblock Personal Counter) to the MSSP.

Name	Description	MOC
MobileUser	See Mobile Signature request message. The MSISDN is mandatory when used in "Register mode".	M
EncryptedData	Not used	Ignored if present
EncryptResponseBy	Not used	Ignored if present
CertificateURI	Not used	Ignored if present
X509Certificate	Not used	Ignored if present
Any	MessagingMode: one of the messaging modes. See description of the parameter in section 9.2.1.1. This parameter is optional; if not provided the MSSP shall execute the request in synchronous mode (behaviour defined in ETSI TS 102 204). If not supported, an error shall be returned.	O
	Mobile Connect uses this field to instruct MSSP about the execution mode: - **Register mode**: to register a new end-user, or an existing end-user with a new Mobile signature profile - **Un-register mode**: to unregister a previously registered end-user. - Reset Personal Code mode - Unblock Personal Code mode	M

	Each mode has its own additional parameters (see tables below)[2]	

Table 73 Mobile Signature registration request message [ReqReg-STD]

Additional fields for "**Register mode**":

Name	Description	MOC
Mobile signature profile	The mobile signature profile (Level of Assurance) for which the end-user has to be registered. The possible Level of Assurances are represented by URI and they are defined in section 9.2.3.	M
Message	A message to be displayed to end-user to acknowledge or reject Mobile Connect subscription. This field is optional, if not provided the MSSP shall consider a default value (according to preferred language).	O
Preferred language	Preferred language of the end-user. This can be used to configure messages handled by the Applet. This field is optional, if not provided the MSSP shall consider a default value.	O

Table 74 Additional fields for Register mode

Additional fields for "Un-register mode", "Reset Personal Code mode", "Unblock Personal Code mode":

None

The schema extension for Mobile Connect is defined in Annex I.

9.3.1.2 Mobile signature registration response message (RegResp-STD)

The following table provides the fields that are part of a message that is a response of the end-user registration (or un-registration, or reset Personal Counter or unblock Personal Counter) by the MSSP.

Name	Description	MOC
Status	Current Status of the transaction (see Annex B). If the registration is successfully performed, the MSSP MUST indicate the status code 408 REGISTRATION_OK, whatever the requested mode (Register end-user, Un-register end-user, Reset Personal Code, Unblock Personal Code, Change Card). In case of 'MessagingMode' asynchronous, if the registration request is accepted, the MSSP shall indicate a status code 100 REQUEST_OK, with 'StatusDetail' element containing the 'MSSP_TransID' generated by the MSSP.	M

[2] More information will be added with Change MSISDN, plus additional provisioning use cases coming from MNO's provisioning system.

	This 'MSSP_TransID' can be used further by the AP, within the 'StatREQ – STD', to get the status of the registration request.	
EncryptedData	Not used	Ignored if present
EncryptResponseBy	Not used	Ignored if present
CertificateURI	Not used	Ignored if present
X509Certificate	Not used	Ignored if present

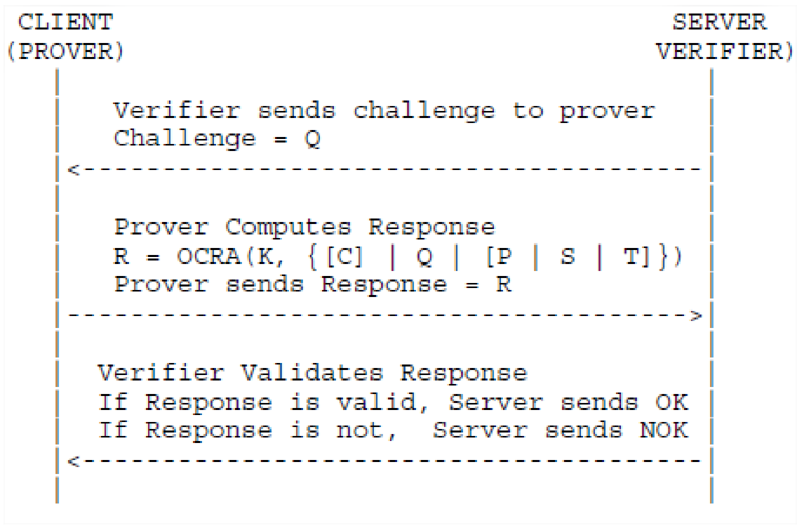
Table 75 Mobile signature registration response message [RegResp-STD]

9.3.2 Other functions

All the other functions of INT3b are an internal implementation for each MNO.

Annex A (Normative) OATH OCRA: indication for One-Way Challenge-Response computation

The one-way challenge-response is defined in RFC 6287 [2] and represented in the following sequence flow.



```
 CLIENT                                   SERVER
(PROVER)                                 VERIFIER)
   |                                        |
   |   Verifier sends challenge to prover   |
   |   Challenge = Q                        |
   |<---------------------------------------|
   |                                        |
   |   Prover Computes Response             |
   |   R = OCRA(K, {[C] | Q | [P | S | T]}) |
   |   Prover sends Response = R            |
   |--------------------------------------->|
   |                                        |
   |  Verifier Validates Response           |
   |  If Response is valid, Server sends OK |
   |  If Response is not,  Server sends NOK |
   |<---------------------------------------|
   |                                        |
```

Figure 25 : one-way challenge response

The ‘Server (verifier)’ is identified to the MSSP.

The ‘Client (prover)’is identified to the Card Authentication Applet.

The first step matches the command message sent by the MSSP to the Card Authentication Applet, and the second step matches the response message sent by the Card Authentication Applet (see section 8.2.6.1).

The last step is not required in the context of this specification.

The Challenge (Q) shall be the result of the concatenation of the ‘Transaction ID’ value (value of the data field indentified by the tag ‘01’ of the SIGN TRANSACTION command), the ‘Transaction date time’ value (value of the data field ‘02’ of the SIGN TRANSACTION command), the ‘end-user response status’ value (value of the data field ‘10’ of the SIGN TRANSACTION command), and the ‘Message’ value (value of the data field ‘8D’, including the DCS byte, of the SIGN TRANSACTION command) in the following order:

Q = TransactionID (4 bytes) || TransactionDateTime (4 bytes) || Auth Handler type (1 bytes) || SHA1(Message) (20 bytes) || 00…(12 bytes)

Convention: x || y Concatenation.

Note: As defined in RFC 6287, Q as to be padded with zeroes to the right up to 128 bytes.

The Key (K) matches the Authentication key of the Authentication Handler.

The Counter (C) matches the counter of the Authentication Handler.

The P, S and T parameters are **not** used in the context of this specification. So the DataInput value shall be:

DataInput = {OCRASuite | 00 | C | Q).

The OCRASuite value to use shall be OCRA-1:HOTP-SHA1-8:C-QH41, and shall be ASCII encoded when included in the DataInput computation.
Meaning that Challenge(Q)is provided as an hexadecimal value of 41 bytes (using this length will allow to move to SHA256, while removing the 12 padding bytes, without changing the OCRASuite value). The CryptoFunction to use is the HOTP-SHA1-8, indicating an 8-digit dynamic Truncation. Further release of this specification may introduce additional crypto functions.

The R shall match the Signature of the response message, returned as 4 bytes containing the 8 digits BCD coded.

Example of signature:

Message =0x0474657374206461746120626569 6E67207369676E6564

Representing:

Data coding schema=0x04 (GSM default alphabet 8 bits)

Text=test data being signed

TransactionID =0xB6F18CBB

TransactionTime =0x543FF588

AuthenticationHandlerTypeTag=0xB6

sha1(hexStringToByteArray(Message)) => 0x39393C6B15F2F11BFFF018AE61EB6C0EC602CE08

Q=
0xB6F18CBB543FF588B639393C6B15F2F11BFFF018AE61EB6C0EC602CE0800000000000000000000000000

OCRASuite=OCRA-1:HOTP-SHA1-8:C-QH41

Authentication key value (Seed) =0x3132333435363738393031323334353637383930

Counter value=0x26

Computed R is 0x00A3300E

Annex B (Normative) MAC computation for 3DES based Authentication Handlers

Block cipher: 3DES double key (ISO/IEC 18033-3 or NIST SP-800-67)

MAC computation process:

- Cryptographic function : 3DES-CBC-MAC (ISO/IEC 9797-1:2011 MAC Algorithm 1 or ANSI X9.19)
- Key length : 128 bits (2 keys, minimum) or 192 (3 keys)
- Initial vector : 00h
- Padding : ISO/IEC 9797 padding method 2 (bytes equal to '80 00 00...')
  - In case the length of the message to sign is a multiple of 8bytes, a complete byte of padding must be added
- Output : 64 bits

Data field processing (MO)

Signature should be applied to TransactionID, TransactionDateTime, End User Response Status APDU and the Message (including the Data coding scheme byte and excluding the tag and length bytes) in the following order:

M = TransactionID (4 bytes) || TransactionDateTime (4 bytes) || Auth Handler type (1 bytes)|| Message

Annex C (Normative) MAC computation for AES based Authentication Handlers

This section provides indication for the Card Authentication Applet to compute, and for the MSSP to verify, the Message Authentication Code (MAC) returned in the function defined in section 8.2.6.1, in the case of Authentication Handler '"Click OK" with authentication based on AES-CMAC' or '"Personal Code" with authentication based on AES-CMAC'.

The Message Authentication Code (MAC) is an AES-CMAC as defined in RFC 4493 [25].

The Key (K), 128-bit (16-octet) long key for AES-128, matches the Authentication key of the Authentication Handler.

The message M of the AES-CMAC algorithm shall be initialized as the concatenation of the 'Transaction ID' (tag '01'), the 'Transaction date time' (tag '02'), the 'end-user response status' (tag '10') and the 'Message' (tag '8D') (including the Data coding scheme byte) in the following order:

M = TransactionID (4 bytes) || TransactionDateTime (4 bytes) || Auth Handler type (1 bytes)|| Message

Convention: x || y Concatenation.

Annex D [Future Option, added for reference for future implementation of Device App + SIM Applet hybrid Authenticator] Mobile Device – Card Authentication Application

The user interface for the user transaction confirmation can be implemented in a mobile device application or by the Card Authentication Applet itself using the STK proactive commands: DISPLAY TEXT + GET INPUT.

Using STK for user interactions implies some limitations:

- Some mobile devices do not support STK proactive commands: DISPLAY TEXT + GET INPUT
- STK provides only a very simple user interface design.

The realization of the user interface in a mobile application might have several advantages:

- The user confirmation can be integrated in the business logic of the mobile application
- The user interface can be individually designed in a mobile application
- The interface between the mobile device and the Card Authentication Application can be additionally secured

For a mobile application based user interface the Card Authentication Application has to provide a set of APDU commands, which can be used by the mobile application to perform the user interactions. Moreover the mobile device platform has to support a Secure Element API that allows an APDU communication from mobile applications to the Card Authentication Application.

Command description

The command is applicable whatever Card Authentication Application implementation (3DES-CBC/AES-CBC, OATH-HOTH/OCRA)

Command message GET REQUEST:

Once the Card Authentication Application has received an authentication request from the server it has an authentication request state set. If this authentication request state is set GET REQUEST returns the required information to perform the user confirmation. After calling GET REQUEST command the authentication request state is reset and the user confirmation information is not available any more.

Note: Since a mobile device application does not know the exact time when this authentication request will be received by the Card Authentication Application it has to use the command GET REQUEST in a loop. This loop might be stopped after a certain time period. This behaviour might depend on the individual mobile application implementation and is out of scope for this specification.

The GET REQUEST command message is coded according to the following table:

Code	Value	Meaning
CLA	‘80’	
INS	‘2A’	GET REQUEST
P2	‘00’	
P2	‘00’	

Table 76 Hybrid Get request codes

Response message:

The data field is only available if the Card Authentication Application has received an authentication request from the server and its authentication request state is set. Otherwise the Card Authentication Application will only return status bytes '90' '00' without data field.

Data Field Returned in the Response Message

Tag	Length	Value	MOC
‘01’	1	Personal code verification mode: ‘00’: no Personal code verification ‘01’: one-step user journey ‘02’: two-steps user journey	M
‘02’	0-XX	Message to be displayed	M

Table 77 Hybrid Data field returned in the response

Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may either return a general error condition as listed in section 6.5.2 or one of the following error conditions.

SW1	SW2	Meaning
		To be completed

Table 78 Hybridge processing state returned in the response message

<u>Command message VERIFY PIN:</u>

Once the Card Authentication Application has the authentication request state the mobile application can request a user confirmation. For performing a PIN verification the mobile application has to send a verify PIN command to the Card Authentication Application. If PIN verification is not required (i.e. the user has just to confirm the transaction) the data field shall be kept empty and LC shall be set to 0.

The VERIFY PIN command message is coded according to the following table:

Code	Value	Meaning
CLA	‘0X’	See ISO/IEC 78116-4 [9]
INS	‘20’	VERIFY PIN
P1	‘00’	
P2	‘00’	
Lc	‘XX’	Length of the data field
Data	‘XXX...’	PIN value

Table 79 Verify PIN command message

Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may either return a general error condition as listed in section 6.5.2 or one of the following error conditions.

SW1	SW2	Meaning
63	CX	Authentication failed (x=current retry counter)

69	83	Too many retries. PIN blocked.

Table 80 Verify PIN processing State returned in the response message

Annex E Packaging Format

The packaging format for the SIM applet is proposed to be "cap".

Annex F SIM Applet Profiles

F.1 Simple Profile

Scope:

For deployment OTA to existing SIMs; applet functionality will by necessity be lowest common denominator in order to minimise applet size (6KB) which directly impacts on size of addressable base (e.g., those SIMs in the field with sufficient space to download, unpack and install the applet) and likelihood of success in downloading the applet OTA (larger applets requiring many more SMS messages to be successfully received)

Feature	Description
Multi Language Support	• Basic • Language per applet instance • Language specified before the applet is added to the SIM
LoA supported	• LoA 2 • LoA 3
Encryption Algorithm	• Same algorithm for LoA 2, 3 • 3DES with minimum 2 keys (112 bits), 3 keys (168bit) also possible • Or… OATH OCRA/HOTP (160bits) • Or… AES (128bits)
SIM capability	• No SMS concatenation needed (MO and MT)
Applet size	• Max 6KB
Authentication Handler Selection	• Pre selecting one and configuring the MSSP/OTA for the same end to end
Applet Deployment Mode	• Applet deployed in ISD using OTA keys (guideline)

Table 81 Simple Profile

F.2 Enhanced Profile

Scope:

- For pre-embedding in new SIMs and/or download to existing Enhanced/NFC SIMs
- Will support higher levels of security which will increase applet size and may preclude OTA delivery to basic legacy SIMs but shouldn't impact on SIM product selection (memory size)

Feature	Description
Multi Language Support	• Basic • Language per applet instance • Language specified before the applet is added to the SIM
LoA supported	• LoA 2 • LoA 3
Encryption Algorithm	• AES – 128 bits • OATH OCRA/HOTP – 160 bits
SIM capability	• SMS concatenation (MO)
Applet size	• Target ~7-10KB (depending on number of algorithms supported)
Authentication Handler Selection	• Authentication Handler OTA Discovery
Applet Deployment Mode	• Applet deployed in ISD using OTA keys (guideline); support for SSD needed for some SPs (Banks, Gov)

Table 82 Enhanced Profile

Annex G Open questions

(1) Problem of spam: how to avoid that a third party uses a known MSISDN to generate unwilling authentication requests?

Some suggestions:

- White list of 3rd parties
- Throttling and duplicate request detection at the ID GW using the Message Level Security layer at the ID GW and also using the Business Activity Monitoring component in the ID GW. The ID GW should detect abnormal patterns (e.g. multiple requests from same SP, for same MSISDN in a configurable short interval of time).

Other options that can be considered are:

- An alias is created during registration and that’s used for use cases where MSISDN needs to be asked from the user. The registered and validated MSISDN is the retrieved from the alias.

- Location can be used as one of verification factor [asked during authentication] and compared with the device location
- Some device characteristics can be asked, e.g. Which device make/model and then comparing with the attached device in the HLR

(2) End-user experience with Personal code when:

- the end-user wants to reject an authentication request, but he is still required to enter its PC for the Applet to be able to generate the response
    - Some suggestions:
        - When rejected, the applet is able to sign the request with REJECTED or EXPIRED without introducing the PC
- the MSSP wants to present an information that is quite confidential and so wants to authenticate the end-user before displaying the message: so PC would be required first before any other operation
    - Some suggestions:
        - Both, confidential and long information should be presented in a PC and only the hash must be sent to be signed
- Some devices provide a different end-user experience for the same STK command; Some don't display the cancel button(it has to be retrieved in a different way in a contextual menu depending on the OS)

(3) Shall we move to 3DES-CBC three keys (168 bits)?

- Some suggestions:
    - According to NIST SP 800-57 this will depend on how these keys will be deployed. However, as a rule, we must use the most secure way to use the algorithms available.

(4) SCP81 HTTPs connection:

How to configure the applet to define the default bearer between SMS and HTTPs?

How to configure the SCP81 connectivity parameters in the application?

Do we need to rely only on the Admin Agent retry mechanism or we prefer have a dedicated applicative retry?

Do we need to the default bearer SMS (if configured) if N (a parameter of the application) tentative of connection failed?

Globally, for SMS and HTTPs, how we will manage the error message to the user if the response cannot be sent (the user previously signed the message, he is waiting for the end of the transaction)?

Annex H (Normative) Card Authentication Application error code mapping

On reception of Card Authentication Application error code, the MSSP may have to return an error code to the requesting AP.

The table below provides the mapping that the MSSP shall use with these general principles:

- Errors returned by the MSSP that are linked to an Applet error are all in range 4xx (to avoid confusion with other root cause)

- MSSP shall return a 4xx code only when MSSP is not really to handle it, i.e. the MSSP shall try any corrective action or retry before returning a final error to AP

- Errors returned to AP are detailed when linked to an end-user experience (e.g. SIGN_TRANSACTION) or when the information may be valuable; else it should remain hidden to the AP by the MSSP

Type	Status Word returned by applet	Usage & Meaning of the error	Status Code returned to AP	Comment
General problems	64 00	Generic: (Execution error) No specific diagnosis	410 APPLICATION_EXEC_ERROR	
APDU problems (APDU badly formatted, missing information…)	67 00	Generic: (Checking error) Wrong length in Lc	410 APPLICATION_EXEC_ERROR	(1) Such error codes returned by the applet are the result from a bad behaviour from the MSSP. MSSP shall handle internally these codes. AP may finally receive this code only when MSSP is not able to handle the returned applet error.
	6A 86	Generic: (Checking error) Incorrect P1 P2		
	6D 00	Generic: (Checking error) Invalid instruction		
	69 84	Generic: (Checking error) Invalid data		
		CHANGE_STATUS: indicates MISSING_DATA		
	6A 80	Generic: (Checking error) Incorrect data		
Invalid data	65 A3	PUT_DATA: indicates INVALID_INSTALL_DATE	410 APPLICATION_EXEC_ERROR or 407 REGISTRATION_NOK	Same comment as (1). Or return a 407 error when applet error, that cannot be handled by the MSSP, happens in a registration process
	65 A4	PUT_DATA: indicates INVALID_MAX_PC_ATTEMPT		
	65 A5	PUT_DATA: indicates INVALID_PC_LENGTH		
	65 A6	PUT_DATA: indicates INVALID_MSSP_ADDRESS		
	65 BB	PUT_DATA: indicates INVALID_AUTH_HANLDER_TYPE		
	65 AA	PUT_DATA: indicates INVALID_AUTH_HANLDER_ID		
		CHANGE_STATUS: indicates INVALID_AUTH_HANLDER_ID		
		DELETE: indicates INVALID_AUTH_HANLDER_ID		
	65 AC	PUT_DATA: indicates INVALID_AUTH_KEY		
	65 AD	PUT_DATA: indicates INVALID_COUNTER		
Personal Code problems	69 90	SIGN_TRANSACTION: Indicates PC_CODE_BLOCKED	402 PC_NR_BLOCKED	
	69 91	SIGN_TRANSACTION: Indicates PC_CODE_NOT_CHANGED	406 PB_SIGNATURE_PROCESS	
	65 03	SIGN_TRANSACTION: Indicates SIGNATURE_TRANSACTION_REJECTED	401 USER_CANCEL	
	65 04	SIGN_TRANSACTION: Indicates SIGNATURE_TRANSACTION_EXPIRED	208 EXPIRED_TRANSACTION	
	65 02	MANAGE_PC: indicates PCODE_NOT_RESET	411 PC_NOT_RESET	
Service/Command Problems	69 85	Generic: (Checking error) Conditions of use not satisfied	410 APPLICATION_EXEC_ERROR	Same comment as (1)

		SIGN_TRANSACTION: indicates SERVICE_NOT_OPERATIONAL	406 PB_SIGNATURE_PROCESS	
		MANAGE_PC: indicates SERVICE_NOT_OPERATIONAL	407 REGISTRATION_NOK	
	62 01	MANAGE_PC: indicates PC_CODE_NOT_BLOCKED	NA	No error shall be returned: PC was not blocked when command was received
	69 86	Generic: (Checking error) Command not allowed	410 APPLICATION_EXEC_ERROR	Same comment as (1)
		SIGN_TRANSACTION: indicates APPLET_NOT_OPERATIONAL	406 PB_SIGNATURE_PROCESS	
	6A 81	Generic: (Checking error) Function not supported	410 APPLICATION_EXEC_ERROR	Same comment as (1)
Security problem	69 82	Generic: (Checking error) Security Status not satisfied	410 APPLICATION_EXEC_ERROR	Same comment as (1)

Table 83 Card Authentication application error code mapping

Annex I (Normative) Schema for Mobile Connect

The Mobile Connect elements are all defined in the namespace "http://namespaces.gsma.org/mobile-connect/1.1"

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema targetNamespace="http://namespaces.gsma.org/mobile-connect/1.1"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

    xmlns:mc="http://namespaces.gsma.org/mobile-connect/1.1"

    xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#"

    elementFormDefault="qualified">



    <xs:import                          namespace="http://uri.etsi.org/TS102204/v1.1.2#"
schemaLocation="Schema_102204.xsd"/>



    <xs:element name="Register">

        <xs:annotation>

            <xs:documentation>
```

```
                This element shall be part of the MSS_RegistrationReq (as allowed by the
#any extension) to provide the additional data required for the registration mode.

            </xs:documentation>

        </xs:annotation>

        <xs:complexType>

            <xs:sequence>

                <xs:element name="MobileSignatureProfile" type="mss:mssURIType"/>

                <xs:element name="Message" type="mss:DataType" minOccurs="0"/>

                <xs:element name="PreferredLanguage" type="xs:language" minOccurs="0"/>

            </xs:sequence>

            <xs:attribute        name="MessagingMode"        type="mss:MessagingModeType"
use="optional" />

        </xs:complexType>

    </xs:element>



    <xs:element name="Unregister">

        <xs:complexType>

            <xs:attribute        name="MessagingMode"        type="mss:MessagingModeType"
use="optional" />

        </xs:complexType>

    </xs:element>



    <xs:element name="ResetPC">

        <xs:complexType>

            <xs:attribute        name="MessagingMode"        type="mss:MessagingModeType"
use="optional" />

        </xs:complexType>

    </xs:element>



    <xs:element name="UnblockPC">

        <xs:complexType>

            <xs:attribute        name="MessagingMode"        type="mss:MessagingModeType"
use="optional" />

        </xs:complexType>

    </xs:element>



    <xs:element name="RegistrationStatusDetail">
```

```
        <xs:complexType>

            <xs:attribute name="MSSP_TransID" type="xs:NCName" use="optional"/>

        </xs:complexType>

    </xs:element>


</xs:schema>
```

Annex J Document history

Version	Date	Brief Description of Change	Approval Authority	Editor/Company
0.1	01/01/2014	Initial Draft	David Pollington	SIM EVO WG
0.8	19/03/2014	First tracked edition: - Remove sections not relevant into this document - Include card Platform section describing UICC and SIM differences. - Add devices requirements - Add definition of functions in INT5	David Pollington	SIM EVO WG
0.8.1	25/03/2014	- End user journey: change part dealing with time management as Applet alone cannot have a fine grain time management. - Add notion of Authentication Handler, including change on most of the INT6 functions - Remove INT6.'Get authentication data' which functional scope is covered by INT6.'Sign transaction' - Add INT6."Delete Authentication Handler' - Add INT7 section to list the expected functions, even if interface is out of scope - Add end-user response status in the SIGN_TRANSACTION response	David Pollington	SIM EVO WG
0.8.2	25/03/2014	- Include comments following internal review between Gemalto and Orange.	David Pollington	SIM EVO WG
1.1	31/03/2014	This release merges GSMA CPAS8 SIM Applet Authentication Specification v1.0 and former 0.8.2. And additionally: - Include comments from Morpho - Add a general architecture overview section focused on Mobile Connect with the Card allowing to introduce interfaces - Change presentation of section "Applet deployment and card architecture options" to address the impacts on back-end more clearly. Add a recommendation section. - Detail the coding of the message in the SIGN_TRANSACTION command to include UCS2 (China Mobile comment) - Detail the coding of the OATH OCRA implementation for function SIGN_TRANSACTION - Update of section "Later phase"	David Pollington	SIM EVO WG
1.2	31/03/2014	- Include comments following internal review between Gemalto and Orange (31/03/2014).	David Pollington	SIM EVO WG
1.3	03/04/2014	Version has the result of the meeting in La Ciotat 03/04/2014	David Pollington	SIM EVO WG

Version	Date	Brief Description of Change	Approval Authority	Editor/Company
		- Include update coming from a version 1.1 of the document issued by GSMA as a 'branch' - Add description of the end-user journeys 'Wrong Personal Code Journey /unblock', 'Reset/Change Personal Code Journey', 'Change smart card within the same MNO', 'MNO change', 'Unsubscribe Mobile Connect' - Add section '7 Detailed procedures' - Include INT3 description in the scope of the document - Add Annex B 3DES-CBC MAC computation		
1.4	04/04/2014	- Cleaning - Review of the figure in section 4, to remove link between IDGW and OTA Platform, as it has been decided that the MSSP will be in charge of the download and install of the Applet. - Fill functions in section 9 (INT3)	David Pollington	SIM EVO WG
1.5	15/04/2014	- Include comments from GSMA based on v1.3 (mainly change term 'Identity Provider' into 'ID GW', change term 'Authentication Service Provider' into 'Authentication Server (MSSP, and add support of AES) - Add textual description for some detailed procedures	David Pollington	SIM EVO WG
1.8	23/05/2014	Updates based on the feedback from the operators and also from the review in the SIM Vendors workshop on the 19/5/2014. Some key updates: • Added the Overlapping Authentication Request scenario • Multi language support updates • Annex F updated to update the answers • Annex D added – for future use only • Annex A updated to use the values only for the calculation of Q	David Pollington	SIM EVO WG
1.9	30/06/2014	• Minor fixes • Description added for the Remote APDU format to use for sending commands and responses • Description added for the specific error codes • Added Profiles • Added packaging proposal	David Pollington	SIM EVO WG
1.10	30/09/2014	- Add complementary information for OATH OCRA computation (Annex A)	David Pollington	SIM EVO WG

Version	Date	Brief Description of Change	Approval Authority	Editor/Company
		- Add description on how to return command response as a SM MO in the case of handling of proactive session. - Add data field 'MSSP address' to be managed by Card Authentication Applet. This field is optional to ensure backward compatibility of the specification. - Add optional MessagingMode parameter in INT3.RegReq-STD (registration request) to be able to handle asynchronous mode required by OTA communication during the execution of this function. And add the MSSP_TransID in the response - Change the 'Unblock_PC' mode of the MANAGE_PC command to optional. This may be used to reduce the code size of the Card Authentication Application, without loss off functionality as this mode is covered by the more generic 'RESET_PC' mode - Add information on the processing of the 'Set Applet data' function - Add missing parameter 'timeout' in command 'PUT_DATA' and 'GET_DATA' (was an inconstancy in the document as the data is described in section 6.6) - Add transactionID on commands MANAGE_PC, GET_DATA, SET DATA, PUT_DATA, CHANGE_STATUS, DELETE - Specify encoding transformation at MSSP level for DataToBeSigned - Add section on error code for the interface INT3 - Add error code mapping annex - Add information on message encoding processing in the SigReq-STD - Simplify the 'Set Authentication Handler data' function for OATH-OCRA handlers. No difference between create and update mode to limit the code size. Both key and counter have to be provided. - Add End to End Transport Keys		
2.0	18/10/2014	- Add details on error code returned by the Applet - Enhance the level of information returned by the 'End-userstatus' byte in the SIGN_TRANSACTION command - Add changes in MSSP error code (section 9.1.2.3) - Split INT3 into INT3a and INT3b - Fixed some issues on the command messages - Updated the detailed procedures to use the MNO systems and keep ID GW for Authentication	David Pollington	SIM EVO WG

Version	Date	Brief Description of Change	Approval Authority	Editor/Company
		- Editorial fixes		
2.0.1	20/10/2014	- Remove MSSP error code 211 REQUEST_NOT_COMPLETED, Use 504 OUTSTANDING_TRANSACTION - Add signature sample in Annex B OCRA-OATH - Editorial Fixes	David Pollington	SIM EVO WG
2.0.2		- Clarification on the version management related to the version returned by the applet (section 6.6.2) - Alignment of section 6.6.5 for 'Max Personal Code attempts' management - Update in the MSSP Address format. - Update in the Time out format - Add directives in TLV management - SIGN_TRANSACTION: minor fix. - Change Applet status: add additional explanation on command management - Clarification in Table 25 (indentation to figure tag inclusion). Add a tag for proprietary features. - Update on the way the 'PC length' shall be managed in the 'Set Applet data' function - Clarification on the MSSP_TransID in the response of the Mobile Signature function (section 9.2.1.2) - Add indentation in table 35 to 42 to figure tag inclusion. - Table 64: add additional management description on the MSSP_TransID field. - Add schema for MSSP - Added section "6.6.11 E2E transport key type" - Remove the requirements to activate the applet in sections "6.6.3 Activation Flag" and "8.2.10 Change Applet Status" - 8.2.5 Sign transaction, two clarifications added. - Editorial fixes.	David Pollington	SIM EVO WG
2.1.0	31/03/2015	- Change some requisites to set PC parameters before authentication handlers requiring it have been activated instead of before have been created - Time Out field removed due to inconsistent Sim Toolkit behavior which may introduce potential risks for ClickOK authenticators. - Added new fields Applet Release and Applet Locale - Added new section "8.2.3 General user interaction considerations" - Edited section "8.2.4 Remote APDU format" to changes the rules allowing	David Pollington	SIM EVO WG

Version	Date	Brief Description of Change	Approval Authority	Editor/Company
		commands requiring user interaction to be included in a list with other commands and to simplify the processing of lists of commands. - Removed from Sign Transaction the parameter “End user response status” due to the difficulties that most people have to understand its description and that it doesn’t provide additional information. Instead it has been added the Authentication handler type to indicate which kind of authenticator has been used to sign the message. - Get Applet Data operation has been modified to include a P1 parameter which allow to recover different parts of the applet data. The response of the command has been edited to remove or add fields, and to change the MOC value for some fields which weren’t correct. - Set Authentication Handler data section: Introduced an error to indicate that the maximum number of authenticators supported by the applet has been reached. - Edited section “9.2.1.2 Mobile signature response message (SigResp-STD)” to describe the Status according to the current state of the document. - Edited Annex A, B and C according to the changes for Sign Transaction command. - Eddited Annex I table to define specific status codes for SIGNATURE_TRANSACTION_REJECTED and SIGNATURE_TRANSACTION_EXPIRED errors.		
2.1.1	14/05/2015	- Clarifications regarding behavior in case of errors about wrong length in “8.2.2 General error conditions” section - Editorial fixes. - Usage of PoR in an SMS exchange was changed from mandatory to optional because the SCP80 session is secured anyway and the support of PoR implies additional code size.	David Pollington	SIM EVO WG
2.1.2	07/10/2016	- Clarification added to section 8.2.4 regarding the use of the GET RESPONSE command.	David Pollington	Pablo / Telefonica
2.1.3	13/10/2016	Edition of section 8.2.6.1 : Interoperability recommendations updated to : “Applet should be able to process the APDU payload of 2 concatenated SMSs”, and “The encoded text to be displayed should not exceed 220 bytes”	David Pollington	Matthieu Verdier / Orange

Version	Date	Brief Description of Change	Approval Authority	Editor/Company
2.1.4	21/10/2016	Edition of section 8.2.6.1 Command description and creation of 8.2.6.2 Interoperability issues and 8.2.6.3 Command response processing : Main addition is a section concerning impacts on MNOs using v2.1.1 and earlier versions of this specification .	David Pollington	Pablo[Telefonica] / Matthieu Verdier [Orange]
2.1.5	21/10/2016	Creation of 8.1.4 INT5b: SCP81 (RAMoHTTP). Support of SCP81 channel between the UICC and OTA as described in GlobalPlatform Amendment B [27]. Update of 8.2 INT6: Authentication Server (MSSP) – Card Authentication Application and 8.2.4 Remote APDU Format	David Pollington	Alexander S/ Matthieu Verdier
2.1.6	23/10/2016	Converted to PRD format.	David Pollington	Siva (v Boyalakuntla) / GSMA
2.1.7	24/10/2016	Amended missing information related to backward / interoperability from previous documents.	David Pollington	Siva (V Boyalakuntla) / GSMA
2.2.1	18/07/2019	Edotirial updates and updated references as a result of document refresh and transition of MC to BAU	David Pollington	Nick Spencer
2.2.1	06/12/2022	Go through TG approval	TG	Yolanda Sanz/GSMA

Table 84 Document History

J.1 Other Information

Type	Description
Document Owner	IDG
Editor / Company	Yolanda Sanz, GSMA

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

Your comments or suggestions & questions are always welcome.