TRUSTED CONNECTIVITY ALLIANCE

Mobile Connect SIM Applet Interoperability Stepping Stones

Published by simalliance now Trusted Connectivity Alliance

December 2016

Intellectual Property Rights (IPR) Disclaimer

Attention is drawn to the possibility that some of the elements of any material available for download from the specification pages on Trusted Connectivity Alliance's website may be the subject of Intellectual Property Rights (IPR) of third parties, some, but not all, of which are identified below.

Trusted Connectivity Alliance shall not be held responsible for identifying any or all such IPR, and has made no inquiry into the possible existence of any such IPR. TRUSTED CONNECTIVITY ALLIANCE SPECIFICATIONS ARE OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NON-INFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY IMPLEMENTATION OF ANY TRUSTED CONNECTIVITY ALLIANCE SPECIFICATION SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER TRUSTED CONNECTIVITY ALLIANCE, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER DIRECTLY OR INDIRECTLY ARISING FROM THE IMPLEMENTATION OF ANY TRUSTED CONNECTIVITY ALLIANCE SPECIFICATION.

Contents

1. Introduction

The Mobile Connect programme is a set of initiatives organised by the GSM Association (GSMA) to support the positioning of operators as trusted providers of identity services to third party service providers. In the context of Mobile Connect, one of the authentication mechanisms specified by the GSMA is based around the use of an authentication application provisioned on the user's SIM/UICC.

For a technical deployment of the SIM/UICC-based Mobile Connect, in general there are four components in the Mobile Connect ecosystem:

- The Card hosting the application (called in the document the SIM or the UICC).
- The Mobile Connect authentication application.
- The OTA platform.
- The MSSP platform.

While it is possible that the Mobile Connect authentication application and the OTA Platform / MSSP platform are provided by the same company (simplifying the interoperability of that specific interface), or that the Mobile Connect authentication application and the card hosting the application are provided by the same company, in general the four components may be provided by four different actors, leading to a need for interoperability of the interfaces between them.

In this context, the SIMAlliance Interoperability Working Group has used its experience in interoperability to provide guidelines for operators, service providers and solution providers to improve interworking and to simplify the interaction of the various components of the solution.

The document will first analyse the most relevant card features required to properly support the Mobile Connect application, then it will provide guidelines to applet providers and OTA/MSSP platform providers to enhance interoperability of the solution; finally, specific suggestions for testing are defined in the last chapter.

2. Abbreviations and references

2.1 Abbreviations

Abbreviation	Description
AES	Advanced Encryption Standard
3DES	Triple Data Encryption Standard
HOTP	HMAC-based One-time Password
HTTP	Hypertext Transfer Protocol
ISD	Issuer Security Domain
MAC	Message Authentication Code
MNO	Mobile Network Operator
MSL	Minimum Security Level
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
SPI	Security Parameter Indication
SSD	Supplementary Security Domain
SIM	Subscriber Identity Module
SMS-MT	Short Message Mobile Terminated
SMS-MO	Short Message Mobile Originated
SMS	Short Message Service
TLV	Tag Length Value

Figure 2.1 Table of abbreviations

2.2 References

Ref	Title
[1]	CPAS-8 - SIM Applet Authentication Specification v2.1.1
[2]	CPAS-Test - Mobile Connect SIM Applet Test Plans for CPAS 8 v1.2.1
[3]	CPAS-13 CPAS 13 – Mobile Connect SIM Applet – LoA 4 Extension Functional Specification
[4]	ETSI TS 102 223 - Smart Cards; Card Application Toolkit (CAT)
[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 v2.2 Amendment A v1.0
[10]	GlobalPlatform Card Specification v.2.2 Amendment B v1.1.1
[11]	3GPP TS 23.040 - Technical Specification Group Core Network and Terminals;Technical realisation of the Short Message Service (SMS)
[12]	GSM 03.40 - Digital cellular telecommunications system (Phase 2+); Technical realisation of the Short Message Service (SMS) Point-to-Point (PP)
[13]	3GPP TS 43.019 Subscriber Identity Module Application Programming Interface (SIM API) for Java Card Release 5
[14]	ETSI TS 102 241 Smart Cards; UICC Application Programming Interface (UICC API) for Java Card (TM) Release 6
[15]	ETSI TS 102 127 Smart Cards; Transport protocol for CAT applications Release 6
[16]	SIMAlliance - Interoperability Stepping Stones Release 7
[17]	3GPP TS 31.111 Technical Specification Group Terminals; USIM Application Toolkit (USAT)
[18]	Mobile Near Field Communication, (Mobile NFC) Stepping Stones, Version 1.0.0
[19]	ETSI TS 102 483 Smart Cards; UICC-Terminal interface; Internet Protocol connectivity between the UICC and terminal

Figure 2.2 Table of references

3. Mobile Connect card requirements

3.1 The challenge of Mobile Connect

In this chapter we will analyse the requirements on the SIM/UICC card to support deployments of Mobile Connect.

One of the major key requirements of the Mobile Connect initiative is the ability to be compatible with the majority of the cards already in the field, to guarantee strong penetration of the service.

The specification of the Mobile Connect solution has been designed since the beginning with this broad scenario in mind. In any case card technology has evolved over recent years, moving from very low end SIM cards with small memory sizes and compliant with earlier versions of specifications to high end UICC cards that provide higher memory resources, computation capabilities and software functionalities. The Mobile Connect card authentication application, then, presents different features that depend on the specific generation of the hosting card.

The goal of this chapter is to analyse how the behaviour of the Mobile Connect card authentication application depends on the operating system. Some of these dependencies will impact the interaction of the Mobile Connect card authentication application with the other components of the system and require the application or other such components to be adapted; other dependencies will also impact and limit the functionality of the application itself.

As a consequence, it is envisaged that, as there are several generations of UICC cards on the field, there will be several implementations of the Mobile Connect card authentication application, each of them matching specific implementations of the ecosystem. The rest of the chapter identifies such versions, defining the impact of specific choices in order to allow the selection of the version best suiting requirements. Then, for each version, the interoperability of the solution is analysed, providing specific suggestions to simplify interaction among the different components.

3.2 The technology of implementing the Mobile Connect card application

There are several ways to implement the Mobile Connect card application. Mobile Connect can be implemented as a standalone toolkit application, as an application with a GUI on the handset or as an extension to a browser (e.g. an extension to the WIB).

For any of the above options, there are several technologies to develop Mobile Connect on the smart card; Java Card is the most widespread solution for interoperable implementation, MultOS is an alternative option; in addition, it is possible to implement Mobile Connect as a “native” application, i.e. directly as part of the card operating system.

Java Card is considered by SIMalliance members the most widespread and interoperable solution, supported by a full set of specifications for application management. The rest of the document will focus on the Java Card implementation of Mobile Connect.

SIMalliance members agree that the most interoperable solution for the Mobile Connect card application is Java Card based.

3.3 Evolution of the standard

During the last two decades, telecom cards have evolved from early SIM cards with low memory and compliant with early versions of the standards to more recent UICC cards with higher memory and compliant with later standards.

In the scope of Mobile Connect, specific features of the telecom cards impact the deployment of the application. For each feature, a specific label is defined that will be used in the rest of the document.

Acronym	Feature	Major impact
JCAPI	Version of the Java Card API implemented on the UICC	Availability of specific authentication handlers and security algorithms
GP	GlobalPlatform version	Deployment model of the Mobile Connect
TOOLKIT	Version of the toolkit API implemented	Memory resources consumed by the application Availability of specific robustness behaviour
OTA-SMS	Version of OTA over SMS supported	Compatibility with the OTA server Management of specific communication scenarios
OTA-ENHANCED	Support of enhanced versions of protocols (CAT-TP; HTTPs)	Card content management capabilities and efficiency
PKI-SUPPORT	Support of PKI algorithms at Java Card level	Availability of LoA 4 Authentication handlers

Figure 3.1 Card features and related impacts to the Mobile Connect application functionality

3.3.1 Java Card API support (JCAPI)

The Java Card version supported by the UICC impacts the Mobile Connect deployment in two ways:

- A version compiled against a specific version of the Java Card specification cannot be deployed on UICCs supporting earlier versions of Java Card.

- In addition, in several versions of the Java Card specifications new cryptographic features have been made available to the applications.

With regards to the first point (the version of the Java Card supported), in order to guarantee maximum compatibility with the cards on the field, developers are advised to compile the Mobile Connect application with the minimum version of the JDK required by the Mobile Connect application implementation.

Minimum JDK requirement is also related to the toolkit version implemented; SIM toolkit specification requires JDK v. 2.1, UICC toolkit specification requires JDK v. 2.2.1.

SIMalliance members agree that all cards supporting sim.toolkit APIs support at least JDK 2.1, while cards supporting uicc.toolkit APIs support at least JDK 2.2.1

For the cryptographic services available to the application, the following constants are relevant to the Mobile Connect services:

For Authentication Handler OATH OCRA

- KeyBuilder.TYPE_HMAC
- Signature.ALG_HMAC_SHA1

For Authentication Handler DES

- Signature.ALG_DES_MAC8_ISO9797_M2

For Authentication Handler AES

- Signature.ALG_AES_MAC_128_NOPAD

Note: the indicated algorithms are the best option for implementing the Mobile Connect cryptographic functionalities but not the only option; e.g. ISO9797_M2 can be implemented by the application starting from DES/AES primitives if the relevant mechanism is not available on the UICC or if the application needs to be compatible with earlier generations of the UICC.

For the AES authentication handler, current required support from the Java Card consists in providing the AES algorithm; the application needs then to perform additional operations to implement the CMAC algorithm. It is worth noting that future versions of the Java Card are supposed to directly add CMAC support.

PKI Authentication handlers (LoA 4) are specified in CPAS-13 [3]**Error! Reference source not found.** and t hey are considered out of scope from this current version of Stepping Stones; they will require additional cryptographic mechanisms.

Despite the above constants having been introduced in specific versions of the Java Card specification, their support is optional, so it is not guaranteed that a card supporting a specific version of Java Card supports all the algorithms introduced in that specification; hence, the related support must be explicitly checked on the UICC.

It is worth noting that the availability of a cryptographic functionality in the UICC does not necessarily imply that the relevant cryptographic mechanism is available at Java Card level. So, even if the UICC supports RSA or AES algorithms in its operating system, it is not guaranteed that the same algorithms are also available to the Mobile Connect application.

To gather a reference to an object performing the cryptographic and signature operations, Java Card applications need to indicate a specific parameter that is named “externalAccess”. This parameter indicates if the object can be accessible by multiple instances of the applet and by other applications or not. In some smart card implementations, when the parameter indicates “do not share with other applications” consider the related

object accessible only when the applet is *selected*. As toolkit applets are not selected when executing the *processToolkit* method, the object would not be accessible.

Interoperability tip: SIMalliance members agree to suggest usage of "externalAccess=true" parameter in toolkit applications including the Mobile Connect SIM application.

3.3.2 GlobalPlatform support (GP)

The GlobalPlatform card architecture has an impact on the deployment of the application itself and on the security mechanisms provided to the application, in particular via the OTA protocol.

Mobile Connect deployments that rely on the OTA security fully use the GlobalPlatform specification to provide security mechanisms to the application.

In particular, the possibility of having several security domains and the ability to associate an application to a specific security domain to reuse that specific key set has been specified in the early versions of the GlobalPlatform specification; deployment schemes "Applet deployed in ISD using OTA keys" and "Applet deployed in a dedicated SSD using OTA keys" are supported by the majority of the cards supporting GlobalPlatform.

On the other hand, SCP02 over SCP80/SCP81 support has been clarified only in the Confidential Card Content Management Amendment [9], and it is available only on more recent and advanced cards (such as NFC cards). For models requiring SCP02 over SCP80, the Mobile Connect application must be developed accordingly, so it is advised not to adopt the SCP02 over SCP80 model, in order to enhance the compatibility with cards on the field.

In fact, an application supporting the SCP02 over SCP80/SCP81 needs to import the GlobalPlatform API to use the *SecureChannel* interface. Such an interface is typically not provided by older cards in the field and in that case if an application using the interface is downloaded on a card that doesn't support it, the application loading will fail.

3.3.3 Toolkit support (TOOLKIT)

As the SIM or Card application toolkit is the suggested way of implementing the Mobile Connect application, it is very important that the required toolkit version is supported by the card.

SIM APIs [13] is the first standard specifying the interaction between a Java Card application and the telecom services. Several versions have been defined over the years, the last of which is Release 5.

SIMalliance members guarantee that cards supporting SIM APIs are compliant with Release 5.

For the UICC APIs [14], the following services are relevant to the Mobile Connect application:

- The advanced usage of the re-entrance mechanism (with event Proactive Handler available and the issuance of 0x9300 status word by the application).
- The presence of the Volatile Byte Array in UICCPlatform that reduces the amount of memory required by the application.

Both features are already present in the first version of the UICC toolkit API (Release 6). In order to enhance compatibility of the Mobile Connect application with cards on the field, it is advised to compile the Mobile Connect application referencing Release 6 of the UICC API.

3.3.4 Support of 102 225 / 102 226 (OTA SMS)

The Mobile Connect application interacts with the server using the OTA protocol (ETSI TS 102 225 [5] and ETSI TS 102 226 [6]). In several versions of the standards, specific additions have been introduced:

- Support of enhanced protocols in securing the SMS.
- Support of incoming / outgoing concatenated SMS.

For the first point, the AES protocol for OTA has been introduced in recent cards and may not be supported in older cards; typically, Release 5 and Release 6 cards do not support AES in OTA or support preliminary versions of the standard.

The support of AES in OTA does not impact the implementation of the Mobile Connect application, which is agnostic, but the OTA platform and the MSSP platform may be impacted by the usage of AES.

In addition, for personalising authentication handlers based on AES it is recommended to use AES as transport security, i.e. a Set Authentication Handler Data command setting an Authentication key for AES Authentication handler has to be secured by using AES SCP80 keys.

SIMalliance members suggest supporting both security mechanisms (DES and AES) on the OTA server and using AES protocol when it is supported by the card.

As for support of incoming concatenated OTA SMS, the feature is widely supported by cards on the field.

SIMalliance members confirm that the majority of the cards on the fields support concatenation of incoming SMS.

Outgoing concatenated OTA SMS (response packet) are less well supported by cards in the field; the requirement from the Mobile Connect application is limited to the GET DATA command with a large amount of data (see 5.5).

3.3.5 Support of enhanced OTA protocols (OTA ENHANCED)

Specific enhancements to OTA have greatly improved the capabilities of OTA protocols.

Older versions of OTA protocols are based on SMS, a bearer that is not reliable and very slow. In contrast, enhanced versions of the OTA protocol are based on IP connections (HTTPs [10] and CAT-TP [15]. Such protocols are much more efficient, especially when large amounts of data have to be exchanged between the card and the platform.

As the Mobile Connect application does not transmit large amounts of data, the specification is designed to work only over SMS. Application management on the other hand requires the transmission of larger amounts of data; downloading the Mobile Connect on an UICC may require tens of SMSs, subject to reliability issues.

For this reason, it is recommended to adopt these enhanced protocols to perform high data operations, when available. Such protocols are typically available in high end cards.

3.3.6 Support of PKI algorithms (PKI-SUPPORT)

To support LoA 4 authentication handlers, PKI algorithms are required to be supported by the UICC. This is typically done by means of a cryptographic coprocessor that provides RSA and Elliptic Curve cryptography implementation. ECC support has been introduced quite recently in the UICC.

From the Mobile Connect application point of view, if the UICC does not support the relevant cryptographic algorithm, the application cannot implement LoA 4 authentication. Mobile Connect LoA 4 is specified in CPAS-13 [3] and it is considered out of scope for the current version of Stepping Stones.

In general, the above parameters may be combined in different ways to obtain very different products; as an example, it is possible to configure a card with RAM over HTTPs but without uicc.toolkit support.

Such combinations are unlikely anyhow due to the fact that there are several dependencies among the parameters implying that specific versions of the standards are related each other.

In the field, it is possible to identify several "generations" of the UICC; in the following paragraph a few representative examples are listed, with the proviso that the configurations on the field may be different.

The identified representative examples are:

- SIM card (2G)
    - Typically, these cards have low memory, they support only SIM API and support old versions of Java Card and GlobalPlatform Card Specifications.
    - DES is supported also at Java Card level, AES may be supported but typically is not.
    - Enhanced bearers are typically not supported.

- USIM card (3G) Release 6
    - Typically, these cards have average memory, they support both SIM and UICC API and support Java Card 2.2.1 and above and GlobalPlatform v2.1.1 and above.
    - DES and AES are typically supported also at Java Card level.
    - Enhanced bearers are typically not supported.

- USIM card (3G) Release 8
    - Typically, those cards have average memory, they support both SIM and UICC API and support Java Card at least 2.2.1 and at least GlobalPlatform v2.2.
    - DES and AES are typically supported also at Java Card level.
    - Enhanced bearers are typically supported.

- NFC and Embedded UICC
    - Typically, those cards have high memory, they support both SIM and UICC API and support Java Card at least 3.0.1 and at least GlobalPlatform v2.2.1 plus Amendments.
    - DES and AES are typically supported also at Java Card level.
    - Enhanced bearers are typically supported.
    - Asymmetric cryptography for evolutions of authentication handlers is typically supported.

Card configuration	Typical support	Impact on Mobile Connect implementation
SIM card (2G)	Early versions of JC and GP, SIM API, DES, no OTA enhanced	DES based authentication algorithms; higher RAM occupation; GP deployment models feasibility to be verified; no OTA enhanced to update the application
USIM card (3G) Release 6	Java Card at least 2.2.1 and GP at least 2.1.1, SIM API + UICC API, DES, AES, no OTA enhanced	DES and AES based authentication algorithms; lower RAM occupation; GP deployment models feasible; no OTA enhanced to update the application
USIM card (3G) Release 8	Java Card at least 2.2.1 and GP at least 2.2, SIM API + UICC API, DES, AES, OTA enhanced	DES, AES and OATH based authentication algorithms; lower RAM occupation; GP deployment models feasible; OTA enhanced supported to update the application
NFC and embedded UICC	Java Card at least 3.0.1 and GP at least 2.2.1, SIM API + UICC API, DES, AES, OTA enhanced, asymmetric cryptography supported	DES, AES and OATH based authentication algorithms; lower RAM occupation; GP deployment models feasible; OTA enhanced supported to update the application; possible evolution of authentication handlers to support asymmetric cryptography

Figure 3.2 UICC generations

4. Applet development

4.1.1 Overview of applet development

The Mobile Connect ecosystem empowers the intended customers to use their mobile identity as a key to securely authenticate to any service. The imperative element is that the user passcode is uniquely and securely stored in the secure element (SIM) and never shared with the OTA platform, MSSP server or any other web component.

This works through the concept of applet, OTA platform and MSSP server communication. The communication is always initiated by the OTA platform and/or MSSP server while the applet replies by sending the defined response using a secure transport bearer.

Applet, OTA platform or MSSP server communication is based on commands defined within [1]. Each command is associated with certain procedures, the mapping of which is mentioned below:

S.No.	Commands	Related procedure
1	Sign Transaction	Authentication/Signature request
2	Manage Personal Code	Reset Personal Code/Unblock Personal Code
3	Get Applet data	No specific procedure MSSP Server uses this command to fetch required data from applet
4	Set Applet data	First registration of an end-user/Un-registering an end-user
	Set Authentication Handler data	First registration of an end-user
	Set E2E transport key	No specific procedure; can be used when E2E key need to be set
5	Change Applet Status	First registration of an end-user/ De-registering an end-user
6	Delete Authentication Handler	De-registering an end-user

Figure 4.1 Applet, OTA platform or MSSP server communication commands

In the context of authentication, the Applet specifies the following range of authentication mechanisms, each mapped and certified to a defined level of assurance.

Authentication mechanisms	Level of Assurance (LoA)
Click OK	LoA2: Some confidence
Enter Personal Code	LoA3: High confidence
Mobile Signature (certificates)	LoA4: Very high confidence

Figure 4.2 Authentication mechanisms

An applet can implement any one or all authentication mechanisms depending on the intended vertical where the service is to be deployed; for example LoA3 would possibly be used in banking solutions.

4.1.2 API for applet implementation

The card authentication applet can preferably be implemented using standard Java Card libraries.

While using any standard Java Card API a tradeoff shall be maintained with interoperability and usage of latest version of APIs.

For preferable use of APIs refer to section 3.3.1. However, the decision of API usage lies with the applet developer and depends on assorted factors like the applet deployed at card issuance or by OTA, targeted card profile etc.

APIs within following packages of Java Card can be used:

- Java Card:
    - javacardkit
    - java.lang
    - javacard.security
    - javacard.framework
    - javacardx.crypto.
- Java Card toolkit:
    - sim.toolkit/uicc.toolkit
    - sim.access/uicc.access.

The file access is usually not recommended due to inherited concerns like interoperability or OTA deployment complexity. However, it can be used depending on specific requirements or MNO agreements.

4.1.3 Card architecture options for applet implementation

GSMA implies the following options for deployment of the card authentication applet:

1. **As a standalone applet:** A card authentication applet can be provisioned as a standalone applet coupled with a simple text based user interface. The user interface can be extended as per requirement but a tradeoff has to be maintained with the applet size for interoperable RAM options.

2. **As a SIM browser plug-in:** To achieve a simple applet design together with a reasonable size the card authentication applet can be deployed as a SIM browser plug-in delegating the GUI management to the plug-in.

3. **As a standalone applet coupled with the GUI of the device application:** A card authentication applet can be provisioned as a standalone applet which relies on the native application of the device for GUI management. The interaction between the device application and SIM applet is not defined by the GSMA so currently is in scope of exploration by the service provider.

For interoperable and simple architectures; the first option is most preferred and hence will be in scope for this document.

For browser solutions, as a prerequisite a browser (S@T/WIB) should already be installed on the card. Moreover, even if the browser is installed on the card, the interface between the MSSP server and the card is not compatible with the WIB and S@T specifications; so this interface has to be adapted to change the format of the data exchanged. In WIB/S@T, the engine executes byte code, not APDUs coming in Compact Mode, therefore the browser solution is not preferred option.

A few preferred design considerations are:

- Although the operator is free to deploy the applet at card issuance, to be certain of higher completion rates the applet should be designed to be deployable using an OTA download. For this the applet code i.e. OTA payload; shall be as low as possible.
- To minimise the applet size a balance should be well maintained with user interface text strings stored within the applet, supportability of a number of authentication handlers, supportability of optional or additional features etc; depending on the targeted market segment.
- To take advantage of the security features of card's security domain; the applet can be associated with an ISD or a SSD.

4.1.4 APDU format and transport bearer

The card authentication applet works on commands (defined in [1]) received by the server. A single secure message sent by the MSSP server may contain single or multiple commands. In both cases MSSP can use a compact/expanded structure.

For the current implementation of the Mobile Connect ecosystem the recommended choice is to use a compact format command and response structure.

Since secure authentication is the primary concern of the Mobile Connect system, it is mandatory to use a secure transport bearer for all communications between the server and applet. Depending on the applet implementation; it can use POR and SMS or only SMS as the transport bearer.

PoR may be used by the card application applet to protect the response message by using the SCP80 security of the associated security domain.

However, in the case of POR; a 03.19 design constraint of unavailability of *EnvelopeResponseHandler* comes when a server command enforces the applet to use the *ProactiveHandler.send* method to capture user interaction and further share the user response to the server. In such a case in addition to POR, a SMS containing the user response has to be used. Due to the unavailability of SCP80 security with SMS originating from applet, applicative security should be considered.

4.1.5 Multi-language support

The card authentication applet involves a text based user interface. To afford unequivocal user experience the applet should allow the OTA platform and/or MSSP server to lever the language of textual messages based on user language preferences or other defined criteria.

The messages to be shown to user during various journeys can be categorised as:

1. Static messages owned and stored by/within applet.
2. Messages sent by the MSSP within the authentication request.

The language switch is applicable only for the former as the latter uses data coding schemes to handle the desired language.

A few recommended design considerations are:

- Static messages can be stored in elementary files owned by the applet. For language change, an OTA update via RFM can be used. This is not a preferable option due to inherited concerns with file access but can be used depending on the targeted market and MNO agreement.
- Static messages can be stored in the applet buffer or in a separate resource bundle and a provision for OTA update can be provided.
- For UCS2 characters, support of various text coding schemes like 80, 81 and 82 needs to be handled by applet design and codes.
- UCS2 must be supported by the handset; in addition the character set supported by the handset may differ handset by handset.

To support multiple languages, the card authentication applet can be implemented as:

1. Dedicated applet for each language.
2. Unique applet package configured with a separate resource bundle.
3. Separate command to switch the language.

A dedicated applet for each language comes with a simple design but is not recommended due to high maintenance factors.

A separate resource bundle can be implemented by means of a “message provider” applet that interacts with the Mobile Connect applet through a shareable interface. The “message provider” applet implements a getMessage method. In order to retrieve the message the Mobile Connect applet calls the shareable interface method of the message provider applet that returns the reference to a shareable interface object.

Then the Mobile Connect applet calls the getMessage method passing a global array and the message ID which the application would retrieve.

A separate command to switch the language is not defined by GSMA so currently is in scope of exploration by the service provider.

The applet implementer can choose further options as long as they are compatible with the overall applet design and a tradeoff is maintained with basic parameters like applet size, applet deployment etc.

For the options that are already available, the table below mentions compatibility with a few basic criteria.

Criteria	Dedicated applet for each language	Unique Applet package configured with a separate resource bundle
Applet size	OK	OK
Interoperability of the solution	OK	OK
Pre Issuance Deployment	OK	OK
OTA Deployment	OK	OK
OTA Update (Language/Application)	Less efficient	More efficient
Applet Maintenance and testing	Less efficient	More efficient
Recommended by GSMA (section 6.5 of [1])	No	Yes

Figure 4.3 Applet implementation options

Multi language support implemented by an external resource bundle offers several advantages compared to a dedicated application for each language. The main advantages concern:

- OTA update of the application: the size of application without the text is smaller.
- Maintenance: in the case of new languages all that is needed is to change the language bundle, not to rebuild the complete application.
- Testing benefits: the application can be fully tested with one language while user interaction text can be verified for all other languages.

4.1.6 Applet interaction

The section discusses the Applet triggering once it has been deployed in the SIM card. There are different user journeys where applet triggering is required while some journeys do not require applet triggering. Based on this, user journeys can be divided into two parts.

1. **User Interaction From Applet**
    - "Click Ok" Journey.
    - "Personal Code" Journey.
    - Personal Code Creation Journey.
    - Reset Personal Code Journey.
    - User Cancel the Authentication Request.
    - Unsubscribe Mobile Connect service.

2. **No user interaction from applet**

There are a few cases in which user interaction is not mandatory and will depend upon applet implementation.

- Wrong Personal Code Journey/Unblock.

  For user-friendly behaviour, it is recommended to design the applet to display a message to the user via Display Text proactive command.

- Changing Cards.

4.1.7 Authentication handler object

An Authentication Handler is an authentication method which stores various parameters like authentication key, authentication id etc. as per defined handler type.

An applet can support multiple pre-configured handlers and/or handlers can be created upon server request using set authentication handler command defined in [1].

The handlers will be invoked by authentication server to authenticate the end-user.

- Handler structure

Tag	Length	Value	MOC
'Bx'	var	Authentication Handler type tag	M
'AA'	1 byte	Authentication Handler identifier	M
'AC'	var	Authentication key	O
'AD'	8 byte	Counter	O

Figure 4.4 Handler structure

- Key Size:

Authentication Handler	Authentication Key Length
Secure Messaging layer	Not Required
3DES-CBC	128 bits (2 keys) or 192 (3 keys)
AES-CMAC	128 bits
OATH-OCRA	160 bits

Figure 4.5 Key size

Since the handler's structure is of the TLV type, a possible method to store an authentication handler is to use the BerTLVEditHandler object. The BERTLVEditHandler interface offers methods to set the TAG and to edit the BER TLV list. It is provided by Java Card toolkit Rel6 uicc.toolkit package.

This solution reduces the size of the applet but cannot be implemented in the cards compliant with ETSI Releases earlier than Release 6.

For a solution that is interoperable with cards compliant with ETSI Releases earlier than Release 6, it is recommended to define a dedicated class containing all the elements that identify an authentication handler and implements get/set methods for the different management operations.

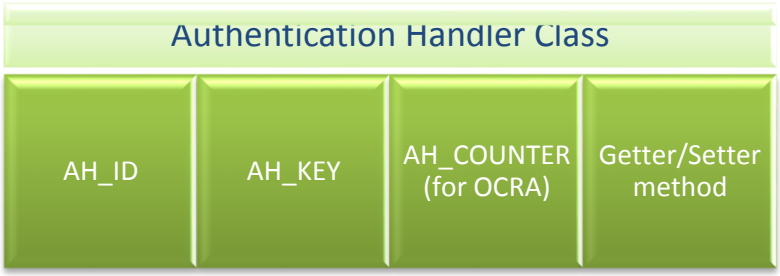

Figure 4.6 Authentication handler class

With this, each operation (create, view, update, delete) is fast and simple and the size of the applet does not increase significantly.

4.1.8 OATH OCRA implementation

OCRA is an algorithm for challenge-response authentication developed by the OATH. It leverages the HMAC-based One-Time Password (HOTP) algorithm and offer one-way mutual authentication and electronic signature capabilities.

It is recommended to use the algorithms provided by the platform because the implementation provided by the operating system is often protected with countermeasures and in high-end cards may also benefit security certifications according to recognised schemas.

The CMAC is in any case not present in any Java Card specification to date and so it must be implemented at application level.

HMAC may be not present in cards compliant to earlier Releases than Release 8 so it may be implemented at application level to increase compatibility but in this case, it is recommended to implement proper security enforcements in the application.

HMACKey is a public interface of the javacard.security library and contains a key for HMAC operations.

Java Card vers.	Card rel.	HMACKey Presence
JC 2.1.1	<= R5	NO
JC 2.2.1	R6	NO
JC 2.2.2	R8	YES
JC 3.0.4	R9	YES

Figure 4.7 HMAC key operations

To solve problems of interoperability arising from the versions of the Java Card as well as the type of cipher algorithm and the available memory space present in the various releases of the cards, the suggested approach could be the following:

Card rel.	Authentication handler algorithm
<=R6	3DES
R8	AES
R9	AES or OATH OCRA

Figure 4.8 Authentication handler algorithm options

4.1.9 Personal Code management

In term of interoperability, the best solution to store and manage the Personal Code is to use the class javacard.framework.OwnerPIN.

This class represents an owner PIN, implements Personal Identification Number functionality as defined in the PIN interface, and provides the ability to update the PIN and thus owner functionality.

4.1.10 MSL

With applicative e2e security SPI 0000 can be used only if a mechanism is deployed for secure management of e2e keys. In such a case pre-shared e2e keys can be an option. The key sharing mechanism is decided by the applet developer. A recommendation is to use a mutually agreed algorithm to share the first key and for subsequent keys to use GSMA E2E key commands.

It is recommended to rely on SCP80 transport security with MSL SPI 1639. For communications involving proactive session e2e security must be used.

Even if the e2e security is configured with SPI 0000, it is suggested to always use a higher level of SPI that includes sender authentication at 102 225 level (such as 1600).

4.1.11 Delete instance

A few recommendations are:

1. Do not store the ToolkitRegistry instance in a static field or if done it should be reset during the uninstall method
2. Use method JCSystem.requestObjectDeletion () to activate the service to delete Java Card RE objects.
3. Implement proper uninstall method to de-reference all static fields. Java Card v2.2.1 interface appletEvent provides the uninstall method.

4.1.12 Java Card rules for specific ecosystems

For any Java Card implementation with commercial scope, some guidelines from standardisation bodies and sector associations shall be analysed in the context of Mobile Connect development.

GlobalPlatform defined guidelines for developers to apply to any basic application (not NFC Application) that is to be loaded on a Composite Product that has been certified by an Evaluation Scheme (in the current version of the GlobalPlatform Composition Model, either Common Criteria or EMVCo). Generally this compliance is requested by the MNO to the Mobile Connect applet providers.

GSMA defined its guidelines for NFC Java Card applet development. Although the rules have been identified for NFC markets, they should be considered also on other ecosystems as the use of them is a quality assurance indicator of the product and of its interoperability.

5. OTA requirements

In general, there are two OTA use cases in the Mobile Connect architecture:

- Management of card authentication application and its associated SD.
- Interactions with card authentication application.

To carry out these OTA operations, there are several standardised ways to communicate with the card authentication application:

- SMS-PP
- CAT-TP
- RAM over HTTPs

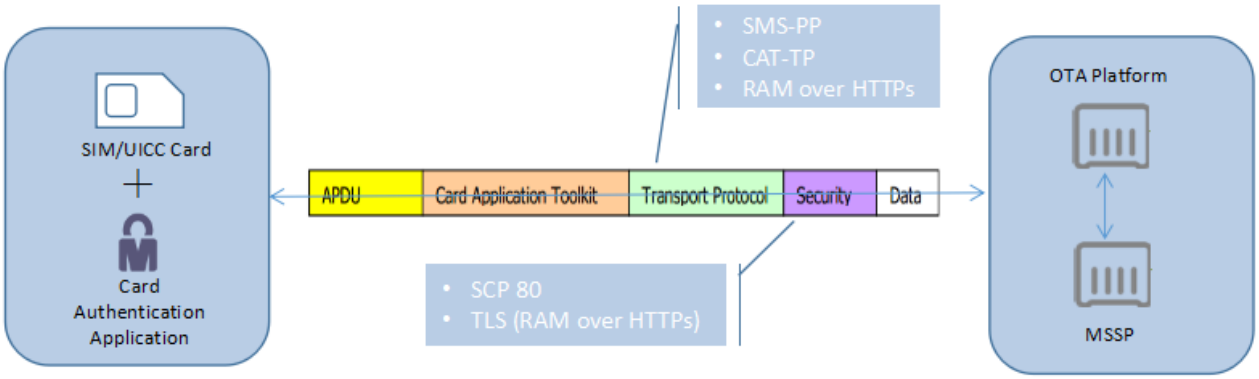

Figure 5.1 OTA protocols for Management and Interactions

Management of the card authentication application and its associated SD is typically the responsibility of the OTA Platform, including downloading, installation, putting keysets for interactions with card authentication application. Section 8.3 in CPAS08 [1] describes what functions are expected to be present in the OTA Platform. In certain deployment architectures, MSSP may also in charge of some of these operations, see section 3.3.3 for further details.

Interactions with card authentication application is in charged by the Authentication Server (MSSP), via the secure channel managed by the OTA Platform. To overcome specific Toolkit API limitations, the E2E transport secure mechanism has to be applied in some interactions. Section 5.2.9 explains what these limitations are.

Despite the fact that in the Mobile Connect Specification SMS-PP is the only way to communicate with the card authentication application, OTA Management and interactions should be done by the best available protocol that in turns depends on the card versions indicated in section 3.3. On some SIMalliance cards those features will not be present in order to save resources, e.g. on card supporting RAM over HTTPs, CAT-TP could also be disabled.

For management, choices can be SMS-PP, CAT-TP or RAM over HTTPs. For interactions, SMS-PP is the only choice in CPAS08 [1].

5.1 OTA protocols

5.1.1 Short Message Service Point-to-Point (SMS-PP) – single and concatenated

SMS-PP - single and concatenated, can be used both in OTA management and interactions. The protocols and protocol layering for (SMS-PP) are defined in 3GPP TS 23.040 [11] for UICC cards and GSM 03.40 [12] for SIM cards. See Stepping Stones R7 (ref [16], §15.1.1) for more details about command packet contained in a SMS-PP.

If payload size in OTA management and interactions are too large to be contained in a single SMS, it is concatenated. The section 5.5 shows the possibility of concatenation command by command.

Release 5 and successive cards supports SMS-PP Single and Concatenated, while Pre-Release 5 cards supports only SMS-PP Single.

ME required features:

- SMS-PP MT and SMS-PP MO as defined in ETSI TS 102 223 [4], and in 3GPP TS 23.040 [11] (or either GSM 03.40 [12] for device with SIM cards).
- The following set of commands as defined in ETSI TS 102 223 [4]**Error! Reference source not found.** a nd 3GPP TS 31.111 [17]**Error! Reference source not found.**:
    - SEND SHORT MESSAGES
    - ENVELOPE (SMS-PP DOWNLOAD).

5.1.2 Card Application Toolkit Transport Layer (CAT_TP)

The Card Application Toolkit Transport Protocol (CAT_TP) protocol is defined in the ETSI TS 102 127 [15]**Error! Reference source not found.**, Release 7.

This protocol uses the BIP mechanism to exchange data between the ME and the card. The BIP mechanism is specified in ETSI TS 102 223 [4].

The MNO will need to deploy a IP router/gateway or similar into the Mobile Connect architecture to accept TCP/UDP connections request from cards, and then hand it to OTA Platform.

See Stepping Stones Release 7 (ref.[16], §18) for more detail about CAT-TP.

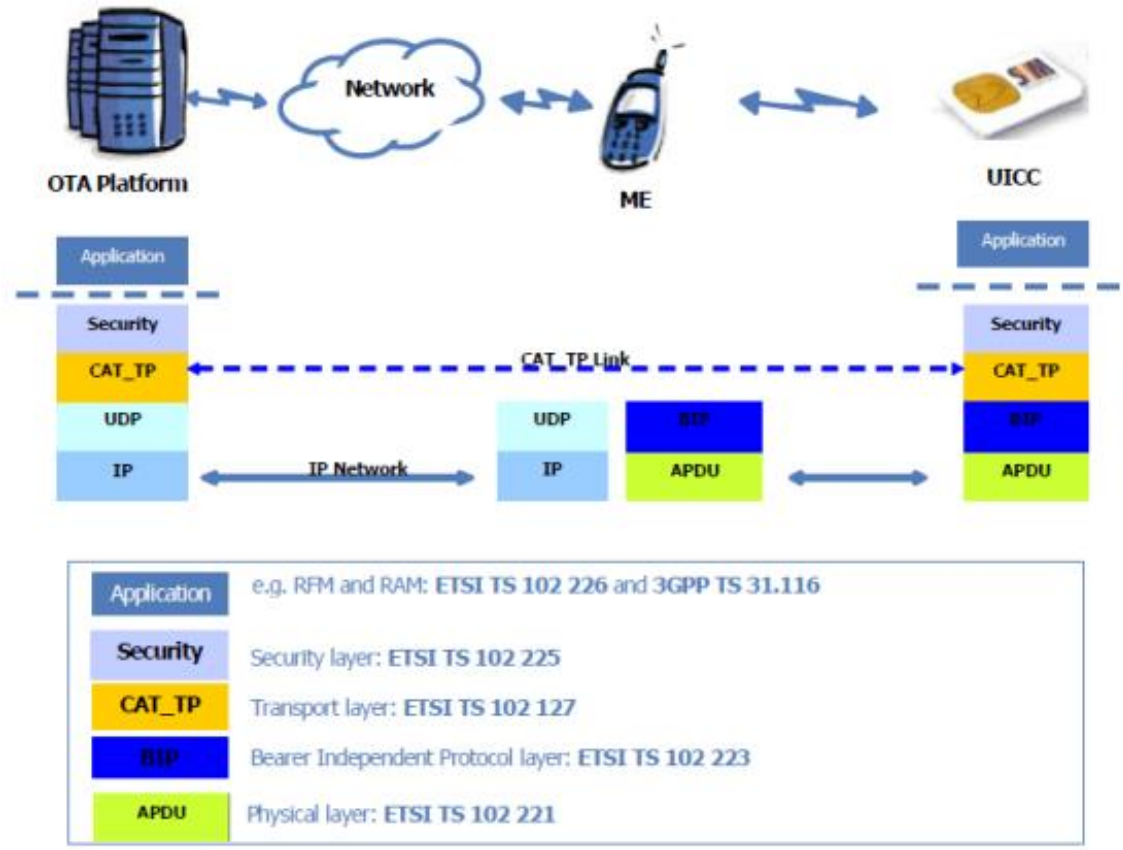

Figure 5.2 Overview CAT_TP protocol layer with associated layers in OTA platform, UE and UICC

ME required features are:

- CAT class “e“ defined in ETSI TS 102 223[4].

5.1.3 RAM over HTTPs

RAM over HTTPs, defined in GlobalPlatform Amendment B [10]**Error! Reference source not found.**, is a m echanism for an Application Provider to perform RAM using the HTTP protocol and PSK TLS security Over-The-Air. Remote File Management via HTTPs is, specified by ETSI TS 102 226 [6] by adding, in Annex B, a RFM scenario to the RAM scenario described in GlobalPlatform 2.2, Amendment B [10].

The HTTP protocol is used on top of TLS to provide encapsulation of the data and information about the receiving entity. TCP/IP transport, from SIM to ME, is provided by the Bearer Independent Protocol specified in ETSI TS 102 223 [4] or by a direct IP connection as specified in TS 102 483 **Error! Reference source not found.**19]. This Transport Layer can be used by an Application Provider to perform Remote Application Management (RAM) of its application i.e. to load, install and personalise using the HTTP protocol and PSK-TLS security over TCP/IP network.

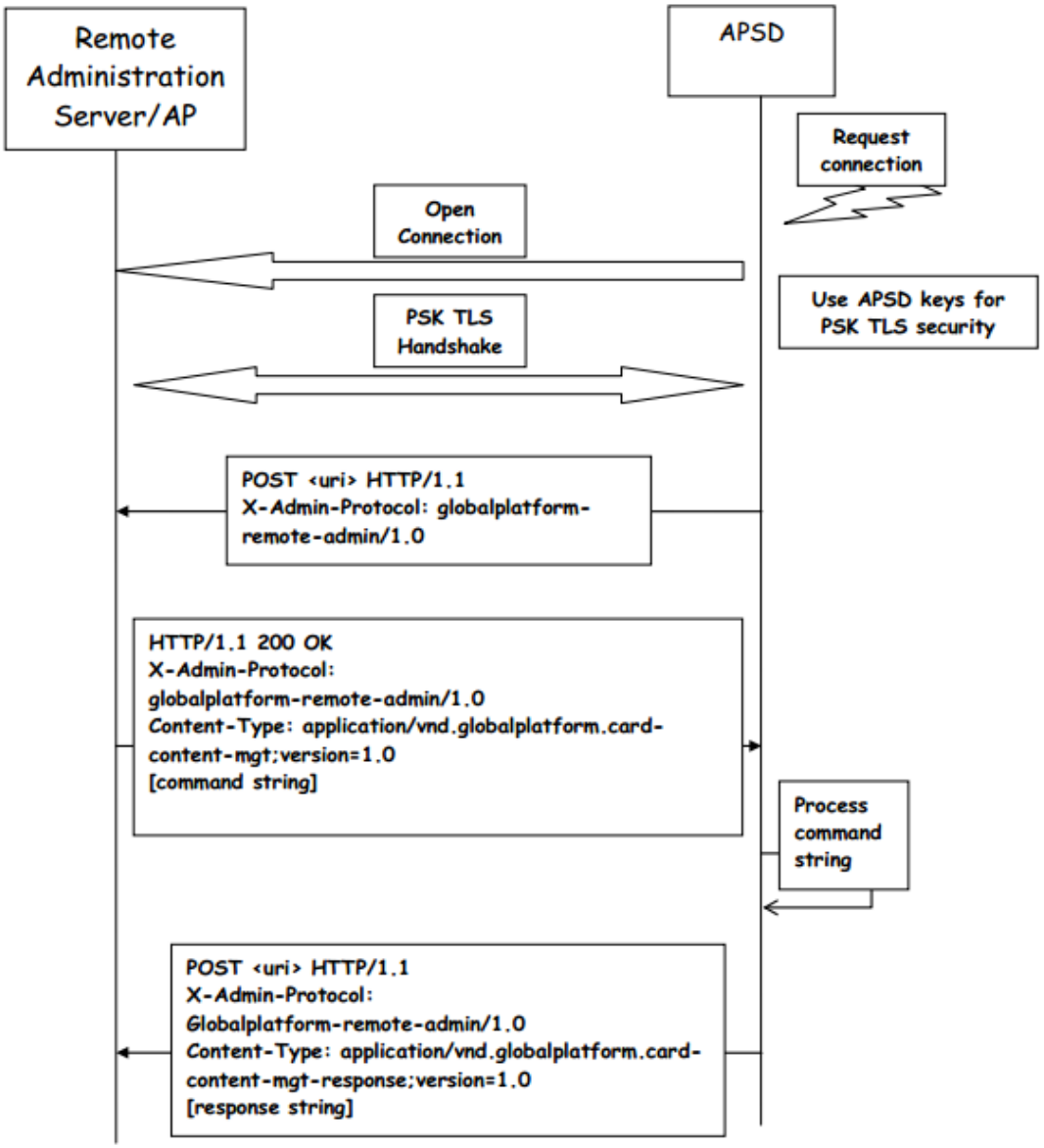

Figure 5.3 Communication flow between an application provider owning a remote administration server and its security domain

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

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

The Security Domain shall receive a triggering message. The Security Domain will handle the administration session, using its own PSK TLS keys for the communication security. It is assumed that the Security Domain knows all parameters needed to establish a connection or to handle its security. These parameters can be parameters of the triggering message or the parameters of the Security Domain itself. See [10] § 4.7 about administration session triggering parameters.

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

As it is a fairly new protocol, RAM over HTTPs is only supported on some Release 9 and successive, GP 2.2.1 enabled UICC cards. The MNO will need to deploy a TLS server and HTTP server to enable this RAM protocol. Smart Card Web Server Stepping Stones [18].**Error! Reference source not found.**

ME required features:

- CAT class “e’” defined in ETSI TS 102 223 [4]**Error! Reference source not found.**.

5.2 SD architecture

In order to have a working solution, a SD Architecture has to be designed, taking into account the considerations below among others:

- card authentication application and its deployment model.
- Applet Issuance procedure.
- Responsibilities of the MSSP and OTA(MNO) Platform.

In the sections below different SD Architectures are presented and the implications of these SD Architectures with respect to the above considerations are mentioned.

5.2.1 Dedicated SSD with scp80 keys and AM privileges

In this SD Architecture, a separate SSD is created with authorised privileges for Mobile Connect purposes and personalised with scp80 keys. The figure below depicts this SD Architecture:

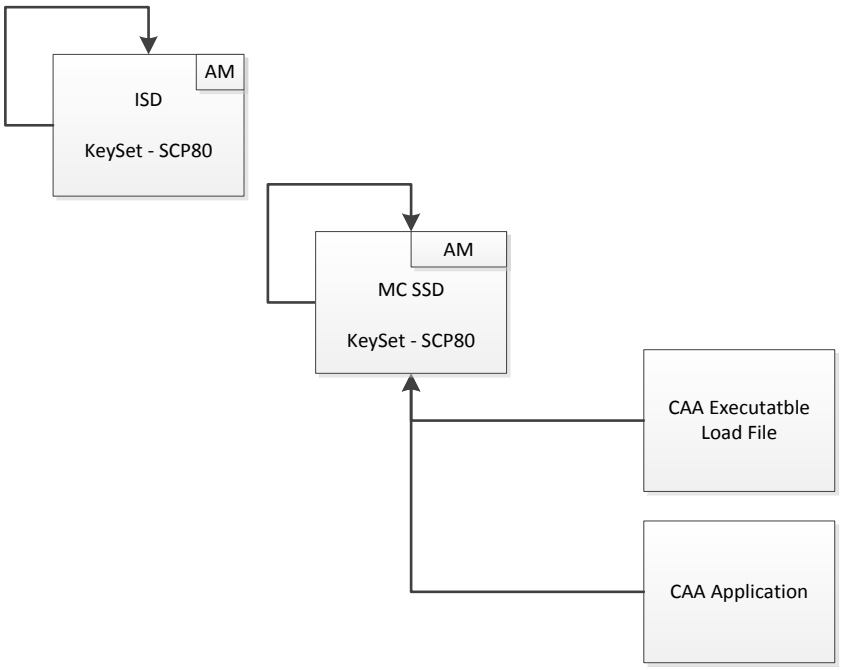

Figure 5.4 SD architecture with AM privileges

This SD Architecture can be applicable in the following scenarios:

- The card authentication application does not use Application Security.
- The “Applet deployed in a dedicated SSD using OTA keys” (section 6.3.2 in CPAS8 [1], deployment model.
- The MSSP is responsible for the Applet lifecycle activities.

5.2.2 Dedicated SSD with scp80 keys and DM Privileges

In this SD Architecture, a separate SSD is created for Mobile Connect purposes with Delegated Privileges and personalised with scp80 keys. The figure below depicts this SD Architecture:

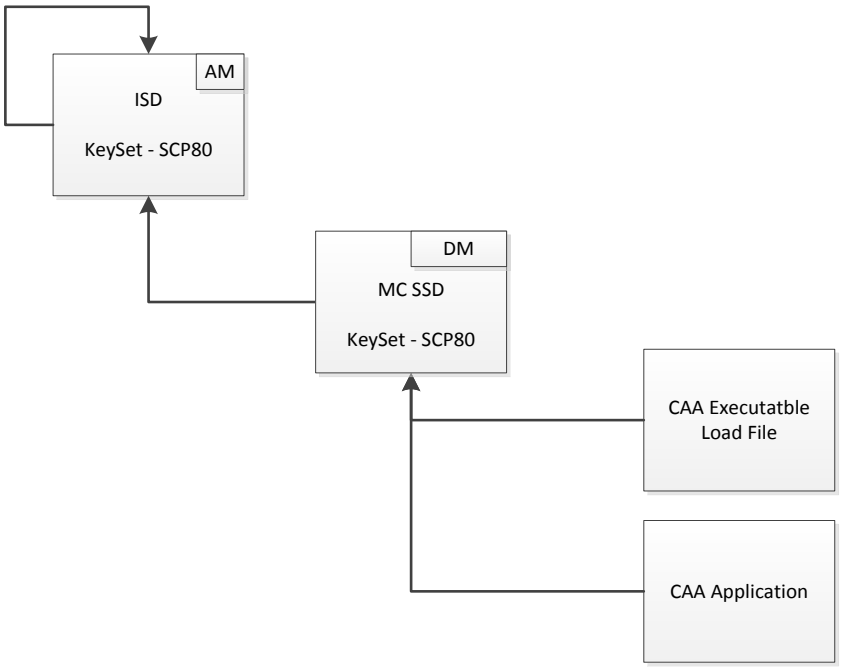

Figure 5.5 SD architecture with DM privileges

This SD Architecture can be applicable in the following scenarios:

- The card authentication application does not use Application Security.
- The “Applet deployed in a dedicated SSD using OTA keys” (section 6.3.1 in CPAS8 [1], deployment model.
- The MSSP is responsible for the Applet lifecycle activities.
- The MNO can control the Content which is being downloaded to the card.

5.2.3 Dedicated SSD with scp80 keys

In this SD Architecture, a separate SSD is created for Mobile Connect purposes without any special privileges and personalised with scp80 keys.

The figure below depicts this SD Architecture where the Content Authentication Application package and instance are both under the SSD dedicated for Mobile Connect:

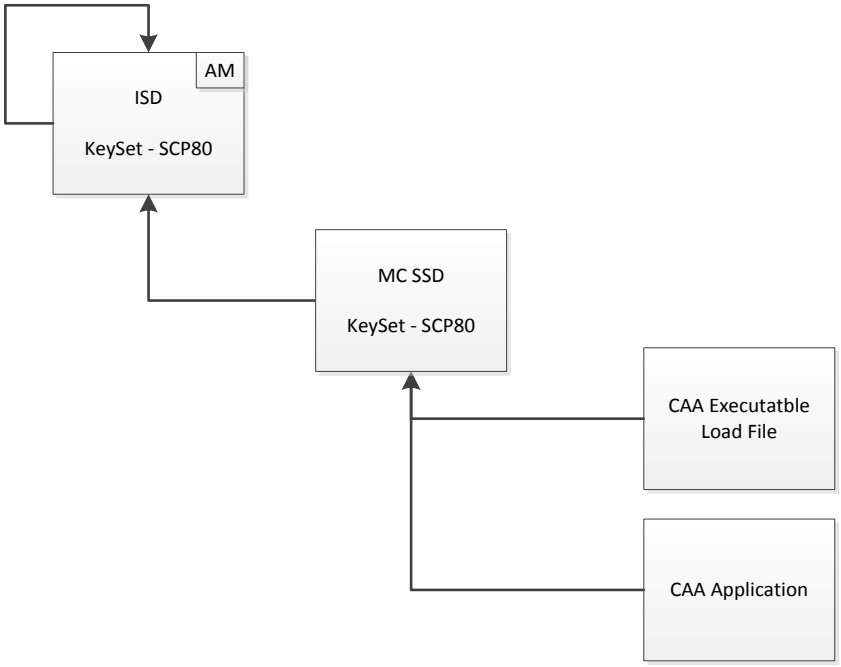

Figure 5.6 SD architecture with no privileges

The figure below depicts this SD Architecture where the Content Authentication Application package is under the ISD and the instance under the SSD dedicated for Mobile Connect:

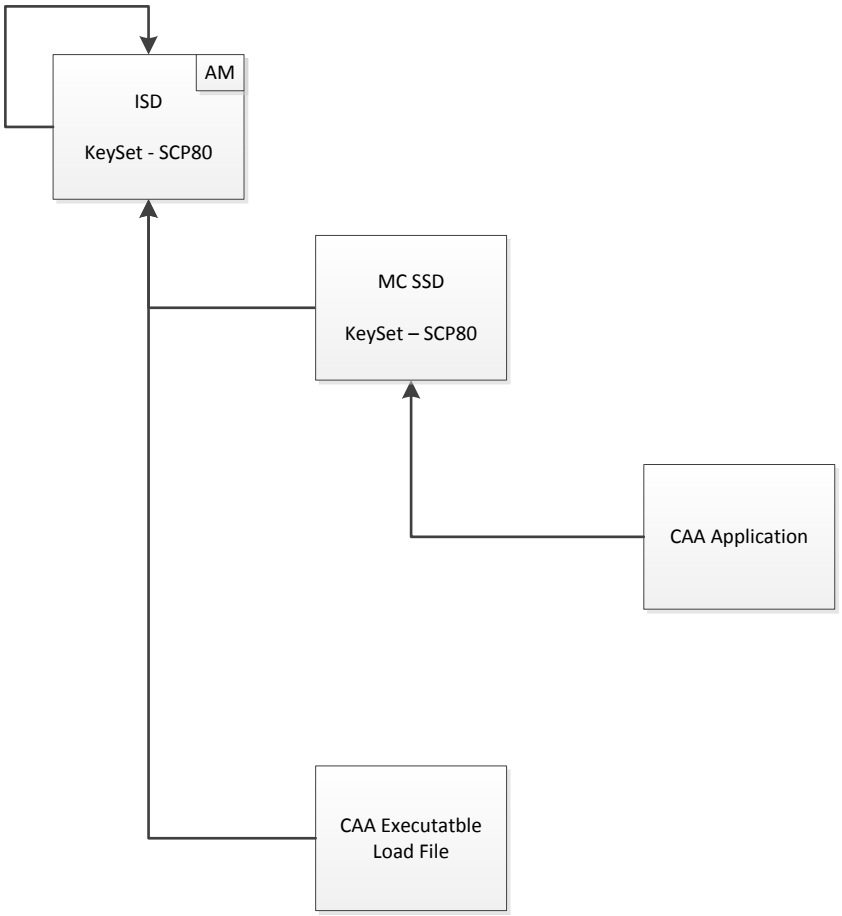

Figure 5.7 SD architecture with CAA under ISD

This SD Architecture can be applicable in the following scenarios:

- The card authentication application does not use Application Security.
- The “Applet deployed in a dedicated SSD using OTA keys” (section 6.3.2 in CPAS8 [1], deployment model.
- The MSSP can be responsible only for the Mobile Connect Operations, The MNO would be responsible for the applet issuance either in the factory or via OTA.

5.2.4 Under ISD with dedicated scp80 keys

In this SD Architecture the applet is installed under the ISD and the ISD is personalised with scp80 keys which will be used for Mobile Connect operations. The figure below depicts this SD Architecture:

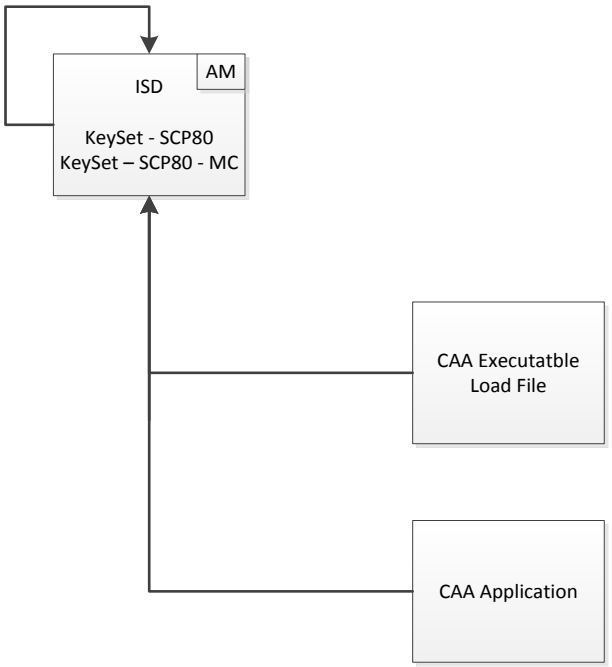

Figure 5.8 SD architecture with dedicated keys

This SD Architecture can be applicable in the following scenarios:

- The card authentication application does not use Application Security.
- The MSSP is responsible only for the Mobile Connect Operations; The MNO would be responsible for the applet issuance either in the factory or via OTA.

5.2.5 Under ISD without dedicated scp80 keys

In this SD Architecture the applet is installed under the ISD. No dedicated scp80 keyset is used for the Mobile Connect operations, instead an already existing ISD Keyset is shared with other applications. The figure below depicts this SD Architecture:

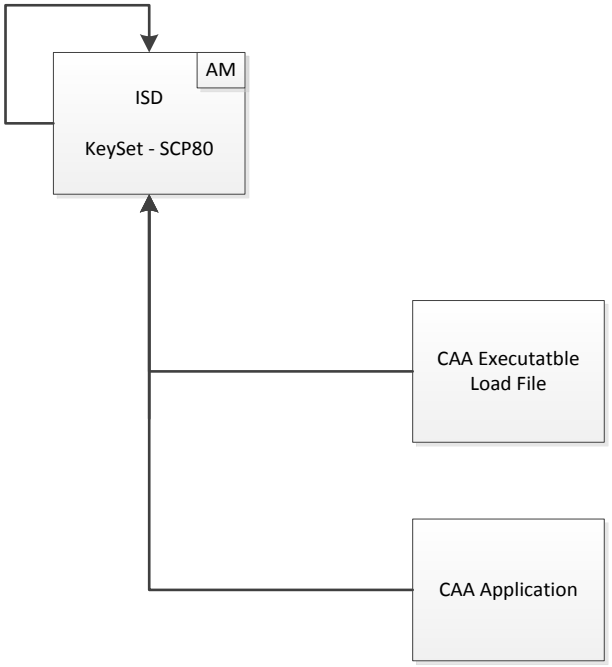

Figure 5.9 SD architecture using ISD keyset

This SD Architecture can be applicable in the following scenarios:

- The card authentication application does not use application security.
- The deployment model “Secure messaging relying on OTA platform” as specified by Mobile Connect (section 6.3.1 in CPAS8 [1]).
- The MNO would be responsible for all OTA Operations, from the life cycle management of the applet to the Mobile Connect operations.

5.2.6 Under ISD without any scp80 keys

In this SD Architecture the applet is installed under the ISD. No OTA security is used for Mobile Connect Operations, instead application security will be used. The below figure depicts this SD Architecture:

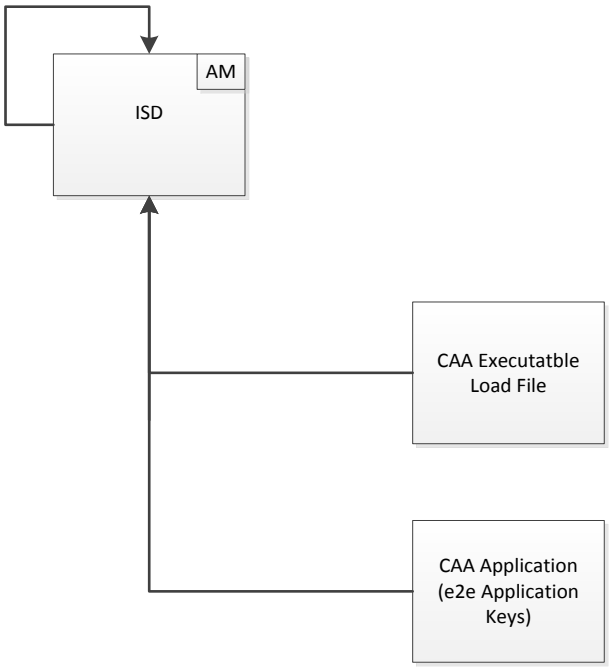

Figure 5.10 SD architecture – no OTA security

This SD Architecture can be applicable in the following scenarios:

- The card authentication application has to use application security.
- The deployment model “Applet deployed in ISD using applicative security” (section 6.3.2 in CPAS8 [1]) as specified by Mobile Connect.
- The MNO would be responsible is responsible for applet issuance either via OTA or in the factory. The MSSP can be responsible for the Mobile Connect operations.

5.2.7 Applet deployed in a dedicated SSD with applicative SCP02/03

In this SD Architecture the applet is installed under dedicated SSD and personalised with a scp02/03 keyset for Mobile Connect purposes. The figure below depicts this SD Architecture:

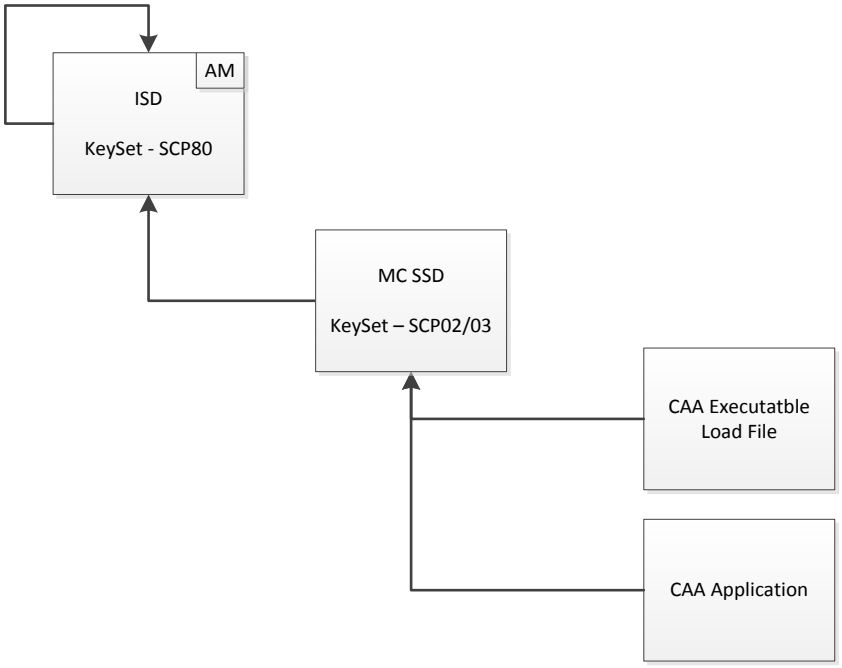

Figure 5.11 SD architecture using scp02/03

This SD Architecture can be applicable in the following scenarios:

- The card authentication application uses scp02/03 Applicative Security.
- The deployment model “Applet deployed in a dedicated SSD with applicative SCP02” as specified by Mobile Connect (section 6.3.2 in CPAS8 [1]).
- The MNO is responsible for applet issuance either via OTA or in the factory. The MSSP can be responsible for the Mobile Connect operations.

5.2.8 Recommendations from GSMA

The Stepping Stones seconds the recommendations (section 6.3.3 in CPAS8 [1]) by GSMA for Mobile Connect card authentication application i.e. using either of the following options.

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 existing cards, minimises the size of the applet and maximises the size available in the message for the applicative payload (e.g. size of the text to be displayed for the 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’ complicates the deployment solution (provisioning of the different parties, MSSP to handle SCP02/03...) and requires card features that will

drastically limit the possibility of leveraging already deployed cards (and also increase cost of the required card). This model will not allow deployment on all cards.

The 'Applet deployed in ISD using applicative security' model has the major drawback that the card platform could be a target of premium attack and cipher clear text attack. The later could pose a threat not only to the card authentication application but also could expose the ISD keys.

5.2.9 Considerations/limitations

Creation of the security domain and personalisation of the scp80/02/03 keys have some limitations. Creation of a SSD and personalisation of keysets have overheads, especially in the case of OTA issuance. Some of the cards which are already on the field might not support creation of SSDs. In case the delegated privileges are to be used for the SSD, only the newer or high end card platforms support this feature. The SSD personalisation of the keyset can be done using PutKey operation, but this is prone to confidentiality issues. One option is to use confidential personalisation (specified in Amendment A of GP); this is only supported in newer NFC/eUICC Cards.

5.3 Data to application: SCP80

The Application Request Format in Mobile Connect between the MSSP and the card authentication application is based on the ETSI TS 102 226 [6] in Compact (mandatory) and Expanded mode (optional).

For the Compact Mode, ETSI TS 102 226 [6] **Error! Reference source not found.**specifies the following f ormat:

Class byte(CLA)	Instruction code (INS)	P1	P2	P3	Data

The above format is considered valid for all the types of APDUs:

- Type 1 (No data in, no data out) – in this case P3 = 0
- Type 2 (No data in, data out) – in this case P3 = Le
- Type 3 (Data in, no data out) – in this case P3 = Lc
- Type 4 (Data in, data out).

For the Type 4 case,

- P3 = Lc;
- A GET RESPONSE (00 C0 00 00 P3) with P3 = Le may follow the previous command to retrieve the data out. In Mobile Connect server/application interaction, the GET RESPONSE command is not required.

5.4 Data from the Application

For Compact mode, the outgoing data from the Application is of the format:

Number of commands	SW1	SW2	Data

For a successful case SW1 SW2 is 0x90 0x00. Error/exception conditions are as per the GlobalPlatform Card Specifications section [7] section 11.1.3 - General Error conditions.

If the data is a set of TLVs, TransactionID is returned in all cases. The MAC is conditional and it is not recommended to return it in the case of error cases to avoid security issues. Other TLVs returned are dependent on the request types.

Here is the summary of the observations and recommendations with respect to PoR and MO SMS for various SPI options which are to be considered:

- The response data can be either sent using PoR or MO SMS depending on the scenario, but PoR is recommended wherever applicable.
- When SPI2 = 0x00 - no PoR is sent.
- When SPI2 not 0x00 a PoR is always sent.
    - When there is no user interaction, the Application response data can be sent using PoR.
    - With applicative security where SPI1 could be 0x00, the response is recommended to be sent using MO SMS since some of the card platforms might reject a PoR to be sent when the incoming request comes with SPI1=00.
- When the user interaction has occurred response data is sent back using MO SMSes. An empty PoR is sent when a PoR is requested.
- In case SPI1=00, application security needs to be always used. It is not recommended to use SPI1 = 0x00.

5.5 The impact of commands on OTA

The table below shows the commands' impact on OTA, such as if 'concatenation is needed', or 'sequence in the same packet'. Note that this table is informative only, the precise number will be changed when the protocol or applicative encryption are applied on the command payload.

No.	Command Name	Maximum input Payload	Maximum Output Payload	Proactive com-mand	Sequence in the same packet	MT Concat-enation needed	MO Conca-tenation needed
1	Sign transaction	5+6+6+2+MSG = 19+MSG	6+6+3+18+3 = 27	Yes	First one and only one	No (MSG>94 not allowed)	No
2	Manage Personal Code	5+6 = 11	6+3 = 9	Yes	Last one	No	No
3	Get Applet Data	5+6 = 11	6+4+4+3+6+3+3+4+12+3+3+2+48+3+X1 = 104+X1	No	Last one	No	Yes if X1>10
4	Set Applet Data	5+6+6+3+3+5+12 = 40	6+3 = 9	No	any	No	No
5	Set Authentication Handler data	5+6+2+2+1+22+10 = 30	6+3 = 9	No	any	No	No
6	Change applet status	5+6 = 11	6+3 = 9	No	any	No	No
7	Delete Authentication Handler	5+6 = 11	6+3 = 9	No	any	No	No
8	Set E2E transport key	5+6+22 = 33	6+3 = 9	No	any	No	No

Figure 5.12 Impact of commands on OTA

- Note 1: For Maximum Output payload, at least
    - 27 bytes footprint in all.
    - 21 in 23.048 layer,
    - 5 in 23040 layer,

- 1 in ETSI TS 102 226 layer.

- Note 2: ‘MSG’ stands for ‘message’, X1 is the total length of ‘FF’ TLV.

6. Test requirements

6.1 Introduction

This section deals with testing the Mobile Connect service, with a focus on the SIM/OTA MSSP mechanisms, first by trying to find the existing tools on the market for which there is test coverage and secondly by assessing which complementary test plans could be useful to achieve the goals of enhanced interoperability in the field.

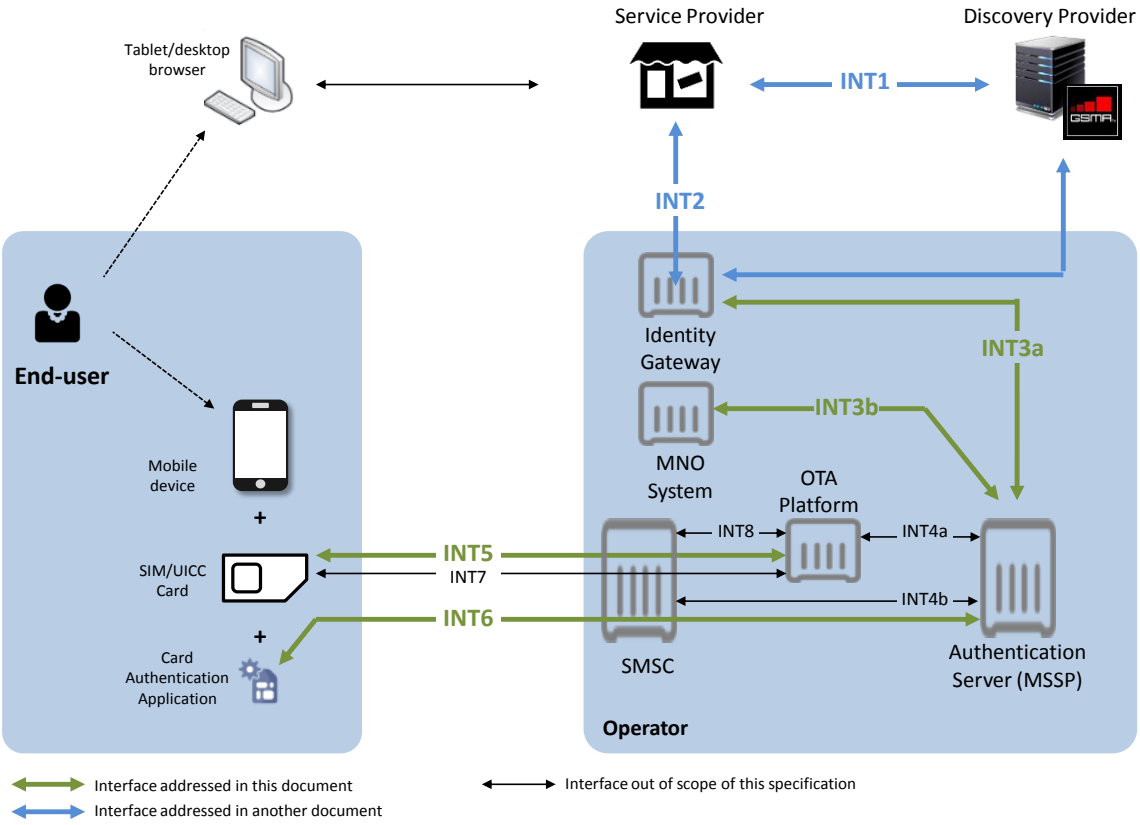

Figure 6.1 Extract of Mobile connect authentication architecture overview (see [1])

As can be seen in this schematic diagram, the number of entities playing an active part in the Mobile Connect infrastructure is important with their corresponding data flows:

- The Mobile network.
- The MNO gateway to the Mobile Connect Service Provider.
- The OTA platform.
- The SMC Centre.
- The MSSP server.
- The handset.
- The UICC/USIM.
- The Applet.

6.2 SIMalliance interoperability testing

Exhaustive interoperability testing can be an extremely complex process due to the combinations of entities involved that can vary from one vendor to the other, as can be seen above. So in this document it is suggested

to restrain the scope of interoperability testing to the sole entities that are within the domain of expertise of the SIMalliance, i.e.:

- The OTA/ MSSP Servers.
- The UICC/USIM platform.
- The Mobile Connect Applet.
- The terminal behaviour on the UICC interface.

This reduces the overall architecture to the following two schemes (Extract from document [1], section 6.3.1).

6.3 Applet deployed in ISD using OTA keys

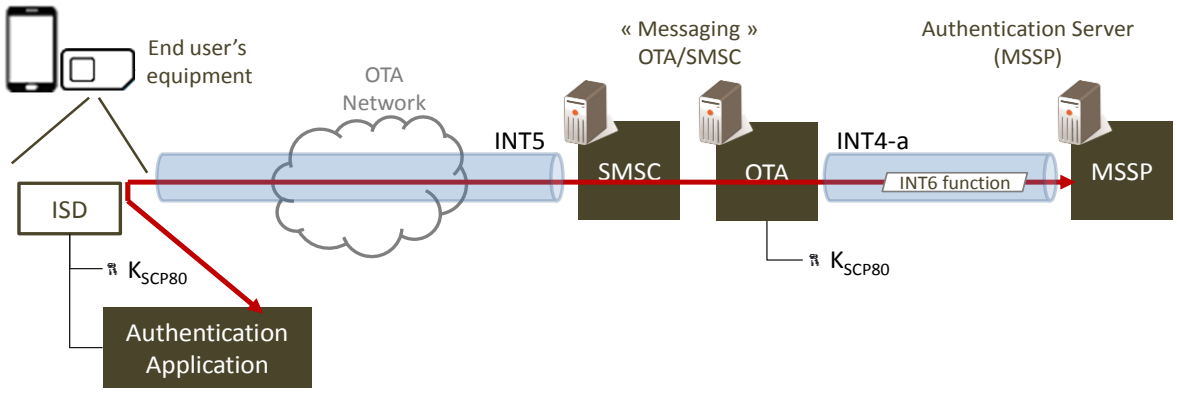

Figure 6.2 Applet deployed in ISD using OTA keys

6.4 Applet deployed in a dedicated SSD using OTA keys

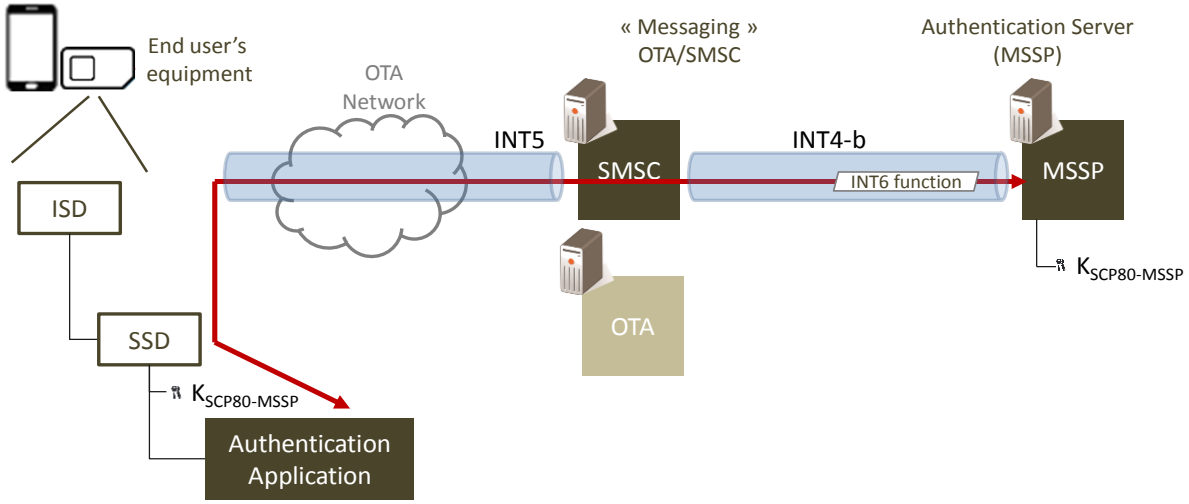

Figure 6.3 Applet deployed in a dedicated SSD using OTA keys

6.5 Existing GSMA test specifications

In this test specification (Document [1] CPAS8 Test Plan for SIM Authentication Specification v 1.2) the *Entity Under Test* is the Mobile Connect Applet. Therefore, the current test specification only focuses on the functional testing of this applet. The following schematic summarises the testing method already specified:

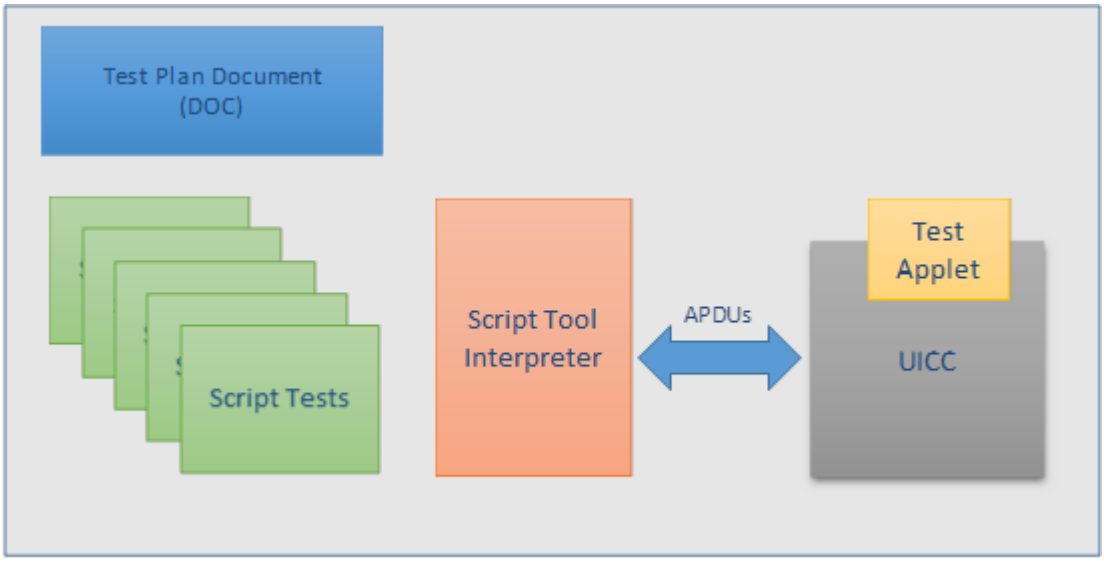

Figure 6.4 Testing method summary

6.6 Existing tools

Within the GSMA there is no certification program associated with the document [1] test plan until now, so even if two tools are mentioned in the document none all of these tools are officially accredited as conformance test tools for the GSMA.

The Movenda Mobile Connect SIM Applet Test Suite is a test solution under the Mobile Connect Vendor License. This allows it officially run specific tests around the applet functionality, MSSP/OTA simulated test and stress interoperability testing. Moreover, it can be interconnected with the real OTA/MSSP Movenda-MCX service.

So in practice, card/applet vendors develop their own testing solution that cover not only the test plan described in document [1] but also include complementary internal tests. On the other side MNOs can run an internal validation session of the acquired applet against their real UICCs by the use of GSMA qualified tools.

The two tools references in document [1] are:

- jCardSim implementation http://jcardsim.org/en
- Movenda-MCX http://www.gsma.com/personaldata/wp-content/uploads/2016/05/Movenda-MCX.pdf

They may be used by MNOs who do not have equivalent test tools:

jCardSim

jCardSim is an open source simulator which implements Java Card v.2.2.1 and which allows tests to be executed against applets in a personal computer with a simulated Java Card environment.

It shall be noted that this open-source simulation tool has some limitations on the versions of Java Card supported and furthermore it has no SIM toolkit support. This makes it not fully suitable as such for complete Mobile Connect Applet testing or even not adapted for UICCs running Java Card from versions higher than 2.2.1.

Movenda

Movenda Mobile Connect SIM Applet Test Suite is proprietary software that allows executing tests against applets running in actual cards. This tool implements the set of test scripts defined within the GSMA "CPAS8 Test Plan for SIM Authentication Specification v 1.2" test plan document. It is possible to create proprietary scripts for MNOs software validation. The scripts are xml-based with a step-by-step user guide. Carrying out these functional tests is a good first step but may not be sufficient when considering the fact that the applet/UICC will be connected through a handset to an OTA/MSSP server. The Movenda Suite has a sequence of non-functional/interoperability.test cases to check specific OTA policy and specific requirements of the MNOs:

- Minimum Security Level and SPI (positive/negative cases).
- Timer expiration event triggers.
- Over the air installation of SIM applet (positive/negative cases).
- Memory leak issue (one command/multiple commands cases).
- Stress test session.

Another point to consider in this Mobile Connect system infrastructure is how easily its elements can be corrected/upgraded after field deployment.

Correcting OTA/MSSP servers after deployment will be a simple software update whereas correcting applets, cards or terminals massively deployed in the field can produce more difficulties.

This constraint implies for the terminal / card /aApplet used the need to ensure deeper testing on these components before rollout and, if possible, the capability to remotely fix them after rollout.

This raises a need for additional interoperability tests regarding the interconnection of the applet to the real OTA and MSSP servers, through real components, in particular about:

- Reliability of the overall solution in real life situations.
- Robustness and efficiency of SMS-PP / and SCP80 implementation by comparison with other secure protocols CAT-TP, GP (SCP81/ SCP 03).
- Testing of potential upgrades or fixes (UICC and applet evolutions, terminal upgrades, migrations of servers, or change of protocol/ security scheme…).
- MSSP Data flow through the OTA-SMS-C to the UICC applet.
- Transparent support of the data by the handset depending on the protocol used.

The Movenda Suite can be inter-connected with real OTA and MSSP servers as for the Movenda-MCX description.

Therefore a three-phase approach of testing can be foreseen that is described thereafter.

6.6.1 Phase1: Functional testing

At this functional level, the tools implement the elementary test cases defined in document [1].

Functional testing of the applet means that for a given input the applet implementations must give consistent responses/errors.

Functional test on a real UICC and Interoperability between applet and UICC platform: the Mobile Connect applet must have the same behaviour whatever the UICC platform.

Phase 1: Functional testing setup:

Entities under test:

- Real or simulated UICC
- Simulated handset.
- Simulated OTA/MSSP.

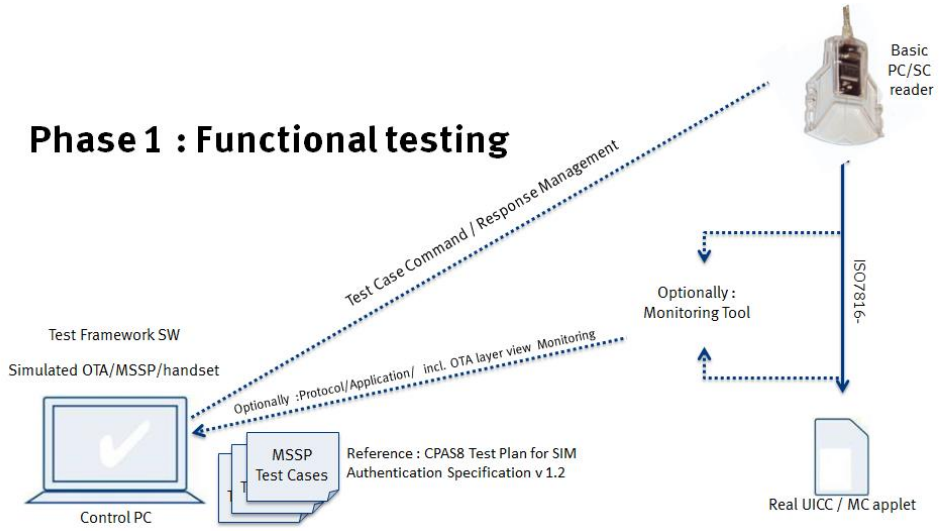

Figure 6.5 Functional testing

Test objectives:

- Applet validation.
- UICC validation (when using real UICC).
- Applet / UICC interoperability testing (when using real UICC).
- Interoperability Test fest events (simple setup).

Test plan: Injection of functional **elementary test cases** as per document [1] for card/applet validation or card/applet interop testing.

Test tool: Tools on the market like JCardSim, Movenda Mobile Connect SIM Applet Test Suite or any equivalent.

6.6.2 Phase 2: Lab testing with simulated OTA/MSSP

At this level the tool does not implement elementary test cases but scenarios with the objective to simulate/reproduce field conditions in a lab.

- OTA/MSSP Lab testing will allow interoperability tests between the applet and simulated OTA servers using real UICCS and real terminals: the Mobile Connect applet must be able to establish a reliable connection to an MSSP server, whatever the UICC and the terminal are.
- This lab setup makes it possible to replicate real-life scenarios prior to field deployment or in parallel to field deployment for troubleshooting purposes

Phase 2: Lab testing with simulated OTA/MSSP setup:

Entities under Test:

- Real UICC.
- Real Handset.
- Simulated OTA/MSSP.
- Simulated network.

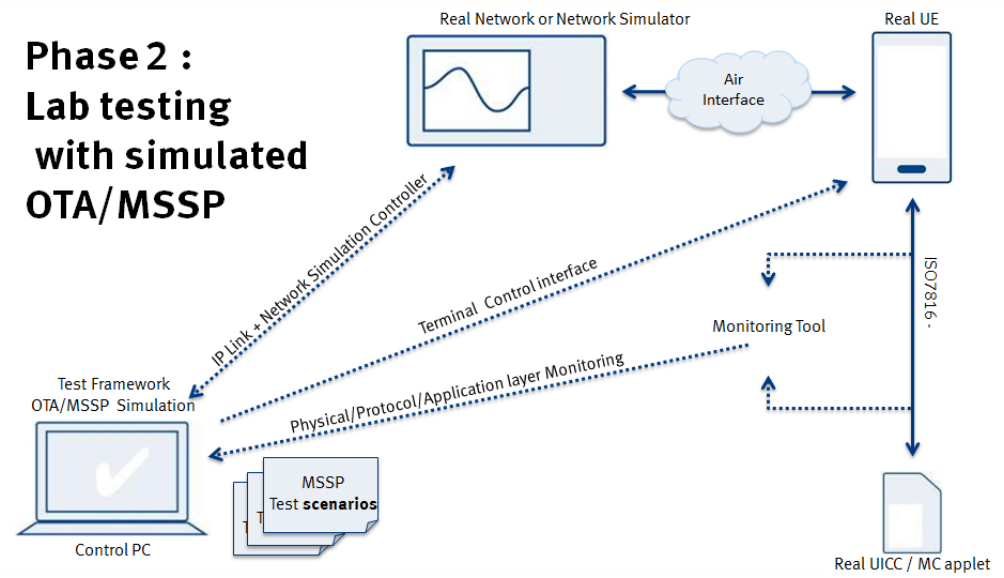

Figure 6.6 Lab testing with simulated OTA/MSSP

Test objectives:

- Protocol and security check to test optimal data channel: protocol parametering, protocol switch (SMS-PP / CAT-TP / GP Amd. B).
- Correct support of Proof of Receipt (POR) mechanism management by OTA and card.
- Secure channel parametering and switch (SCP80 / SCP 81 / SCP 03).
- Terminal testing: validation of compatible phones (possibly using simulated UICCs).
- Troubleshooting by reproducing in a lab the same setup as in real-life but with NW and server simulation: creation of specific test cases / scenarios / fault injections, potential reuse of real MSSP scenarios in a lab environment.
- Applet and UICC remote update or correction.
- Possible Test Fest sessions (simplified setup compared to phase 3).

Test plan: injection of key scenarios created in a rich text-based format (like XML).

Test tool: Tools on the market like Movenda Mobile Connect SIM Applet Test Suite (as described above), COMPRION Otability or any equivalent.

6.6.3 Phase 3: Pre-production testing with real OTA/MSSP setup

At this field level, the test implies using real UICCs, real terminals and connection to a pre-production replication of the real OTA/MSSP servers to be on the field. The tool prepares commercial deployment by injecting the same test scenarios that were executed in a lab (option 2).

Phase 3: Real end-to-end testing:

Entities under Test:

- Real UICC.
- Real Handset(s)
- Connection to a pre-production real OTA/MSSP.
- Real or simulated network.

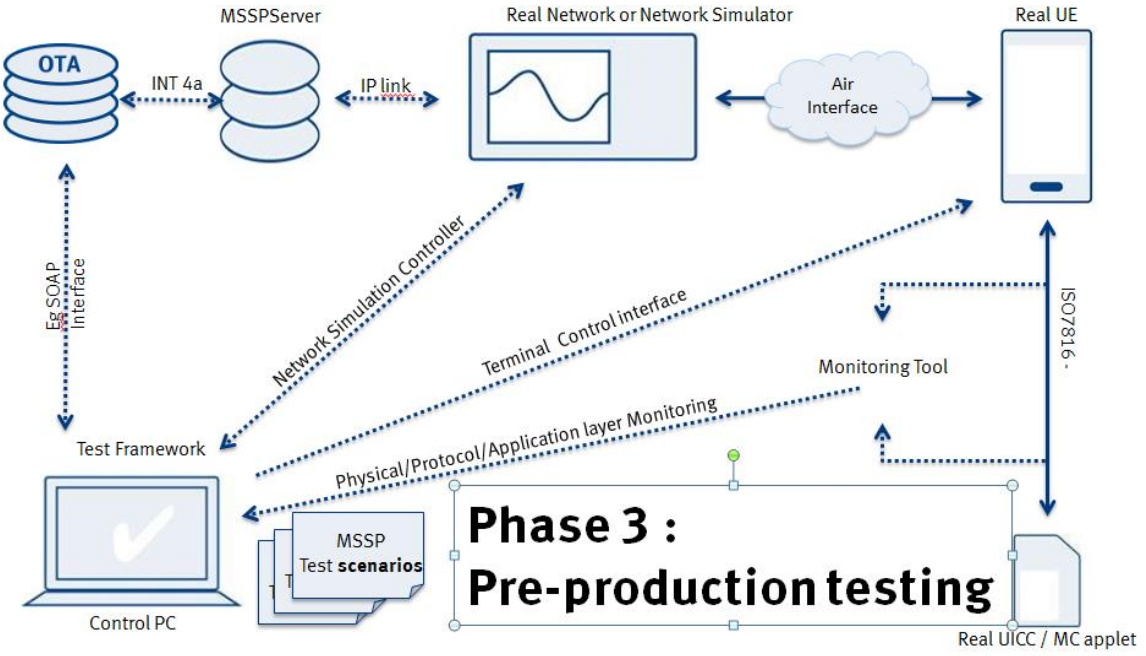

Figure 6.7 Pre-production testing

Test objectives:

- Preparation of field deployment through realistic test scenarios using real components.
- Applet and UICC remote update or correction using real infrastructure.
- Possible test fest sessions (but more complicated setup).

Test plan: Injection of test rich text based scenarios reflecting real-life situations. Selection of key scenarios to be specified.

Test tool: Tools from the market like Movenda Mobile Connect SIM Applet Test Suite (as described above), COMPRION Otability or any equivalent.

Annex A (informative): Assigned TLV tag values

Name	Tag	Length	Note
TransactionID	01	4	
TransactionDateTime	02	4	
End-user response status	10	1	
Message Authentication Code	11	8 or 16	
Message	8D	0-94	
Supported Authentication Handler types	A0	2	
GSMA version	A1	2	
Activation flag	A2	1	
Installation date	A3	4	
Max PC attempt	A4	1	
PC length	A5	1	
Time Out	A6	2	
MSSP Address	A7	2 to 12	
E2E transport key flag	A8	1	
E2E transport key type	A9	1	
Authentication Handler identifier	AA	1	
Authentication Handler state	AB	1	
Encryption key	AC	16	
Authentication key	AC	16, 20, 24,	
Counter	AD	8	
List of Authentication Handler descriptors	B0	var	
Authentication Handler type tag	Bx	6	
Tag reserved to describe proprietary features	FF	var	